Test-Driven Learning in High School Computer Science

Ryan Stejskal
University of Nebraska at Omaha

Test-Driven Learning in High School Computer Science

A Thesis Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements of the Degree

Master of Science

University of Nebraska at Omaha

by

Ryan Stejskal

April 2014

Test-Driven Learning in High School Computer Science

Ryan Stejskal, MS

University of Nebraska, 2014

Advisor: Dr. Harvey Siy

Abstract

Test-driven development is a style of software development that emphasizes writing tests first and running them frequently with the aid of automated testing tools. This development style is widely used in the software development industry to improve the rate of development while reducing software defects. Some computer science educators are adopting the test-driven development approach to help improve student understanding and performance on programming projects. Several studies have examined the benefits of teaching test-driven programming techniques to undergraduate student programmers, with generally positive results. However, the usage of test-driven learning at the high school level has not been studied to the same extent. This thesis investigates the use of test-driven learning in high school computer science classes and whether test-driven learning provides benefits for high school as well as college students.

Acknowledgements

This thesis would not have been possible without the support of many people. I would like to thank the following people who aided me in this work:

My committee members, Dr. Harvey Siy, Dr. Brian Dorn, and Dr. Nealy Grandgenett, who took much time out of their busy schedules to give advice and assistance.

My colleagues at Westside High School who provided suggestions, and also helped me keep my class work under control as I worked on this thesis.

My family, who helped make me the person I am and have always supported me in all my endeavors.

And finally, my students in AP Computer Science, without whose cooperation there would have been no study and no thesis, and who frequently surprise me with the depth of their thinking and the strength of their enthusiasm in AP CS.

Grant Acknowledgement

A portion of this research was supported by the National Science Foundation through the Research Experiences for Teachers program under Grant No. CNS-1201136.

Table of Contents

List of Tables

List of Figures

Chapter 1 - Background

Introduction

Test-driven development (TDD) is a style of software development that emphasizes writing tests for the software being developed. In a TDD style, the programmer creates tests for code being developed prior to writing the code itself. Tests created while using the TDD methodology are run frequently and repeatedly during the development of software, typically with the support of automated testing tools. The typical development cycle when using TDD is summarized as "red-green-refactor" [2]:

- Write a new test case for the feature being developed.
- Run all test cases (both existing test cases and the newly created test) to verify that the newly created test fails. This is the "red" phase of the development cycle: a failing test case is typically marked in red by the automated testing tool (while a passing test is marked in green).
- Write enough code to allow the application to pass the newly created test case, while not failing any other existing test cases. This is the "green" phase.
- Refactor the code to remove any duplication introduced by the newly written segment, while executing all test cases to ensure that they remain "green."

Although TDD is an inversion of the more common "test-last" methodology to which most programmers are accustomed, this style of development provides specific benefits. Since TDD emphasizes a short development cycle where the amount of new code under test at any one time is small, tests are more likely to uncover problems as the software is developed [22]. In addition, since test cases are added to the application's complete test suite rather than run once and discarded, TDD inherently provides for regression testing as new features are added to an application. Also, as programmers write tests in advance of writing production code, they are led to think more specifically about the system requirements. Writing tests helps to clarify programmers' mental model of the system and ensure that requirements are clear prior to writing production code.

The TDD style is a fundamental practice within some software development methodologies, such as Extreme Programming [13] and Agile [22]. However, TDD can also be used alone, in the absence of any particular development methodology. TDD is a known and accepted development practice in the CS industry, and has been used by teams at Microsoft [3] and at NASA [21].

The use of TDD principles in computer science education is less widespread, but test-driven learning (TDL) has been studied at the university level by a number of researchers. College course instructors who have introduced TDL into their courses have found generally positive results [7] [26] [27], although students' willingness to adopt TDL and their ability to apply it effectively vary depending on where TDL is introduced and the means in which it is employed [14] [23]. Students in courses where TDL is employed typically write programs that contain fewer defects and are better organized and documented [7].

Statement of Problem

While test-driven learning has been studied in several different settings at the college level, its use in high school computer science education remains relatively unstudied. This thesis studies the usage of test-driven learning in high school computer science; in particular, within the Advanced Placement (AP) computer science curriculum.

The goal of the thesis is to investigate whether the successful use of TDL at the college level can be replicated for high school students.

Hypotheses

To that end, this study considers the following hypotheses:

Hypothesis 1: High school students can successfully apply test-driven methodologies in a high school computer science course.

While research finds that test-driven learning can contribute to the success of college students in a college course, there are some additional obstacles that apply to applying test-driven learning in high school courses. For example, many high school students in computer science are just starting out with the subject. The additional time and attention investment of learning TDL along with the rest of the material of the CS course itself may pose an additional obstacle to using TDL in high school.

Hypothesis 2: Student learning will be positively impacted by the use of test-driven learning in the high school CS curriculum.

The ultimate measure of any instructional method is its impact on student learning. This thesis evaluates whether applying TDL in the high school CS curriculum will help students better understand programming assignments and create programs that fulfill project requirements. Again, results in college courses suggest but do not guarantee that there will be benefits for high school students using TDL.

Hypothesis 3: Students will report that they feel there are benefits from using test-driven development.

In addition to the objective benefits that may be realized by implementing TDL, this thesis also evaluates high school students' attitudes toward TDL. If students have a positive experience when using TDL and express positive attitudes toward the technique, then it may be more likely that those students will continue to use TDD techniques as they continue to study computer science in college and beyond.

Overview of this Thesis

This thesis has been organized in five chapters. The first chapter is this introduction. In chapter 2, an overview of research related to TDD and TDL is presented. The primary focus is on the results of using TDL in college courses. Information on the use of TDL at the high school level was not found during the literature review, which was one motivation for this study. Chapter 3 presents the setting and design for this research. The author of this thesis is a high school computer science teacher, and this study was conducted in the author's AP CS class. The background of the school and the AP CS class are presented, as well as an overview and logic model of how TDL was inserted into the class and the methods in which data was collected regarding TDL. In chapter 4, data collected during the course of the study is presented and analyzed in the context of the hypotheses. Chapter 5 presents conclusions as well as recommendations for future teachers considering applying TDL in their courses.

Chapter 2 - Literature Review

The Test-Driven Development Process

Test-driven development is a software development process in which test cases are written before related features of software under development. As described by Madeyski [22], before a programmer changes the behavior of production code for software under development, the programmer must first have a failing test. Test-driven development is also known as "test-first" development; the latter term is sometimes used to emphasize the distinction with the more traditional "test-last" practice of software development.

Beck [2] describes the TDD process using a traffic light metaphor, with the mnemonic "red-green-refactor." The first step of TDD development is to create a test case for a feature being developed. Since the production code for this feature has not been written at this point in time, the test case is expected to fail (and if the test case does *not* fail, then it is assumed to not be a valid test case). This phase of the process is the "red" phase, commonly emphasized by automated testing tools displaying failing test cases with a red indicator. After creating the failing test case, the programmer then implements the feature to be developed, generally preferring the quickest or most obvious implementation that will allow the test case to pass. When the test case passes, development reaches the "green" phase of TDD; automated testing tools will typically display passing test cases in green. In addition to the new test case, previously created test cases must remain "green" in order for development to reach the green phase. After all test cases pass, then the developer may move on to the "refactor" phase. In this phase, the developer cleans up the created code, refactoring to eliminate code duplication and ensure that the newly created code meets standards. During the refactoring phase, the programmer seeks to finalize the code quality of the newly implemented feature, using knowledge gained from the first implementation of the feature to help make the code better. The programmer will continue to use automated testing tools during the refactoring phase, and if any test case fails at this step, the programmer must repair or revert any refactoring that has been done to return the software to a point where all test cases are passed.

Another important aspect of Beck's conception of TDD is that the red-green-refactor cycle should be very rapid. One important benefit of the TDD development cycle is that it allows programmers to proceed in very small incremental steps. For the red phase, Beck recommends that a developer should create only one new test case at a time, preferring whenever possible to work on a facet of the system's functionality in which the programmer feels confident in being able to both test and implement quickly. When implementing code corresponding to the new test case, the programmer's goal should be to turn the test "green" as quickly as possible, even at the cost of clean code. The programmer should choose an "obvious implementation" of the new functionality when that implementation is in fact obvious to the programmer, but if the implementation is not obvious, the programmer will then "fake it." By this, Beck means that the programmer should choose *any* implementation that will make the just-written test case pass, even simply making a method return the constant value that the test case expects. Only in the refactoring stage does the programmer clean up this "fake" implementation, incrementally modifying the functionality of the feature toward a correct implementation while verifying that the newly-written test and all existing tests continue to pass.

TDD is commonly used in conjunction with other software development practices, especially Agile development [22]. In particular, TDD is a component of the Extreme Programming process [13]. This process also includes other specific developer behaviors, including Pair Programming. While TDD is commonly used in conjunction with other aspects of Extreme Programming, it can also be adopted as a stand-alone development practice.

Test-Driven Development in the Software Industry

Test-driven development is a fairly well-known process in the software industry. Bhat and Nagappan [3] performed case studies at Microsoft involving two software development teams, one working on Windows networking and the other on MSN, which used TDD practices. For both projects using TDD, the defect rate of the produced software was lower than in a comparably-sized project where TDD was not used. For the Windows networking project team, the non-TDD project showed a defect rate per 1000 lines of code 2.6 times as high as for the project where TDD was employed. For the MSN team, the non-TDD project's defect rate per 1000 LOC was 4.2 times that of the TDD project. While both projects showed increased development time when using TDD (estimated 35% and 15% additional time respectively), this additional time investment was judged to be a worthwhile trade-off for the increase in resulting code quality.

Hammond and Umphress [13] examined the practice of test-driven development in the industry, and found that while many developers are familiar with the concept of test-driven development, the level of actual adoption in industry is somewhat difficult to gauge. Because test-driven development is often associated with other development practices such as Extreme Programming or Agile development, it is difficult to isolate the usage of test-driven development alone as a practice. For example, in a 2008 survey of programmers already using Agile, around 70% of programmers state they use TDD [1]. Mainstream polls show significantly lower adoption rates, but still enough to justify TDD as a technique for software development. Another issue is that the TDD process is not precisely defined within the industry, and different developers take the term itself to mean different things [13].

Test-Driven Learning in Education

Test-driven learning and its application in education has been researched in a number of university courses at several different universities. Although the tools and many of the procedures in using TDL in a course setting are identical or similar to those used in industry TDD, the goals are somewhat different. In particular, computer science courses, due to their limited duration, place less emphasis on maintenance and long-term development than in industry. The size of projects also tends to be smaller in CS courses than in industrial projects. Even in project-oriented courses such as software engineering or capstone courses, the scope of course projects is necessarily limited both by the time duration of the course and the relative inexperience of the student developers [17]. Still, many studies examining the use of TDL in academic settings have found positive results and benefits.

Janzen and Saiedian [16] have applied TDL in CS1 and CS2 courses at the University of Kansas. They did find that students who adhered to the test-first methodology scored higher on most programming project grades than students using a test-last approach. Students using TDL were also more productive, spending less time on project assignments and writing code with fewer errors. However, they also found that many students were reluctant to adopt TDL in their work. This study used the C++ language, and the authors speculate that reluctance to adopt TDL may be related to the rudimentary style of automated testing (assert statements) used in the studied courses.

Further research by Janzen and Saiedian produced conflicting conclusions on whether student programmers are more likely to adopt test-first techniques when those techniques are introduced early or later in the CS curriculum. In a 2006 study [14], they found that student resistance to TDL is generally lower when TDL is introduced early in the curriculum, as more experienced programmers are likely to prefer to continue to use more familiar test-last methods. Due to this factor, they recommend that teaching novice programmers using TDL may have a greater impact. However, in another study conducted in 2007 [15], they found that students and industry programmers with more experience were likely to express positive opinions of test-first development. For both new and experienced programmers, programmers were more likely to express positive opinions of test-first development after having tried it in at least one project.

Melnik and Maurer [23] investigated the use of an automated testing tool, Framework for Integrated Testing (FIT), to help specify project requirements for student programming projects. FIT is normally used as an acceptance testing framework; its role when used in industry is to allow non-programmers to help create acceptance tests that confirm that a system conforms to specifications. In this investigation, FIT and a related tool, Fitnesse, were used along with JUnit in two upper-level courses: a software testing and maintenance course and a web-based systems course. Three approaches to specifying projects with FIT were investigated: the instructor providing a complete test suite for the project, the instructor providing a partial test suite which students were required to extend, and the instructor assigning students to write test suites in teams which were then exchanged with other teams. Full student-written test suites were generally found to be less successful, as students' ideas of what the test suite should actually test varied widely. However, students were very receptive to using instructor-created test cases as a complement to the prose description of the assignment, and a large majority (80%) stated that they preferred having the assignment specified using acceptance tests rather than entirely with text.

Lappalainen et al [20] found that while there were benefits to using tools such as JUnit, the introduction of JUnit in an early programming course can be difficult due to creating an additional learning burden for students. Asking students to use TDL requires learning both the technical details of creating test cases as well as the need to determine what a test case should actually test. To lessen the cognitive load of TDL for novice programmers, a tool called ComTest was developed, along with a plugin for the Eclipse IDE [11]. This tool allows students to write test cases within JavaDoc comments, using a macro language and a special @example tag to identify the test cases.

```
/**
  * @param n number to be studied
  * @return smallest divisor for n, 1 if n is a prime
  * @example
  * <pre name="test">
  *   smallestDivisor(2) === 1;
  *   smallestDivisor(4) === 2;
  *   smallestDivisor(5) === 1;
  *   smallestDivisor(6) === 2;
  * </pre>
  */
public static int smallestDivisor(int n) {
     for (int i = 2; i <= n / 2; i++)
          if ( n % i == 0 )
               return i;
     return 1;
}
```

Figure 1: Sample ComTest unit test using @example tag

The goal when creating this tool was to shorten the amount of code required to create test cases, as well as to place the test cases physically close to the tested code. In ComTest, test cases appear in a JavaDoc comment immediately above the function being tested, whereas in JUnit, test cases are generally placed in separate files from production code. While the intermixing of test and production code would generally be considered undesirable in an industry setting, this concern was considered less important in a CS course. The use of ComTest was investigated in two CS1 and two CS2 courses. In the CS1 courses, students were not required to write their own tests, but had the opportunity to do so as a bonus on their assignments. CS2 students were required to write unit tests and were allowed to choose to use either ComTest or JUnit. While the CS1 students were somewhat uncomfortable writing their own unit tests using the ComTest tool, they did take advantage of tests that were provided by the course instructor. CS2 students generally were found to write more unit tests with ComTest than with JUnit and to show less resistance to using unit testing with ComTest.

Briggs and Girard [6] took a similar approach when introducing TDL into their CS1 course. Based on evaluation from a lab experience, they found that students were able to adapt to a test-first approach and that TDL improved the quality of students' submissions. However, while their students were able to use instructor-provided JUnit test cases easily, those students had more difficulty in creating their own test cases with JUnit. Therefore, the authors created a plugin for the Eclipse IDE that they used in their CS1 class. This plugin used a wizard interface to help students create a JUnit test case. The plugin would generate the test case based on user input for the inputs and expected results of a method to be tested, and would also perform the background tests of setting up the student's project to link to JUnit and to allow the student to run the generated JUnit tests. By having their plugins take responsibility for the mechanics of setting up JUnit, the authors stated that students would be free to generate tests without needing to worry about generating code.

Test-driven learning is used heavily by Buffardi and Edwards in teaching computer science courses at Virginia Tech [7]. Here, JUnit is used as a TDL tool in both CS1 and CS2 courses. Students are introduced to using test-first techniques early in the CS1 course, and these techniques are used throughout both CS1 and CS2. Further, in the second course, an automated system called Web-CAT [32] is used by students to submit their programs. Web-CAT uses instructor-provided JUnit tests, as well as other program analysis techniques such as code coverage, to evaluate student project submissions and provide feedback to students. Students may submit their program to Web-CAT an unlimited number of times, with the system providing additional feedback as students improve the quality of their submission. As has been found by other authors, some students in the CS2 course adopt test-first programming styles more readily than others. Buffardi and Edwards found that students who consistently adhered to TDD techniques produced solutions with higher correctness, as well as with a higher level of code coverage by test cases. Interestingly, their study also found that students who tended to procrastinate and start assignments late were less likely to use test-first development style, as demonstrated by Web-CAT submissions that showed that these students were more likely to write all their production code first and then create test cases.

In another study [8], Buffardi and Edwards investigated further why some students readily adopt test-first practices while others are reluctant to do so, using in-depth student interviews to investigate students' development habits. Seven students were interviewed in a group setting for about an hour to investigate their development strategies and motivations in their CS2 course. The interviews showed that students used a wide variety of testing strategies when working on course assignments. Four of the seven students said that they generally wait until most of the source code for their project is complete before testing. Three of those said that even though they understood that their instructor encouraged test-first practices, it didn't make sense to them to test until their code was substantially complete. The remaining students interviewed didn't consistently follow one testing strategy, instead varying their technique from one assignment to another. The students were also split on their preference for increment size in code writing: four preferred to write and test in large increments, two preferred small increments, and the remaining one alternated between both strategies. One additional finding of this study was a correlation between students' confidence in their own programming skills and preference for a test-last development cycle. While the sample size for this particular study was small, this correlation appears to be consistent with findings in earlier studies that students who are more experienced programmers and who are more advanced in the computer science curriculum tend to express more resistance to adopting TDL.

Summary

Overall, most studies that have examined the use of TDD and TDL have found that programmers that adhere to these methodologies find them beneficial. Industry programmers generally report fewer defects when using TDD, and student programmers who use TDL techniques create higher quality code.

However, TDD and TDL are not without their costs. A reluctance to adopt test-driven and test-first approaches is common among programmers who have previously programmed using test-last techniques, though resistance can sometimes be overcome by exposing students to TDD and demonstrating benefits to TDD techniques. This reluctance is generally reported to be higher among programmers with more experience and more confidence in their own skills. Novice programmers and programmers who do not have as much confidence in their own skills may be more open to trying TDD. The novice programmer, with fewer entrenched behaviors during development, can be more likely to be open to non-mainstream approaches. The uncertain programmer may appreciate the small incremental cycle of development that is inherent in TDD as well as the extra checks of correctness that automated tests provide.

In the academic arena, research in the use of TDL focuses virtually exclusively on the university setting. There does not appear to be significant research in the use of TDL by high school computer science courses. There are several potential reasons which may contribute to the lack of study of TDL in high schools:

- High schools typically offer only a small handful of computer science courses, if any are offered at all. Many smaller high schools simply do not have the resources to offer computer science courses [19].
- Many high school computer science teachers are not formally trained in computer science themselves, and may not be familiar with TDL. In many cases, high school teachers of computer science specialize in other areas, such as math or business, and teach CS courses due to being the most knowledgeable teacher in the field rather than being specifically certified or qualified in CS [10].

- Researchers of TDL are mostly located at universities and teach university CS courses, so it is natural that researchers would focus on TDL in their own course or other courses at their own institution.

Although the benefits of TDL in high school CS have not been significantly studied, it seems plausible to conjecture that high school students would experience similar benefits to college students from the use of TDL. This is especially true for AP Computer Science courses, which are designed to be similar in scope to college CS1 courses. The curriculum specified by the College Board for an AP Computer Science class and for the AP CS test models a typical university CS1 class curriculum. Since students in AP CS are studying at a college CS1 course level, tools and techniques that help college CS1 students learn seem likely to help AP CS students learn as well.

Chapter 3 - Research Design

Introduction

This chapter describes the methodology of the study, including a description of the subject school and course, instrumentation, and instructional procedures used during the course of the study.

Description of Westside High School

The study was conducted at Westside High School, a public high school located in Omaha, Nebraska. Westside High School is the main high school in Nebraska's District 66, which also includes ten elementary schools, one middle school, and a Career Center alternative high school environment. High school enrollment typically ranges around 2000 students (1934 students enrolled in grades 9-12 for the 2012-13 school year [24]). Of those students, 79.5% were white and 20.5% were not. 26.1% of students received free or reduced-price meals and 0.9% were English language learners. The average composite ACT score of students taking the ACT in 2012-2013 was 24.2. Approximately 37% of enrolled students live in other districts and attend Westside through option enrollment [33].

Westside is one of a small number of high schools nationwide that follows a modular schedule system. In this system, the school day is divided into ten periods of approximately 35-40 minutes each. However, students have unique schedules for each day of the week. Some courses meet more or less than once daily, and some classes may extend for 1.5 or 2 periods at a time. The course in the study, AP CS, meets for one mod per day (approx. 35 minutes), five days a week.

Westside's District 66 offers a one-to-one laptop program, in which laptop computers are checked out to all middle and high school students for the duration of the school year. For the 2013-14 school year, these laptops are Apple MacBooks. Hardware and software support for student computers is provided by a tech support department in the school building. Tech support provides a central avenue by which all students enrolled in a given course can receive the same software installation. For AP CS, tech support installed the Java development platform and the NetBeans IDE.

Computer Science at Westside High School

Westside High School offers three computer science courses, all of which are year-long, elective courses. The first course offered is Introduction to Computer Science, open to any student who has completed Algebra 1. This course was taught using the FutureBasic language (a dialect of Basic) until the 2013-14 school year. Starting in 2013, this course has transitioned to Python. Enrollment is typically about 30 students per year.

Students who have completed Introduction to Computer Science may continue to take either AP Computer Science or C++ Programming, each of which has about 10 to 15 students enrolled per year. As stipulated by the College Board, which administers the AP CS test [9], AP Computer Science is taught in Java. AP Computer Science, in addition to being an Advanced Placement course, offers students the opportunity to earn college credit through a dual enrollment program with the University of Nebraska at Omaha. The UNO equivalent course for which students receive credit is CIST 1400, Introduction to Computer Programming [31].

Officially, there is no prescribed order between these two courses, though both list Intro to CS as a prerequisite. However, some guidelines apply. Students who definitely plan to take both courses are encouraged to take C++ first to acquire an extra year of computer science experience before taking the AP course. Conversely, students whose schedule will only permit them to take one of the two advanced computer science courses are encouraged to enroll in AP CS, due to the additional benefits of the dual enrollment and AP test options. Seniors (and only seniors) are permitted to take both courses simultaneously. Younger students are restricted to taking only one of the two courses at a time, to avoid students overloading their schedule on computer science courses.

In both AP CS and C++, students use the NetBeans development environment [25] to create code. NetBeans is installed on the laptops of students enrolled in either CS course by Westside's tech support team. NetBeans was chosen as a development environment due to its free software status and due to the fact that it supports both Java and C++, allowing students who are taking both courses (sequentially or simultaneously) to transfer knowledge about the IDE from one course to the other.

In addition to these computer science courses, Westside also offers several other technology-related courses. The business department offers Information Technology, a one-semester computer usage course devoted primarily to the teaching of software applications. This course is a requirement for graduation at Westside. DigiTools, another one-semester course, is a continuation of Information Technology, focusing on image editing, web page design, and media editing programs. The business department also offers a one-semester Mac OS X course, which teaches troubleshooting and administration of Macintosh computers.

Two one-semester courses on robotics are offered through the Engineering Tech department. In these course, students work with robots such as CEENBots, along with associated programming, with a view toward participating in robotics competitions. The Engineering Tech department also includes a Macintosh Technician course, which is a subsequent course to the Mac OS X course offered through the business department.

AP Computer Science Curriculum

Test-driven learning was added to the AP Computer Science class at Westside. This class was chosen for TDL implementation due to several factors:

- Automated testing tools are widely available for Java.
- It was desired to avoid using TDL in the Introduction to CS course since students in that class are not expected to have any prior programming experience.
- The existing AP CS curriculum allowed for the addition of TDL with minimal disruption to other material covered in the course.

The overall outline of AP CS at Westside closely follows the standard curriculum for AP Computer Science as specified by the College Board, but continues to teach the AP CS AB curriculum rather than only the CS A curriculum. The course is divided into twelve chapters, averaging approximately three weeks of class time per chapter. The main topics of these chapters are listed in Table 1 below.

Table 1: Chapter topics in AP Computer Science at Westside

Chapter	Main topics	Chapter	Main topics
Fall Semester		*Spring Semester*	
1	Introduction to Java, Java syntax	7	Interfaces, exceptions
2	Classes and objects	8	GridWorld (the AP CS case study)
3	Creating GUIs with NetBeans GUI editor, event handling	9	Stacks, queues, trees
4	Arrays, ArrayList, LinkedList	10	Sets, maps, hashing
5	Inheritance	11	Computers and society
6	Recursion, sorting and searching algorithms	12	Final project

Preliminary Study

A preliminary investigation of the use of TDL in the AP CS course was made during the 2012-2013 school year [30], based on a lesson design created through a Research Experiences for Teachers program held at the University of Nebraska - Omaha during the summer of 2012. In this study, TDL techniques were introduced to AP CS students during the fall semester in a multi-phase approach:

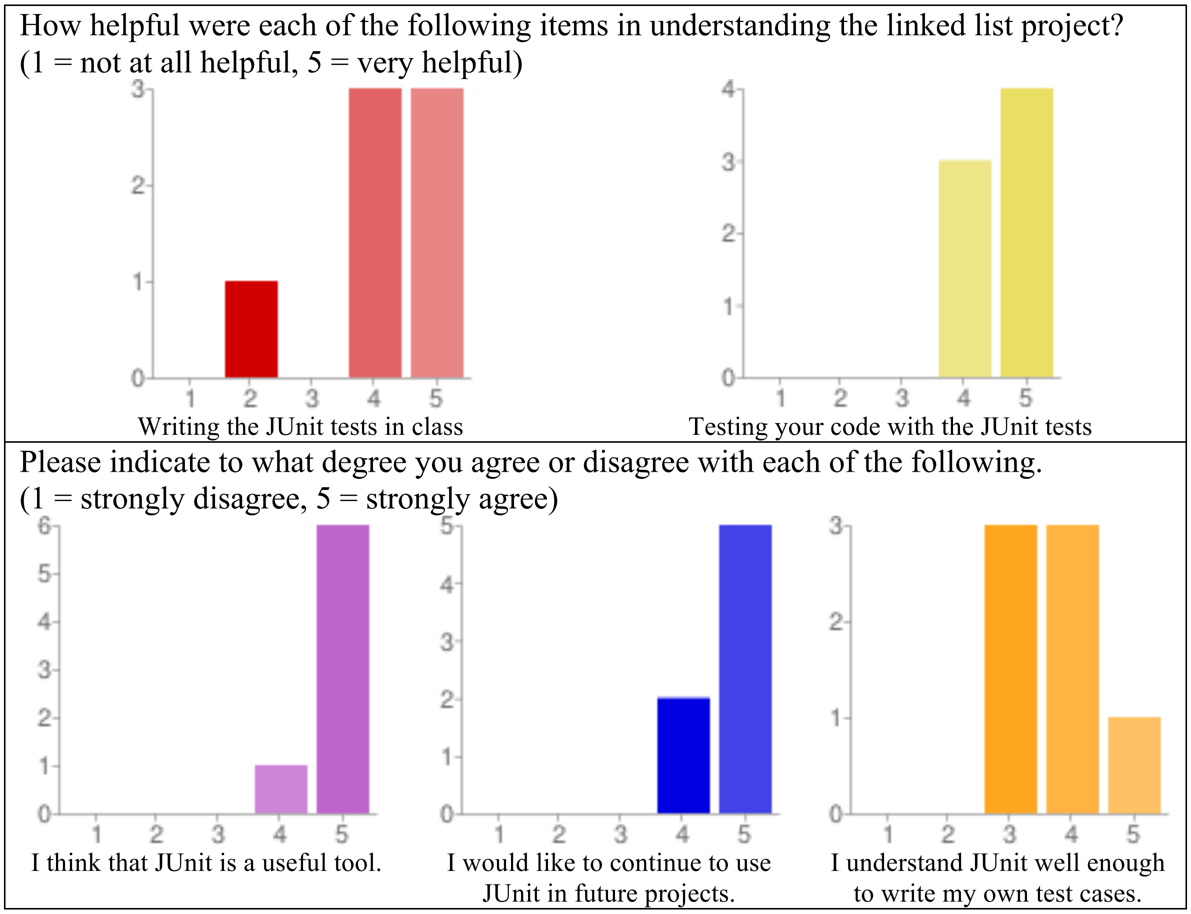
- In the first phase, students were introduced to the concept of testing and test cases. The online tool CodingBat was used to make this introduction. CodingBat is a website that teaches Java and Python through the use of short programming problems. Users create functions to solve the problems in their chosen language, and the functions' correctness is verified by automated testing. Since like most testing tools, CodingBat displays passed tests in green and failed tests in red, this tool helps introduce students to the concept of automated testing. CodingBat was introduced during chapter 1 of AP CS, since students can use the site effectively as soon as they understand Java syntax.
- Next, students were introduced to JUnit by having test cases provided to them as part of a project specification. This phase was implemented during chapter 2, both in an in-class practice and as part of the project specification for this chapter. The intent here was to familiarize students with the use of JUnit as a tool, as well as to expose students to JUnit syntax. This phase was made significantly easier due to the use of NetBeans, as the IDE has built-in support for JUnit.
- In the next phase, the roles were reversed: students were given code that had been written by the instructor, and asked to use JUnit to develop test cases for that code.

This phase of the study was implemented during chapter 4 of the AP CS course. (JUnit use was not used in chapter 3 due to additional difficulty of using unit testing to test GUI code.) The goal of this phase was to allow students to experiment with the syntax and meaning of JUnit, as well as to introduce concepts of code coverage and edge cases to help students create more complete test suites.

- Finally, students were asked to develop both source and test code. This phase was implemented in the project of chapter 5 of the AP CS course. This project requires students to implement their own linked list class. Past experience has shown that this project is typically very difficult for AP CS students, relative to other project assignments in the course. In the past, students have reported that the difficulty primarily stems from the abstract nature of the assignment. Since a linked list implementation usually either works completely or fails (often with a difficult-to-track NullPointerException), students get little feedback from a typical "test by running" strategy. Some test cases were created in class, while students were required to create others as part of the assignment. The intent was that by planning and creating test cases before creating the list itself, students would build a better mental model of how the linked list operates internally, as well as be able to use test cases to help isolate any failures in their own code.

Preliminary Study Participants and Results

Students enrolled in AP Computer Science at Westside during the 2012-13 school year were asked to participate in the preliminary study. Seven students were enrolled in AP CS during this school year. Of those seven students, three were 10th grade students and the remaining four were 12th grade students. Five students were male and two were female. Students were not offered compensation in exchange for participating. A survey was given to these students immediately after the completion of the linked list project (the final phase of the preliminary study). This survey was administered through Google Docs, ensuring anonymity for students completing the survey. All seven enrolled students consented to participate.


How helpful were each of the following items in understanding the linked list project? (1 = not at all helpful, 5 = very helpful)

Writing the JUnit tests in class

Testing your code with the JUnit tests

Please indicate to what degree you agree or disagree with each of the following. (1 = strongly disagree, 5 = strongly agree)

I think that JUnit is a useful tool.

I would like to continue to use JUnit in future projects.

I understand JUnit well enough to write my own test cases.

Figure 2: Summary of student responses to preliminary TDL study survey

The results of this survey were generally very positive. Students were asked to rate the usefulness of various information given to them regarding the project on a 1 (least useful) to 5 (most useful) scale. In addition, they were asked to what extent they agreed or disagreed with three JUnit-related statements on a 1 to 5 scale (1 = strongly disagree, and 5 = strongly agree).

Students on this survey indicated that they found the use of JUnit in this project useful. They also showed strong agreement that JUnit was a useful tool, although they didn't yet show strong confidence in their ability to write their own tests. Some highlights of this survey are shown in Figure 2. The full results of this survey are included in Appendix D.

Main Study Design

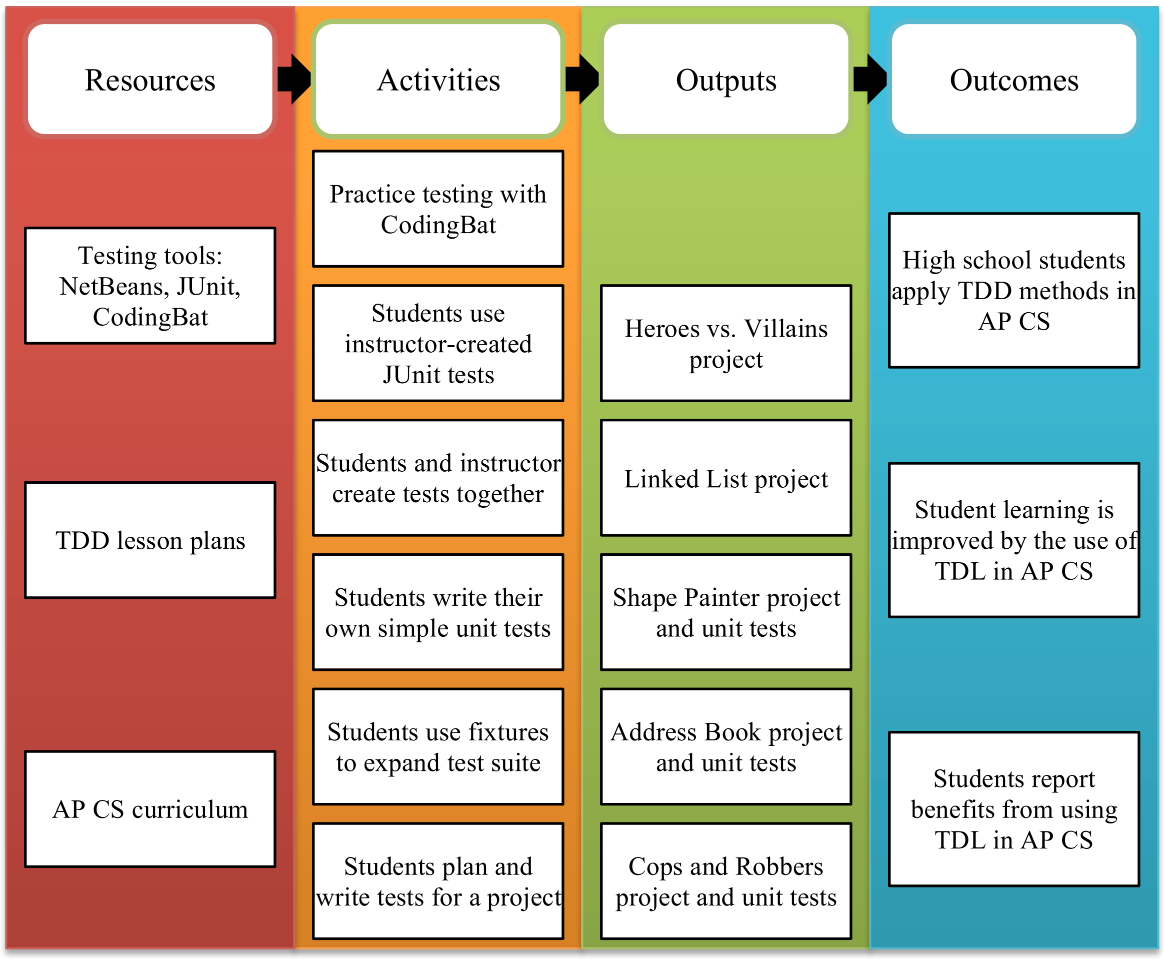

Resources
Activities
Outputs
Outcomes
Testing tools: NetBeans, JUnit, CodingBat
TDD lesson plans
AP CS curriculum
Practice testing with CodingBat
Students use instructor-created JUnit tests
Students and instructor create tests together
Students write their own simple unit tests
Students use fixtures to expand test suite
Students plan and write tests for a project
Heroes vs. Villains project
Linked List project
Shape Painter project and unit tests
Address Book project and unit tests
Cops and Robbers project and unit tests
High school students apply TDD methods in AP CS
Student learning is improved by the use of TDL in AP CS
Students report benefits from using TDL in AP CS

Figure 3: Logic model of TDD introduction in AP CS

Based on students' strong ratings of JUnit and its usefulness in the preliminary study, a more thorough study was planned for the 2013-14 school year. This study included an expansion of the usage of JUnit, combined with a more thorough study of student benefits and experiences. In this study, JUnit and other automated testing tool were used more consistently throughout the year, and students were surveyed three times during the school year regarding their JUnit experiences. A logic model overview of the addition of TDL to AP CS and the anticipated outcomes is presented in Figure 3. The specific times during the academic year where unit testing concepts were taught to AP CS students are indicated in Table 2. (Each chapter in the table has a duration of approximately three weeks.)

Table 2: Insertion of TDL use in AP CS

Chapter/Topic	TDL Topics Taught	How Introduced	Data Collection
1. Introduction to Java	Automated testing using CodingBat, "red/green" idiom	In-class practice with CodingBat	
2. Classes and Objects	Introduction to using JUnit with NetBeans	Instructor-provided JUnit tests, both during in-class practice and as part of project assignment	
4. Linked lists	Choosing test cases, writing test cases with JUnit	Students cooperatively create unit tests for Linked List project before the project is assigned	Student survey #1
5. Inheritance	Writing test cases independently	As part of chapter project Shape Painter, students write unit tests for the classes they create	
7. Interfaces	Using fixtures to shorten test code	As part of chapter project Address Book, students add additional test cases to a provided test suite	Student survey #2
8. GridWorld	Planning both tests and code	As part of chapter project Cops & Robbers, students write unit tests for their own classes	Student survey #3

The three surveys given during the course of the study varied in form, with the intent of collecting both quantitative and qualitative data from students. The first survey, given shortly after students completed the Linked List project assignment, was the same survey that was given to students participating in the preliminary study. This survey focused on quantitative evaluation of students' experiences while working on the Linked List project and asked students to rate their experience with JUnit with this project. Students were asked to rate the usefulness of various items in completing the project on a 1 to 5 scale, and were asked to agree or disagree with statements related to their impressions of JUnit.

The second survey, given to students shortly after completing the Address Book project assignment, was a more open-ended instrument. The survey contained only five questions, but all questions were open-ended questions allowing free response rather than responding numerically. This survey asked students to tell about any changes in their opinions of TDL over time, asking them to give both their initial and current opinions regarding JUnit and test-driven development.

The third survey was given after students completed the Cops & Robbers GridWorld project. This survey combined aspects of both of the first two. Students were again asked to rate the usefulness of items including JUnit testing while working on the project. In addition, students were asked to rate their own programming style and their use and opinion of test-first techniques, using questions adapted from Buffardi [7]. Finally, as in the second survey, students were asked to give free responses about their impressions of TDD and any changes in their opinions of TDD over the course of the school year. Copies of all survey instruments can be found in Appendix C.

In addition to data collection through student surveys, student project grades were collected for individual project assignments where JUnit testing had been added. Student project grades were compared to corresponding grade data from earlier years in AP CS where JUnit had not been used.

Main Study Participants

As in the preliminary study, students enrolled in AP Computer Science at Westside were asked to participate in the study and contribute by completing student surveys. For the 2013-14 school year, eleven students enrolled in AP CS. Eight were in 12th grade, two in 11th grade, and one in 10th grade. One student was female and the remaining ten were male. However, two students withdrew from the AP CS course at the end of the fall semester (one cited excessive course load as a reason for dropping and the other was struggling in the course), leaving only nine students enrolled during the spring semester.

As in the preliminary study, students were not offered compensation in return for participating, and data collection was done using online surveys. Again, Google Docs forms were used for administration of surveys, helping ensure anonymity for students completing the surveys.

Chapter 4 - Results and Discussion

Data was collected from four major sources during the study. As described in the previous chapter, students were surveyed in regard to their experiences at three times during the school year. In addition, project grades from projects where TDD techniques were included in the assignment was collected and compared to data on the same projects from previous years where TDD was not used.

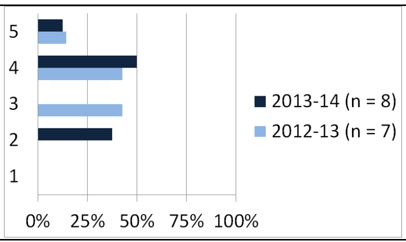
Student Survey #1

Students in AP CS in the 2013-14 school year were first surveyed about JUnit and their opinions on it shortly after the due date of the linked list project. As noted previously, this survey was identical to the survey given during the preliminary study in the 2012-13 school year. This survey asked students to evaluate the usefulness of various scaffolding items provided to students to help them complete the linked list project. Eight students responded to this survey in the 2013-14 school year. For comparative purposes, Figure 4 shows survey data from both the 2012-13 and 2013-14 years.

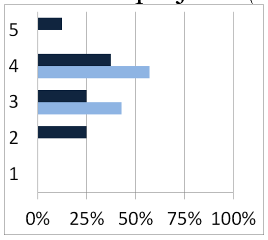
In both 2012-13 and 2013-14, students reported that they found the JUnit components of the linked list project very helpful in completing their requirements. 13 of 15 (combined) respondents responded 4 or 5 (where 1 = not at all helpful and 5 = very helpful) regarding both writing the project's JUnit tests in class and testing their projects with the provided JUnit tests. These items were the two highest-rated items in the survey for helpfulness in completing the project. In addition, 14 of 15 students responded 4 or 5 (agree or strongly agree) to the statement "I think that JUnit is a useful tool", and all 15 students responded 4 or 5 to the statement "I would like to continue to use JUnit in future projects." Students were not as confident in their own JUnit skill at this point, with only

9 of 15 responding 4 or 5 to "I understand JUnit well enough to write my own test cases." However, the overall results show that students found JUnit very helpful on the linked list project.

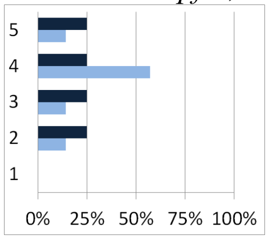
How hard did you find the linked list project, compared to other projects so far in this course? *(1 = very easy, 5 = very hard)*



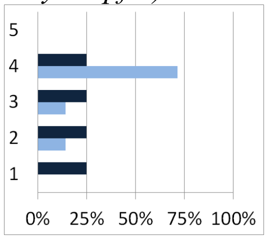
How helpful were each of the following items in understanding the requirements in the linked list project? *(1 = not at all helpful, 5 = very helpful)*



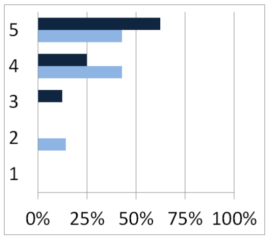
The written project description posted on Blackboard



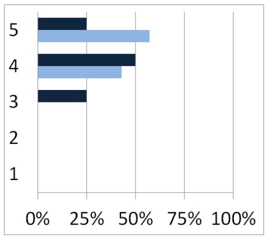
In-class discussion of the project



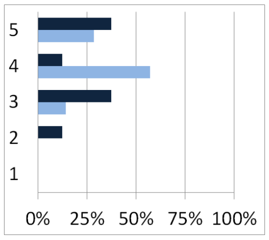
Discussing the requirements with classmates



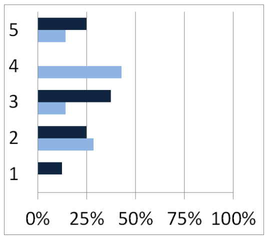
Writing the JUnit tests in class



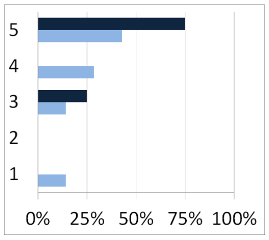
Testing your code with the JUnit tests



Testing your code with the sample main() function

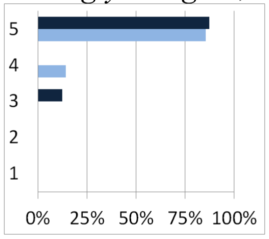


Reading the JavaDoc comments included in IntLinkedList.java

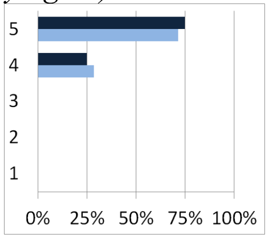


The list diagrams (blue note sheet handed out in class)

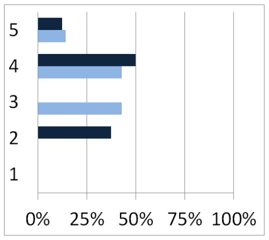
Please indicate to what degree you agree or disagree with each of the following. *(1 = strongly disagree, 5 = strongly agree)*



I think that JUnit is a useful tool.



I would like to continue to use JUnit in future projects.



I understand JUnit well enough to write my own test cases.

Figure 4: Results of Linked List project survey

Student Survey #2

The second student survey was administered early in the spring semester, shortly after the completion of the Address Book project. In this project, students were asked to create several classes that implement the Java Comparator interface, using those classes to sort a simple address book by various fields. Students were also assigned to create unit tests that verified the correctness of their comparators, with the help of a model JUnit test provided by the instructor.

Since the second student survey contained only free-response questions, thematic analysis [5] was applied to the student responses from this survey. The Weft QDA tool [12] was used to assist in this analysis.

Eight students responded to this survey. The survey consisted of five questions. Two questions asked students to describe their initial opinion of TDD and their current opinion. The next two questions asked students what advantages and disadvantages they saw in TDD. The final question simply asked students for any other comments they might have about TDD or JUnit.

Several themes emerged from student responses to this survey. One common theme was initial apprehension: many students reported various initial misgivings when first introduced to TDD. Many of these concerns reflected students' lack of knowledge of the TDD process. The initial skepticism of TDD is consistent with initial attitudes of students toward TDD in other TDL studies, as discussed in chapter 2; students who have only experienced test-last development have difficulty seeing benefits of TDD.

> Student 2.4: I thought it was unnecessary because we can test the program by running it.
>
> Student 2.5: I did not really get the purpose of J-unit because I didn't know the importance of testing programs

> Student 2.6: I thought that it seemed hard, mostly because I had never done things like that before.
>
> Student 2.7: I honestly thought it would be more work than it was worth. This was mostly because it seemed like you had to create another program to test your current program when you could just run the program to test it.

Despite initial apprehension, though, many of the survey respondents expressed current positive TDD opinions. After having worked with JUnit and test-driven development in class and in projects, . Again, this is consistent with previous TDL studies: actually using automated testing tools and TDD patterns helps students to recognize potential benefits to the TDD style of development.

> Student 2.2: It was a new way to think about programming.
>
> Student 2.4: It's very useful because we don't test everything when running a program
>
> Student 2.5: I think it is helpful in many projects a good example is In the linked list project; it helped me determine results, without it I would of struggled a lot more than I did.
>
> Student 2.8: However throughout learning more about JUnit test I realize that they can be helpful to debug especially larger and/or complex programs. Because in larger programs it may take an extremely long time to pinpoint the issue and JUnit tests can do just that.

Current positive TDD opinions encompassed two subthemes which students identified as advantages of TDD. One theme was TDD reduces errors: students stated that TDD helped eliminate program bugs. The other was TDD helps understanding: by using TDD, students were better able to understand project requirements.

> *TDD reduces errors:*
>
> Student 2.2: Doing this will eliminate bugs.
>
> Student 2.3: Reduced the error in the program
>
> *TDD helps understanding:*
>
> Student 2.5: I thought it was easy to make because you expect certain results, majority of the time your results should be predictable. It a helpful debugging tool for making abstract project
>
> Student 2.6: It makes you think about what you need to make your objects do.

> Student 2.7: I think that it is a sort of design in that you know what you are going to create before you create it

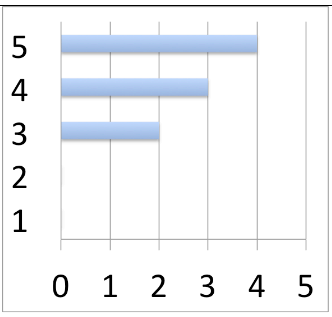
While many students reported that they found JUnit and TDD useful, the opinions were not universally positive. Among the downsides of TDD expressed by students, three themes were identified: TDD takes too long (it required more work compared to test-last), TDD is expensive (students perceived that there were benefits to TDD, but for the projects where TDD was used, the benefits did not outweigh the costs), and tests can have bugs (the value of TDD is diminished or lost if the test suite itself is incorrect or incomplete). Many of the objections to TDD paralleled objections identified by students in earlier studies [8].

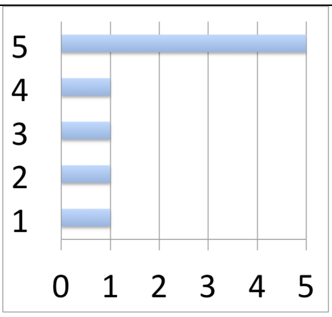
Student Survey #3

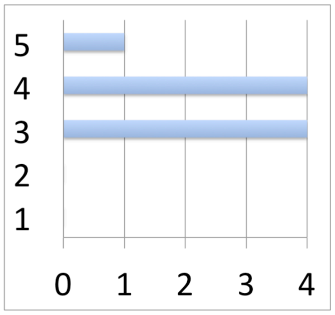
The third and final survey given to students was administered in the spring semester, after completion of the Cops & Robbers project. In this project, students were asked to create three classes that interacted within the environment of the GridWorld case study [9]: a bank class, a bank robber class, and a police officer class. Students were also assigned to create their own test suites for the bank and bank robber classes (the test suite for the police officer class was provided by the instructor). By this point in the course, students were expected to analyze the requirements for the classes in the project, determining and creating appropriate test cases relative to the project requirements.

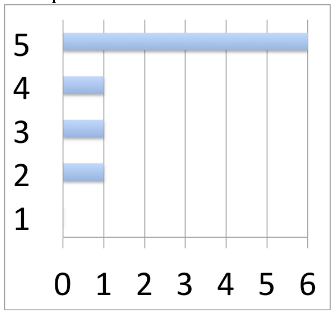
This survey contained a combination of Likert items similar to those used on the first survey and free-response questions similar to those of the second survey. The first set of survey items asked students to evaluate how helpful various items were in completing the Cops & Robbers project. The results of these items are shown in Figure 5.

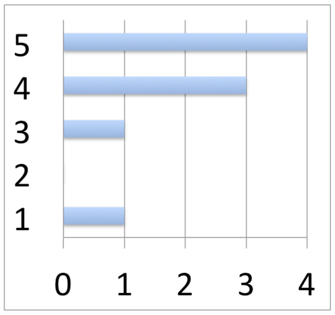
How helpful were each of the following in completing the GridWorld Cops & Robbers project? (1 = not at all helpful, 5 = very helpful)


The written project description posted on Blackboard

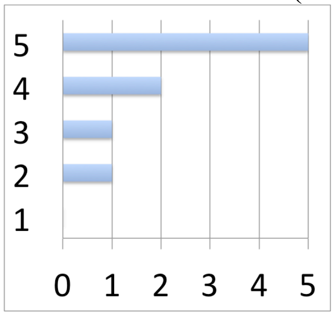

Discussing the project with others

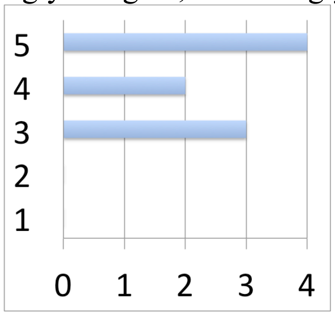

Writing JUnit tests for Bank and Robber


Testing your project by running it


Testing your project with JUnit tests

Please indicate to what extent you agree or disagree with the following. (1 = strongly disagree, 5 = strongly agree)


I understood JUnit well enough to create test cases for Bank and Robber.


Creating JUnit tests helped me think about what Bank and Robber should do.

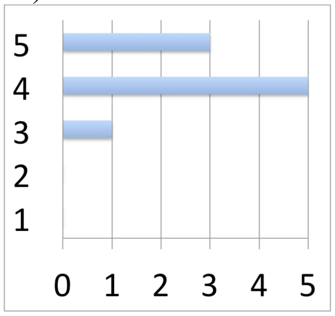

I feel that my JUnit tests adequately tested my program code.

Figure 5: Results of GridWorld project survey: Project-specific survey items

Students again reported that using JUnit to test the project was helpful, with 7 of 9 students responding 4 or 5 to that question. Students were a bit more ambivalent about the helpfulness of actually creating the JUnit tests. 5 of 9 students rated this aspect of the project with a 4 or 5. One interesting result of this portion of the survey is that students responded very strongly that testing their project by running it was helpful (7 of 9 students responding 5 / "very helpful"). This result probably alludes to the graphical nature of the GridWorld case study. Unlike the Linked List project discussed earlier, students received immediate visual feedback on their code and its behavior in the GridWorld user interface.

Students were also asked to evaluate their understanding of JUnit. Most of the students (7 of 9) reported that they understood JUnit well enough to create test cases as required in the project. Students also felt that their JUnit tests were adequate for testing their production code, with 8 of 9 students responding with a 4 or 5 to this question. A majority of students (6 of 9) reported that the process of creating tests helped them think about the requirements of the classes which were tested by those tests.

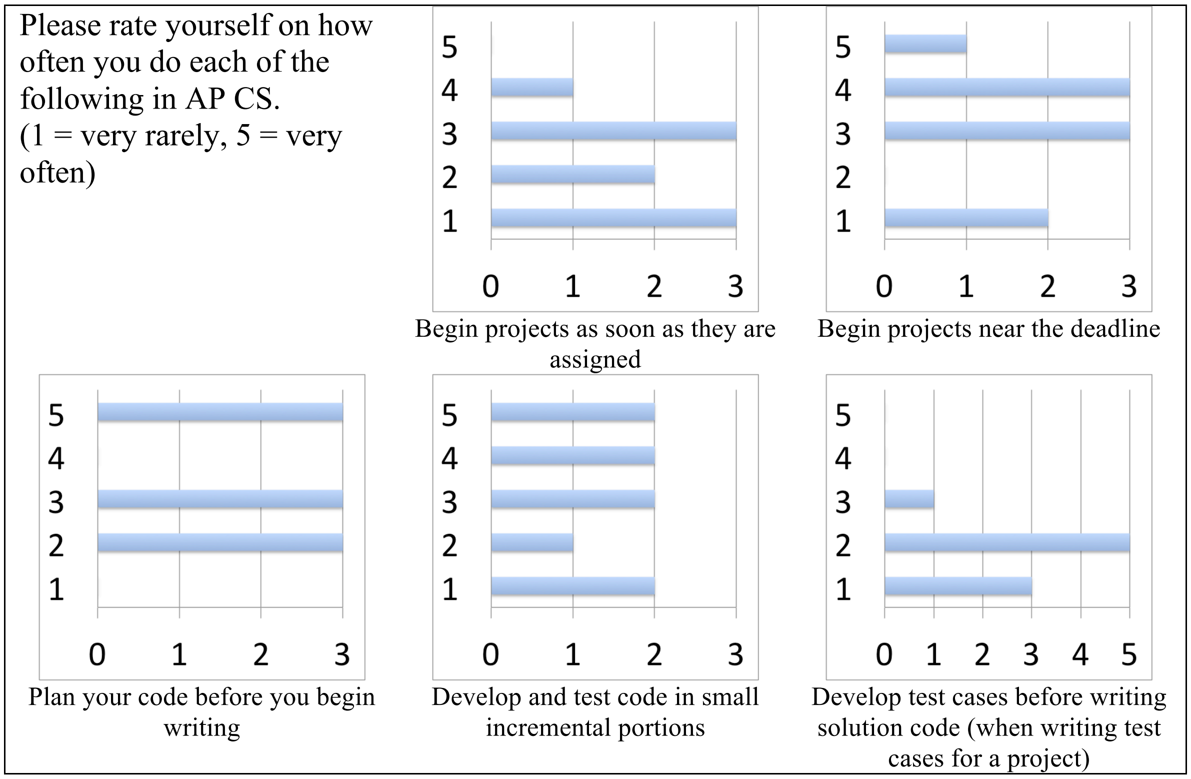

Please rate yourself on how often you do each of the following in AP CS.
(1 = very rarely, 5 = very often)

Figure 6: Results of GridWorld project survey: Student testing behaviors

The next portion of the survey included items asking students to evaluate their own programming practices, and then to evaluate their overall opinions of TDD. These items were adapted from a survey administered to CS2 students by Buffardi and Edwards [7]. Results from this survey are shown in Figure 6 and Figure 7.

Students showed mixed adherence to TDD behaviors in this portion of the survey. In particular, no students reported that they often or very often develop test cases before writing solution code on projects. This response seems to indicate that students either do not fully understand the ideas behind test-first development or else are rejecting test-first development as not being worthwhile, despite earlier responses showing that they favor TDD as a whole. It is also possible that students simply placed excessive time pressure on themselves and then abandoned test development as an afterthought, as 4 of 9 students responded that they often or very often began projects near the deadline.

Although students did not frequently use test-first development, several of them did report that they tended to develop and test in small portions (4 of 9 students responding "often" or "very often", with 3 responding "rarely" or "very rarely"). Thus, while students reported that they found JUnit useful in the earlier section of the survey, that was not enough to lead them to test-first development. These results are consistent with earlier studies; even when students recognize benefits of test-first development, many students are reluctant to adopt this particular aspect of TDD.

Despite not embracing test-first development, students still expressed positive opinions of TDD as a whole on the final scale items of the survey. Out of 9 students, 7 agreed or strongly agreed that TDD was easy to understand, and 8 of 9 each felt that TDD helped them write better test code and better program code. 7 students expressed that TDD helped them to understand project requirements. Students were emphatic that TDD helped them to better design their programs, with all 9 students responding "strongly agree" to this item. Students were asked both if they felt that TDD saved them time and if TDD took time away from other tasks; 6 students agreed that TDD saved them time while only 2 reported that TDD took away time. Finally, 8 of 9 students agreed that they would at least consider using TDD in programming beyond the AP CS course.

Please rate to what extent you agree or disagree with each of the following statements. (1 = strongly disagree, 5 = strongly agree)

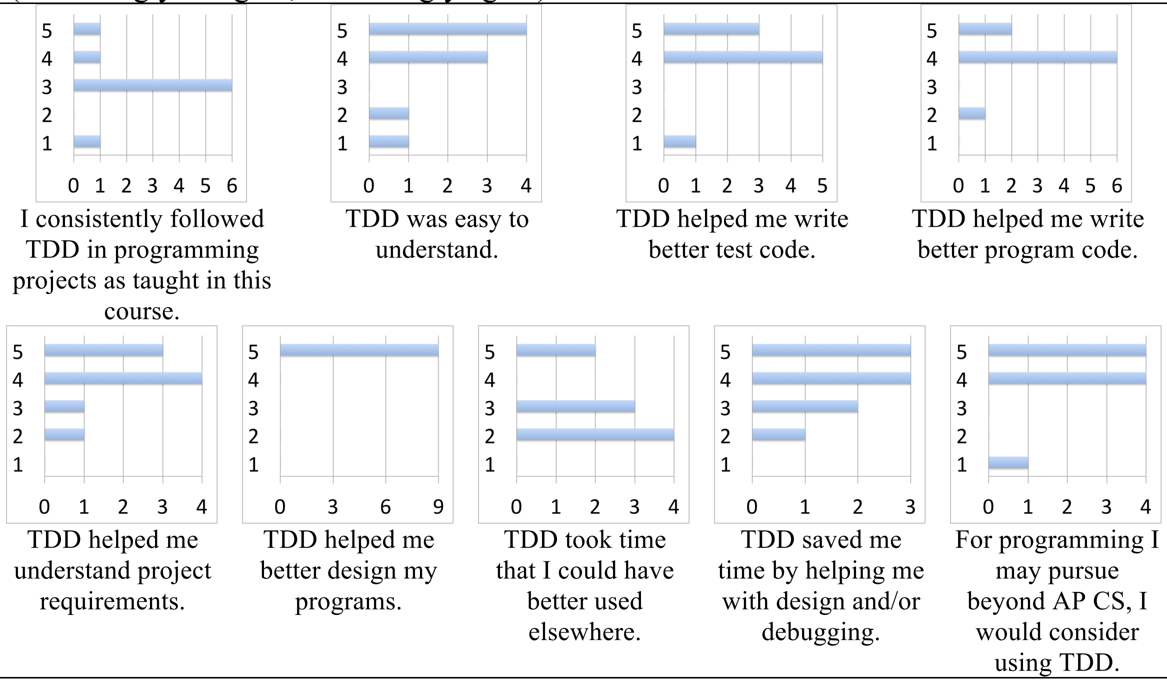


Figure 7: Results of GridWorld project survey: Students' overall evaluation of TDD

These items show that students continued to have overall positive impressions of the TDD process. Students' opinions reflected that using TDD helped them to think about the requirements of their projects and to plan their production code. They also expressed a continued interest in using TDD later in their CS careers.

It must be acknowledged that some part of students' positive opinions may have been biased by earlier teaching in the course where the instructor identified the expected benefits of TDD. However, these benefits were identified early in the course when first introducing TDD, rather than immediately before the survey, so any bias effects are not expected to be overly large.

The final portion of the third student survey contained five open-ended questions:

- When we first started learning about TDD, what was your opinion of it?
- What is your current opinion of TDD?
- What advantages and/or disadvantages do you see in TDD, as compared to writing code and then testing it?
- What differences do you think might apply to using TDD in a business project rather than in a school class?
- Do you have any other comments regarding JUnit or TDD?

As with the second survey, thematic analysis was used to identify common themes in the responses to the in this survey. Many of the themes that emerged from student responses were similar or identical to those seen in the earlier survey. Students continued to identify two major advantages to TDD: TDD reduced errors in their programs and TDD helped understanding (of project requirements). In the third survey, student responses focused mainly on the TDD helps understanding theme rather than TDD reduces errors.

> <u>Student 3.1</u>: TDD gives an outline as to what things should do.
>
> <u>Student 3.2</u>: knowing that I can use it to see the requirements
>
> <u>Student 3.3</u>: makes design a bit better
>
> <u>Student 3.4</u>: I think it can be very helpful to know what you want your program to do
>
> <u>Student 3.7</u>: it allows me to easily understand what the program should do and lets me understand the program faster
>
> <u>Student 3.8</u>: I thought that it could be rather helpful in designing the main project.

The main disadvantages that students identified to TDD in this survey continued to be TDD takes too long and TDD is expensive (when balanced against the benefits for the projects students completed in class). Even though students were overall neutral to the statement "TDD took time that I could have better used elsewhere" in the closed-form portion of the survey (2 students strongly agree, 3 neutral, 4 disagree), when prompted to respond in more detail, students still continued to focus on the TDD is expensive theme in their responses.

> *TDD takes too long:*
>
> Student 3.6: It takes more time developing a program
>
> Student 3.8: When we first began creating Junit tests I saw it as a waste of time because I did not use it to help me code and it did take much longer to code than if I were to use TDD.
>
> *TDD is expensive:*
>
> Student 3.4: However, I don't think that for some of the projects we've done it was necessary as I could very easily test them in the main class/gui.
>
> Student 3.7: I thought that TDD was a bit over the top, probably because the program was simple when we first wrote them.

When students gave responses related to the TDD is expensive theme that was identified on both survey 2 and survey 3, the response sometimes reflected that TDD might offer more benefit in a larger long-term project. (This observation is similar to that made by Blackwell's attention investment model [4].) To further explore this theme and elicit responses from students, the question "What differences do you think might apply to using TDD in a business project rather than in a school class?" was added to the third survey. In responding to this question as well as to other questions on this survey, students frequently identified that they felt that TDD would be more useful on a larger project such as would be built in the CS industry, as opposed to a smaller project for class.

> Student 3.3: It is useful in big projects, or on projects where you jump into already written code, like grid world
>
> Student 3.4: although the programs that we've made in class I haven't felt are big enough to really take full advantage of it.
>
> However, I don't think that for some of the projects we've done it was necessary as I could very easily test them in the main class/gui.

> Student 3.7: It would show more usefulness in a business project because the code we write in class is examples of how it is used. A business would most likely have a large problem that would be hard to write without TDD in to my opinion.
>
> Junit would be good with a larger group project.
>
> Student 3.8: I think TDD would be different in industry because in academics TDD is used as a learning tool while in industry TDD is used to produce real world results. In industry one would use TDD as tool to further the industry and this would be, in my opinion. more effective in the CS industry because TDD could develop both the CS side and industry side.

It is noteworthy that as students worked more with TDD and gained a greater understanding of the TDD process, they more readily identified TDD as being more likely to be useful on larger projects. This suggests that as students learned more about TDD, they became better able to foresee the advantages of TDD in projects, as well as the disadvantages.

Some students were skeptical about businesses' willingness to use the TDD process due to cost concerns:

> Student 3.6: TDD would be useful in CS industry, but I feel a lot of companies might skip that step, just so they can throw a product out and then send out repair or update packages because that will earn more money for them.
>
> Student 3.9: I don't think business's use TDD because they want to make money quickly and then send updates to patch the problems in it.

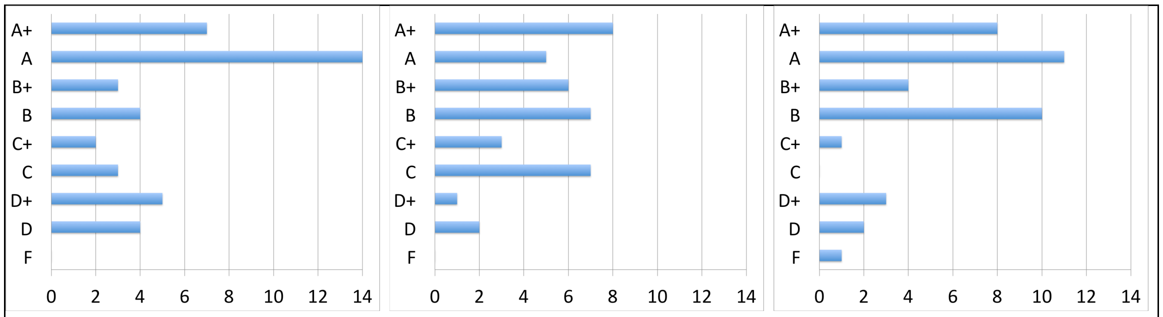
It is well-known that TDD is far from a universal practice in the CS industry [1] [13], and the students in the study were aware of this as well (through both direct instruction and classroom discussion of TDD as a practice). It is interesting that some students ascribed non-adoption of TDD in industry to a deliberate business decision to create inferior software and then charge for upgrades, rather than developers' unfamiliarity with or dislike for TDD techniques. These students appear to be making the conclusion that if businesses did adopt TDD techniques, the result would be that software produced would contain fewer defects.

TDD Project Grade Data

Student grade data was also collected from selected projects on which unit testing was added as a component of the project, and the grades analyzed to determine whether the data supported hypothesis #2 (student learning will be positively impacted by TDL). Grades were collected for the following five projects: Heroes vs. Villains, Linked List, Shape Painter, Address Book, and GridWorld Chase. These five projects were chosen for grade analysis as projects that met two criteria:

- All of these projects had a testing component of some sort added to the project requirements in the 2013-14 school year.
- All of these projects were assigned, in substantially identical form except for the addition of the testing component, in prior school years, and those prior years' grade data were available.

For comparison purposes, student grades in the 2013-14 school year were compared to those in the 2010-11 and 2011-12 school years. The 2012-13 school year was omitted since this was the year in which the TDL preliminary study was performed. Since TDL was partially but not completely integrated into that year's AP CS curriculum, data from that year could not be classified as either with-TDL or without-TDL data.



2010-11 (no TDL): 10 students, 42 project grades

2011-12 (no TDL): 9 students, 39 project grades

2012-13 (TDL): 11 / 9 students (fall/spring), 40 project grades

Figure 8: Student project grade counts for TDL projects

Histograms summarizing the distribution of student grades for these five projects in each year are shown in Figure 8. The full grade data can be found in appendix D. In many cases, not all students enrolled in the course completed all the projects assigned. Grades are included only for students who completed the selected projects. The total enrollment in AP CS in each school year is also shown.

It is assumed that these projects differ in difficulty. Since the number of project submissions varied from one project to another and from one year to another, there is a concern that the different distributions of projects from one year to another might influence the overall grade distribution. To test for possible bias based on the different distribution of project submissions, Fisher's exact test was applied to the project count data (Table 3) to help determine whether a significant difference existed in the distribution of project submissions for the TDL class vs. the non-TDL classes. This test produced a *p*-value of 0.903, which supports the null hypothesis that the distribution of project submissions by project is comparable. Therefore, it is assumed that the two sets of grade data represented an overall approximately equal distribution of project difficulty. Thus the grade data was examined to determine whether students performed significantly differently on projects when using TDL techniques as compared to students who did not use TDL.

Table 3: Total number of student grades available for TDL projects

	Heroes vs. Villains	Linked List	Shape Painter	Address Book	GridWorld Chase	Total
Non-TDL	17 (21%)	16 (19.8%)	16 (19.8%)	16 (19.8%)	16 (19.8%)	81
TDL	11 (27.5%)	7 (17.5%)	9 (22.5%)	7 (17.5%)	6 (15%)	40

Student grades on projects in AP CS are reported to students as letter grades, but the letter grade is determined by a grading rubric which is specific to each project. The rubric has a number of points available (this number of points varies by project, but is generally either 10 or 20). These points are not directly awarded as a grade, but are instead converted to a letter grade based on a predefined scale. For example, for a typical project with 20 points available on the rubric, 19 or 20 points results in a grade of A, 18 points in B+, 16 or 17 points in B, 15 points in C+, and 13 or 14 points in C. When students earn less than 13 points on the rubric, they are expected to address their submission's shortcomings and resubmit the project. From the basic grade determined by the rubric, two adjustments are made to the grade awarded:

- Students may complete optional portions of the project which are specified on a per-project basis. Completing a specified number of optional extra challenges raises the student's grade by half a letter grade: A becomes A+, B+ becomes A, and so on. This is the only way in which a student can earn an A+ on a project.
- Conversely, students' grades are lowered if the project is submitted after the specified deadline. The penalty for late submissions is half a letter grade for a project submitted up to two (school) days late, or beyond that, one letter grade per week late, with a maximum deduction of two letter grades.

It would have been preferable to compare the original point totals earned by students on the project grade rubrics, since the rubric point totals offer finer-grained performance measurement than the final letter grades. Unfortunately, these point totals are not available, as grading rubrics are returned to students and a copy is not kept by the instructor. Therefore, the letter grades earned by students on the various projects were compared. To facilitate ranking of student letter grades for analysis, the grades were converted to their numeric equivalents, using the scale used in AP CS: A+ = 100, A = 96, B+ = 89, B = 86, C+ = 79, C = 76, D+ = 69, D = 66, and F = 59. The distribution of grades from the TDD and non-TDD data, based on this scale, are shown in Figure 9.

Since project grades do not fall into a normal distribution, the non-parametric Mann-Whitney U test was used to compare the data in the two test sets. However, this test failed to find a statistically significant difference between the TDD and non-TDD grade collections (two-sided test W = 1478, $n_1 = 40$, $n_2 = 81$, $p = 0.43$). Therefore, the conclusion cannot be made that the TDD students in our sample group performed significantly better than the non-TDD students.

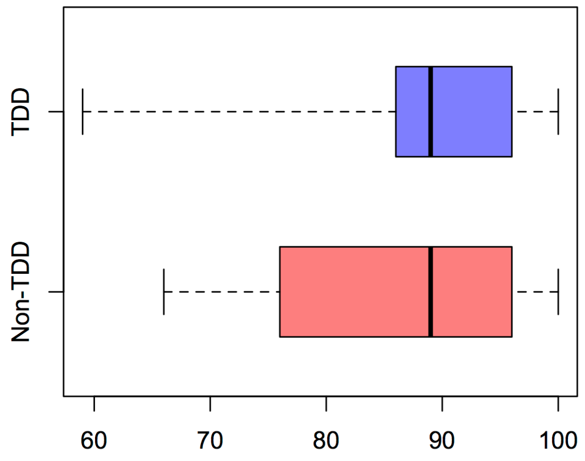


Figure 9: Project grade distribution in TDD and non-TDD classes

Although the Mann-Whitney analysis did not show a statistically significant difference between the scores of students using TDD and not using TDD, the box plot shown in Figure 9 does show that the 25th percentile grade was higher among the TDD students than non-TDD students (although the lowest grade was lower), suggesting a possibility of a lower variance in grades in the lower half of the class. Intuitively, this result seems plausible. AP CS is an elective course and most of the students enrolled in it are self-motivated and skilled in CS. Therefore, many students in the course normally earn high grades even without the help of TDD. For students already earning an A or B on most projects (which includes the majority of students in AP CS), there is limited room for TDD to help improve students' project grades. Therefore, it is reasonable that TDD might have a greater effect for students who are struggling more in AP CS.

To further investigate this possibility, the Mann-Whitney analysis was repeated, this time including only project grades of students whose overall grade in AP CS was in the lower half of their class. Figure 10 shows the box plot of these students' project grades.

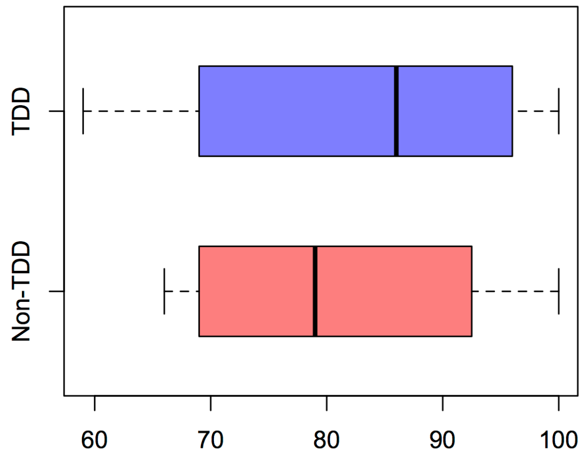


Figure 10: Project grade distribution in TDD and non-TDD classes (lower-scoring students only)

When only students in the lower half of their classes were considered, students using TDD had higher mean and median project scores than students not using TDD (median for TDD students = 86, for non-TDD = 79; mean for TDD students = 83.47, for non-TDD = 81.74). Although central measures of grades of TDD students were seen to be higher than those of non-TDD students, the Mann-Whitney test again did not show a significant difference (two-sided test W = 213, $n_1 = 15$, $n_2 = 31$, $p = 0.65$).

Analysis of Data

Next, the overall applicability of the data collected to each hypothesis of the thesis was considered. Student survey responses show support for two of the three hypotheses.

Analysis of Hypothesis 1: High school students can successfully apply test-driven methodologies in a high school computer science course

Student project submissions as well as data from the three student surveys support hypothesis 1. Students in the AP CS course were able to both use instructor-provided test cases and create their own test cases using JUnit. Further, students reported that using TDD helped them understand the requirements of projects and think about how to design and develop their code.

In the first student survey, 7 of 8 respondents found writing JUnit tests in class helpful in completing the project, and 6 found testing their code with JUnit tests helpful. 7 respondents stated that JUnit was a useful tool, and all 8 stated that they wanted to continue to use JUnit. In the second survey, students reported initial misgivings when they first started learning about TDD, but their opinions of TDD at the time of the survey were positive. Students also reported that JUnit and TDD helped them find and eliminate errors in their programs. In the third survey, 5 of 9 respondents found writing JUnit tests helpful on the Cops & Robbers project, and 7 found testing their code with JUnit helpful. 7 students reported that TDD was easy to understand, 8 reported that TDD helped them write better program code, and 6 reported that TDD saved them time designing and/or debugging their programs.

These responses show that students were applying TDD on their projects as specified in assignments and that TDD was helping their understanding (or at the very least, their perception of understanding) of project requirements. High school students were indeed able to successfully apply TDD in this study.

Analysis of Hypothesis 2: Student learning will be positively impacted by the use of test-driven learning in the high school CS curriculum

Collection of student grade data did not show a statistically significant increase in student grades when using TDL. On the other hand, mean and median scores for students in the lower half of the class using TDL were higher than the same measures from comparable classes not using TDL. Although this improvement was not found to be statistically significant in this study, it is possible that a larger study with more students might find a significant difference.

Given that other studies of TDL have shown that college students who adhere to TDL practices tend to produce better code, it is recommended that further research in this area continue to study whether high school student programmers also show improved learning when using TDL.

Analysis of Hypothesis 3: Students will report that they feel there are benefits from using test-driven development

Students were consistent on all three surveys in reporting that they felt that there were benefits from using TDD. Students recognized some disadvantages of TDD as well, but overall, student response data reflects that the students surveyed did recognize benefits of TDD.

Student responses to free-response questions on surveys 2 and 3 indicated two specific benefits that students identified from TDD: TDD helped them understand project requirements, and to a smaller extent, it helped them reduce the number of bugs in their project code. Students on the third survey noted in particular that they felt that TDD would increase in utility as the size of their projects increased.

These responses were consistent with objective question responses on surveys 1 and 3. In the first survey, students rated writing and testing their project with JUnit among the most helpful among tools used to help them complete the linked list project. In the third survey, 6 of 9 respondents stated that creating their own JUnit tests helped them think about the requirements of their Bank and Robber classes in the Cops & Robbers project. 7 respondents stated that TDD helped them understand project requirements (in general), all 9 felt that TDD helped them better design their programs, and 8 reported that they would consider continuing to use TDD beyond the AP CS course.

These responses show that students did feel that TDD was beneficial to them. Students found TDD helpful in understanding project requirements and debugging programs. Further, they identified that TDD's benefits would be larger for larger projects and remained open to using TDD in the future.

Threats to Validity

Although student response to TDD in this study was positive in many ways, we do recognize that there are threats to both the internal and external validity of the study results. Some of the threats to validity that were identified include:

- The sample size is small (threat to internal validity). Due to the small enrollment in AP CS both during the time of the study and during the earlier classes used for comparative data, a limited amount of data on the use of TDL was collected. A larger sample size could show negative effects of TDL that did not emerge in the limited data collected in the study. Further, larger classes might make the teaching of TDL more difficult, reducing its effectiveness as a learning tool. Conversely, increasing the number of students studied or measuring student performance using metrics other than project grades might (or might not) show a statistically significant benefit to student learning with TDL, although this study did not find such an effect in the data collected.

- Selection bias and experiment mortality (threat to internal validity). Students self-selected for participation in this study (by enrolling in AP CS), and results for AP CS students might not be generalizable to other courses. It is assumed likely that students enrolled in this particular AP CS course are representative of AP CS students in general, but representativeness was not addressed in this study. The study also unfortunately lost two students during the duration of the school year, as those two students withdrew from AP CS after the fall semester. It is likely that the withdrawing students were the least enthusiastic about computer science, and it is possible that the withdrawal of these students had an impact on the results of the later data gathered.
- Testing effects (threat to internal validity). Students may have been biased when completing surveys. As part of obtaining informed consent from students when completing surveys, students were informed that the surveys they were completing would be used as data in this thesis, and were aware that the topic of the study was TDL. It is possible that students were influenced to overstate their positive feelings regarding TDL and understate their negative feelings, in the belief that doing so would help the instructor in this study.
- Generalizability concerns (threat to external validity). The instructor/researcher has background knowledge in computer science that may not be available to most high school CS instructors, and Westside High School provides advantages to

teaching AP CS that are not available in all high schools. The instructor in this AP CS course was able to apply his own experience with teaching and using computer science to help teach TDL. Many high school CS instructors do not possess the same level of training in computer science, and therefore may not be as comfortable using and teaching TDL. Also, the one-to-one laptop program at Westside ensures that all AP CS students have access to computers both at home and in the classroom. It also ensures that all students have standard installations of course software such as the IDE, Java compiler, and testing tools. In a school where technology is not as universally available to students, TDL may be harder to implement since students may not all have access to the hardware and software needed. This threat to validity would be best addressed by further studies attempting to replicate the study results at other high schools with other CS instructors.

While there are a number of possible threats to validity for this study, the overall results of the study are positive. Further study is needed to confirm that TDL can benefit other high school computer science classrooms, but there

Chapter 5 - Conclusions

In this study, high school students were introduced to test-driven learning in an AP Computer Science course. TDL was introduced to students on a gradual basis, first introducing students to the basic concept of automated testing, then having students write code and use automated tests to test that code, finally progressing toward requiring students to write unit tests for their code as they wrote the code itself. Based on survey data collected from students at various points during the AP CS course and on project grade data, we were able to evaluate the use of TDL in this high school CS course.

Students in this course showed that they were able to use TDL and to apply concepts of TDL in their project assignments. When surveyed in the fall, 13 of 15 students over two school years reported that testing their code with JUnit on the linked list project was helpful or very helpful in completing that project. In the final survey after using TDD on multiple projects, 7 of 9 students reported that TDD was easy to understand, 8 of 9 reported that TDD helped them write better program code, and all 9 reported that TDD helped them to understand project requirements. Based on these strong responses, it is clear that students were using JUnit to help test their code and that using JUnit was helping student to understand their project requirements. Therefore, we conclude that students in this study were able to use and apply TDL in their work.

It could not be concluded, however, that TDL in this course had a significant influence on student performance. Although a larger study or a study that addressed other measures of student performance might be able to show a statistically significant improvement in grades from the introduction of TDL in high school classrooms, this study was not able to do so. On the other hand, it also was not shown that TDL has a detrimental effect on student performance. Rather, mean and median grades of students using TDL were the same or slightly higher than those of previous students who did not use TDL. This difference was not statistically significant enough to allow conclusions regarding TDL's effect on student grades to be made, but it did suggest that further research in the use of TDL at the high school level and its effect on student performance is merited.

Finally, students responding to surveys showed strongly that they felt that there were benefits to using TDD. This support was consistent across both numerical scale items and free response questions, and across all three surveys that were administered. These results were not universal, as some students did report a dislike of TDD and some students did not report the benefits that other students found. However, the results consistently showed that the number of students reporting that TDD was beneficial to them outweighed the number reporting that it was not, generally by large margins.

Overall, use of TDL in the course studied was regarded positively by both the students and instructor. Students successfully applied TDL principles in their projects and felt that doing so helped them understand their code and its requirements. The instructor saw students use TDL to think critically about their projects and did not dedicate excessive classroom or planning time to TDL usage. TDL is a viable teaching technique in high school computer science that can complement other CS instructional techniques.

Implications for Future Research and for Other Teachers

In this study, we have seen that students at the high school level are capable of making use of JUnit and applying the concepts of test-driven learning. Further, TDL did not have a detrimental effect on students' grades. While students using TDL did not show a statistically significant improvement in project grades, the students reported other benefits of TDL that justify its continued use and study.

The high school students studied were overall quite accepting to test-driven development techniques, especially after the introduction of TDD in a gradual manner that made benefits of TDD visible to students. In other studies, students were found to be more likely to recognize TDD as beneficial (and less likely to object to the use of TDD) after having actually tried the TDD process [15]. Also, studies found that the difficulty of using JUnit was a potential obstacle to TDD adoption [6] [20]. Therefore, a gradual introduction to TDD is recommended, allowing students to explore the concepts of unit testing and creating test cases, and to build familiarity with unit testing tools before creating their own test cases.

A fourth hypothesis was considered for testing in this thesis: "TDD can be integrated into high school curricula without excessive instruction time or instructor effort." Ultimately, this hypothesis was excluded as too difficult to test within the scope of the study planned. Experience in this study has shown that in the course studied, integrating TDD into the curriculum did not require excessive time for preparation or instruction. Of course, other instructors' experiences may vary, but TDD is a topic that can be introduced into a CS curriculum on a small or large scale. Any high school CS instructor who understands TDD well enough to introduce it in class can experiment with adding TDD to their lessons.

An instructor who is not comfortable with tools such as JUnit can still introduce concepts of testing with tools such as CodingBat, which can be used for as little as a single lesson on a particular topic. At the other extreme, some instructors might wish to embrace TDD and automated testing throughout their course. Tools such as Web-CAT can help teach TDD, while also providing feedback to both students (as they progress on their program) and instructors (to observe how fully students are implementing TDD). Web-CAT's automated grading tools may, after the initial setup period, actually help reduce instructors' time requirements.

Also, although this study has focused on AP CS and Java, there are comparable tools available in a number of programming languages. The CodingBat website contains Python problems as well as Java problems, so courses using Python can use CodingBat to help teach beginning Python and to introduce concepts of testing. As for actual testing tools, JUnit has counterparts in a number of other programming languages, most of them named using a {foo}Unit convention. For example, the Python analogue to JUnit is called PyUnit [28], and C++ can be unit tested with CPPUnit [29] or CxxTest [32]. The usage of these tools varies somewhat from one language to another, but the basic concepts of unit testing are generally the same regardless of language. On the one hand, introducing very new programming students to automated testing may not be effective since those students are still learning the concepts of programming itself. On the other hand, once students have a reasonable base of programming skill, adding unit testing tools to the curriculum becomes a possibility. As with most new concepts, instructors may prefer to start slowly, perhaps introducing testing by first distributing instructor-provided unit tests with programming assignments.

For future studies, there are some changes that could be made to the research design that might aid the research, and some additional avenues available for study. One weakness of the current study is that since students only submitted completed projects, it was difficult to determine how well students were following TDD practices. Students can self-report on these practices in surveys (and did in this study), but self-reporting has limitations. A future study might instrument students' IDEs or use recording software on school-issued computers to further study students' actual development habits with regard to TDD. Another possible route for future studies to follow is to evaluate student performance using multiple metrics rather than only project grades, which in this study were not sufficient to confirm or refute student performance gains from TDL. Some examples might include using unit tests themselves to evaluate projects for correctness, using code coverage or similar measures to evaluate student-written unit tests, and evaluating student understanding of testing by asking students to suggest, create, or critique test cases. One other possible (but difficult) avenue of study would be a longitudinal study, following high school or college students after completing the course where TDD is introduced. This study could evaluate how many students who have studied TDD continue to use the practices they have learned in other courses or in the CS industry.

References

[1] Ambler, S. 2008. Test-Drive Development Survey Results. Retrieved April 1, 2014, from http://www.ambysoft.com/surveys/tdd2008.html.

[2] Beck, K. 2002. *Test Driven Development: By Example* (The Addison-Wesley Signature Series), : Addison-Wesley Professional.

[3] Bhat, T and Nagappan, N. 2006. Evaluating the Efficacy of Test-driven Development: Industrial Case Studies. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (ISESE '06). ACM, New York, NY, USA, 356-363.

[4] Blackwell, A, 2002. First Steps in Programming: A Rationale for Attention Investment Models. *In Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments* (HCC'02). IEEE Computer Society, Washington, DC, USA, 2-10.

[5] Braun, V and Clarke, V. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3 (2), 77-101.

[6] Briggs, T and Girard, C.D. 2007. Tools and Techniques for Test-Driven Learning in CS1. *J. Comput. Small Coll.* 22, 3 (January 2007), 37-43.

[7] Buffardi, K and Edwards, S. 2012. Exploring Influences on Student Adherence to Test-Driven Development. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* (ITiCSE '12). ACM, New York, NY, USA, 105-110.

[8] Buffardi, K and Edwards, S. 2014. A Formative Study of Influences on Student Testing Behaviors. Presented at the 45th ACM technical symposium on Computer science education (SIGCSE '14), pending publication.

[9] The College Board, 2014. AP Central - AP Computer Science A Home Page. Retrieved April 9, 2014, from http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/4483.html.

[10] Cuny, J. 2012. Transforming High School Computing: A Call to Action. *ACM Inroads* 3, 2 (June 2012), 32-36.

[11] The Eclipse Foundation, 2014. Eclipse - The Eclipse Foundation open source community website. Retrieved April 9, 2014, from http://www.eclipse.org.

[12] Fenton, A. 2013. Weft QDA - a free, open-source tool for qualitative data analysis. Retrieved April 9, 2014, from http://www.pressure.to/qda/.

[13] Hammond, S and Umphress, D. 2012. Test Driven Development: The State of the Practice. In *Proceedings of the 50th Annual Southeast Regional Conference* (ACM-SE '12). ACM, New York, NY, USA, 158-163.

[14] Janzen, D and Saiedian, H. 2006. Test-Driven Learning: Intrinsic Integration of Testing into the CS/SE Curriculum. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education* (SIGCSE '06). ACM, New York, NY, USA, 254-258.

[15] Janzen, D and Saiedian, H. 2007. A Leveled Examination of Test-Driven Development Acceptance. In *Proceedings of the 29th international conference on Software Engineering* (ICSE '07). IEEE Computer Society, Washington, DC, USA, 719-722.

[16] Janzen, D and Saiedian, H. 2008. Test-driven learning in early programming courses. *SIGCSE Bull.* 40, 1 (March 2008), 532-536.

[17] Jarzabek, S. Teaching Advanced Software Design in Team-Based Project Course. In *26th International Conference on Software Engineering Education and Training (CSEE&T),* San Francisco, 31-40.

[18] JUnit: A programmer-oriented testing framework for Java. Retrieved April 9, 2014, from http://junit.org.

[19] Kaczmarczyk, L, Dopplick, R, and Education Policy Committee. Rebooting the Pathway to Success: Preparing Students for Computing Workforce Needs in the United States. Technical report, Association for Computing Machinery (ACM), 2014.

[20] Lappalainen, V, Itkonen, J, Isomöttönen, V, and Kollanus, S. 2010. ComTest: A Tool to Impart TDD and Unit Testing to Introductory Level Programming. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (ITiCSE '10). ACM, New York, NY, USA, 63-67.

[21] Larman, C and Basili, V. 2003. Incremental and Interactive Development: A Brief History. *Computer* (June 2003), 47-56.

[22] Madeyski, L. 2010. *Test-Driven Development: An Empirical Evaluation of Agile Practice*. Springer-Verlag, Heidelberg.

[23] Melnik, G and Maurer, F. 2005. The Practice of Specifying Requirements using Executable Acceptance Tests in Computer Science Courses. In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (OOPSLA '05). ACM, New York, NY, USA, 365-370.

[24] Nebraska Department of Education, 2013. 2012-2013 State of the Schools Report: A Report on Nebraska Public Schools. Retrieved April 9, 2014, from http://reportcard.education.ne.gov/Default.aspx?AgencyID=28-0066-001.

[25] Oracle Corporation. 2013. NetBeans IDE: The Smarter and Faster Way to Code. Retrieved April 9, 2014, from https://netbeans.org/.

[26] Proulx, V and Jossey, W. 2009. Unit Test Support for Java via Reflection and Annotations. In *Proceedings of the 7th International Conference on Principles and Practice of Programming in Java* (PPPJ '09). ACM, New York, NY, USA, 49-56.

[27] Proulx, V. 2009. Test-Driven Design for Introductory OO Programming. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09). ACM, New York, NY, USA, 138-142.

[28] Purcell, S. 2013. PyUnit - the standard unit testing framework for Python. Retrieved April 9, 2014, from http://pyunit.sourceforge.net/.

[29] SourceForge.net. 2009. SourceForge.net: CppUnit. Retrieved April 9, 2014, from http://sourceforge.net/apps/mediawiki/cppunit/index.php.

[30] Stejskal, R and Siy, H. 2013. Test-Driven Learning in High School Computer Science. In *26th International Conference on Software Engineering Education and Training (CSEE&T),* San Francisco, 289-293.

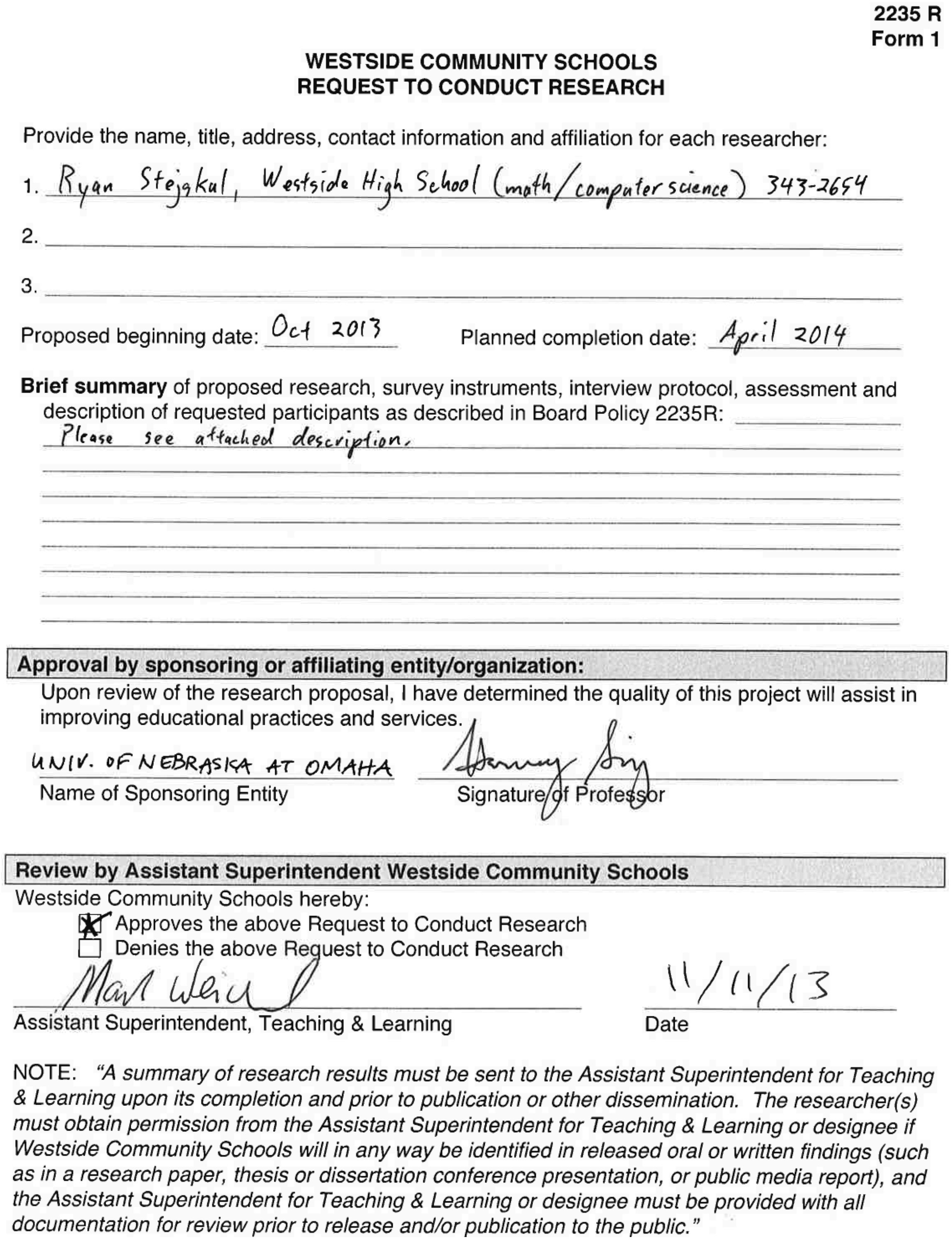
[31] University of Nebraska Omaha, 2014. CIST 1400, Introduction to Computer Programming. Retrieved April 9, 2014, from http://unomaha.smartcatalogiq.com/en/2013-2014/Undergraduate-Catalog/Undergraduate-Courses/CIST-College-of-Information-Science-Technology/1000/CIST-1400.

[32] The Web-CAT Community: Resources for automated grading and testing. Retrieved April 9, 2014, from http://web-cat.org/.

[33] Westside Community Schools, 2014. A Snapshot of Westside Community Schools. Retrieved April 9, 2014, from http://westside66.org/modules/groups/homepagefiles/cms/701798/File/2013-14%20Annual%20Report.pdf.

Appendix A - Permissions to Conduct Research

Westside High School approval to conduct research



2235 R
Form 1

WESTSIDE COMMUNITY SCHOOLS
REQUEST TO CONDUCT RESEARCH

Provide the name, title, address, contact information and affiliation for each researcher:

1. Ryan Stejskal, Westside High School (math/computer science) 343-2654
2.
3.

Proposed beginning date: Oct 2013 Planned completion date: April 2014

Brief summary of proposed research, survey instruments, interview protocol, assessment and description of requested participants as described in Board Policy 2235R:
Please see attached description.

Approval by sponsoring or affiliating entity/organization:

Upon review of the research proposal, I have determined the quality of this project will assist in improving educational practices and services.

UNIV. OF NEBRASKA AT OMAHA [illegible]
Name of Sponsoring Entity Signature of Professor

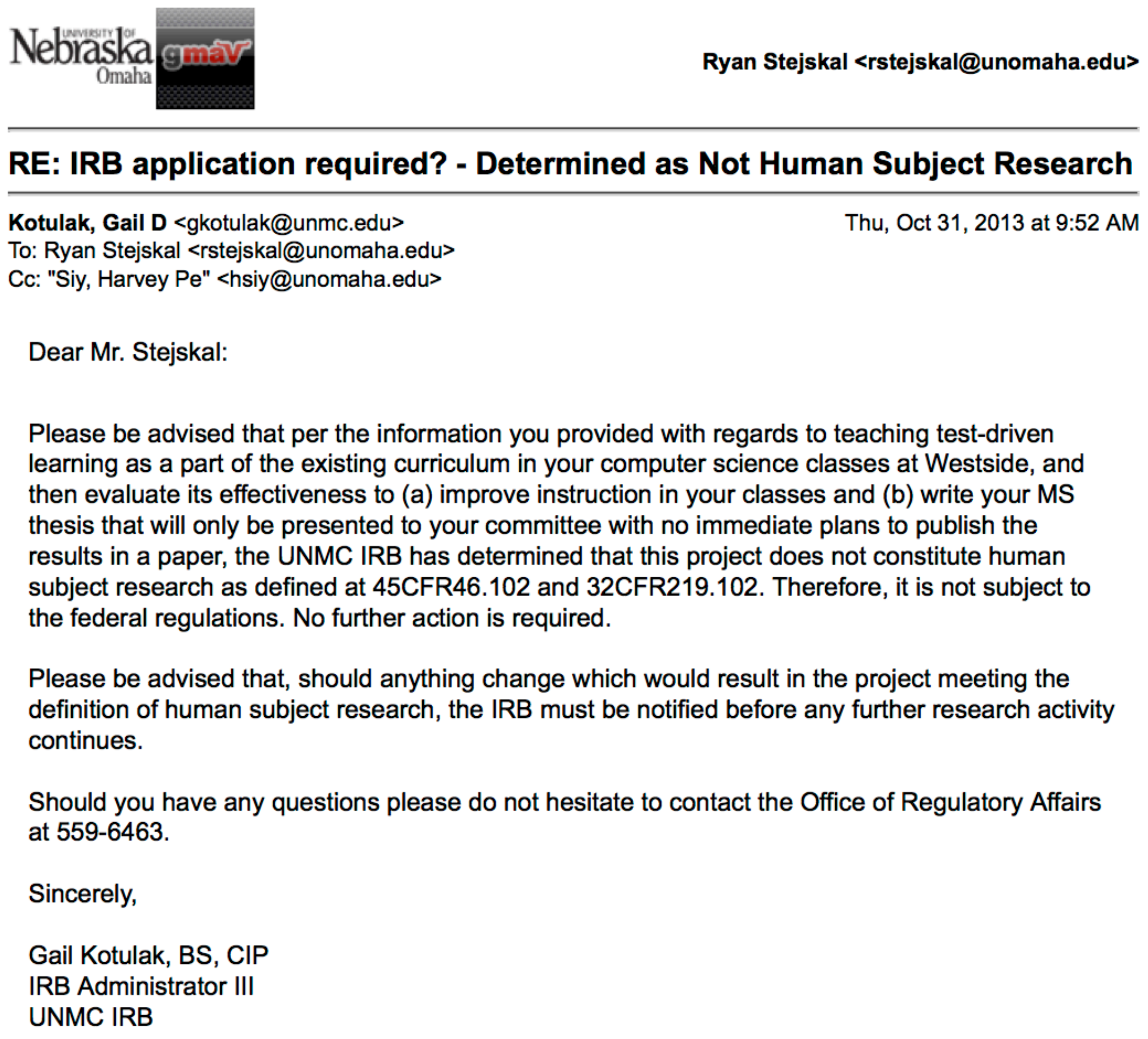
Review by Assistant Superintendent Westside Community Schools

Westside Community Schools hereby:
[✓] Approves the above Request to Conduct Research
☐ Denies the above Request to Conduct Research

[illegible] 11/11/13
Assistant Superintendent, Teaching & Learning Date

NOTE: *"A summary of research results must be sent to the Assistant Superintendent for Teaching & Learning upon its completion and prior to publication or other dissemination. The researcher(s) must obtain permission from the Assistant Superintendent for Teaching & Learning or designee if Westside Community Schools will in any way be identified in released oral or written findings (such as in a research paper, thesis or dissertation conference presentation, or public media report), and the Assistant Superintendent for Teaching & Learning or designee must be provided with all documentation for review prior to release and/or publication to the public."*

UNMC IRB Communication: Study not subject to federal regulations



Nebraska Omaha gmail

Ryan Stejskal <rstejskal@unomaha.edu>

RE: IRB application required? - Determined as Not Human Subject Research

Kotulak, Gail D <gkotulak@unmc.edu> Thu, Oct 31, 2013 at 9:52 AM
To: Ryan Stejskal <rstejskal@unomaha.edu>
Cc: "Siy, Harvey Pe" <hsiy@unomaha.edu>

Dear Mr. Stejskal:

Please be advised that per the information you provided with regards to teaching test-driven learning as a part of the existing curriculum in your computer science classes at Westside, and then evaluate its effectiveness to (a) improve instruction in your classes and (b) write your MS thesis that will only be presented to your committee with no immediate plans to publish the results in a paper, the UNMC IRB has determined that this project does not constitute human subject research as defined at 45CFR46.102 and 32CFR219.102. Therefore, it is not subject to the federal regulations. No further action is required.

Please be advised that, should anything change which would result in the project meeting the definition of human subject research, the IRB must be notified before any further research activity continues.

Should you have any questions please do not hesitate to contact the Office of Regulatory Affairs at 559-6463.

Sincerely,

Gail Kotulak, BS, CIP
IRB Administrator III
UNMC IRB

Appendix B - Definitions of Terms

The following specific terms are used within this study:

Test-driven development (TDD) is an industry practice where a test-focused approach is taken to software development. The key aspects of a TDD approach include creation of test code before production code, a short development cycle involving the creation of small units of test and production code, and the use of automated testing tools to execute and maintain test code.

Test-driven learning (TDL) is the application of TDD techniques to an educational setting. While TDD and TDL are very similar, the distinction between the two emphasizes that the goals of TDD and TDL are not identical. In particular, TDD places an emphasis on regression testing and maintenance, which is often a low priority in education since the lifetime of a program is usually limited by the length of a course.

Test-first is a software development approach where test cases are written before production code. Test-first is an essential component of TDD, but test-first development is not entirely synonymous with TDD.

Test-last is the alternative to test-first; test-last development means that production code is written first and then test cases are developed. Test-last development is more familiar to most developers than test-first.

Unit tests are the individual test cases produced during TDD or TDL. A unit test focuses on testing an individual unit of a program (typically a function or a class).

Automated testing tools are software tools that facilitate the creation and execution of unit tests.

JUnit [18] is an automated testing tool commonly used with the Java programming language for creating unit tests. Unit tests in JUnit are written as methods within a Java class. JUnit unit tests are identified either with specific method names starting with "test" (in version 3 of JUnit) or with the Java annotation @Test (in version 4).

AP Computer Science (AP CS) is the high school computer science course curriculum specified by the College Board [9], taught in Java and intended to be roughly equivalent to a university first-semester computer science course.

Appendix C - Copies of Survey Instruments

Survey #1 (given after Linked List Implementation project)

Linked List project evaluation

Please complete this survey about the chapter 4 project (linked list). This survey is anonymous; please do not enter your name or any other information that would identify you. Other than the first question (research permission), you may skip any item that you do not feel comfortable answering or that you feel you do not have sufficient information to answer. Thank you!
* Required

May Mr. Stejskal use your survey responses in his research at UNO? *
* Mr. Stejskal is currently studying the impact of various teaching techniques in computer science at UNO. If you check "YES" to this question, Mr. Stejskal may use your responses on this survey in this research. Any data you submit will be used only in combination with other students' data; no student will ever be named personally. If you check "NO", Mr. Stejskal will still consider your responses as he seeks to improve his teaching at Westside, but will not include your response in his research data.

○ Yes
○ No

Evaluate the linked list project as a whole:

How hard did you find the linked list project, compared to other projects so far in this course?

	1	2	3	4	5	
Very easy	○	○	○	○	○	Very hard

Which technique(s) did you use to help verify the correctness of your project and/or to debug it?
Check all that apply.

☐ Used Mr. Stejskal's main() function to check the results
☐ Modified the main() function to include my own tests
☐ Added System.out.println() statements to check output
☐ Used the debugger in NetBeans
☐ Ran the project against the JUnit tests we created in class
☐ Created my own JUnit tests
☐ Other:

How helpful were each of the following items in understanding the requirements in the linked list project?

The written project description posted on Blackboard

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Reading the JavaDoc comments included in IntLinkedList.java

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

The list diagrams (blue note sheet handed out in class)

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

In-class discussion of the project

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Discussing the requirements with classmates

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Writing the JUnit tests in class

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Testing your code with the JUnit tests

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Testing your code with the sample main() function

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Any other items that you used to help understand the project requirements?

Any other comments about the linked list project?

Evaluate JUnit in particular

Please indicate to what degree you agree or disagree with each of the following. (This topic is new this year, so Mr. Stejskal wants to evaluate it a little more.) Just a couple more questions - thank you!

I think that JUnit is a useful tool.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I would like to continue to use JUnit in future projects.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I understand JUnit well enough to write my own test cases.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Any other comments about JUnit in particular?

Survey #2 (given after Address Book project)

TDD Survey #2

* Required

Does Mr. Stejskal have permission to use your survey responses in his research at UNO? *
Mr. Stejskal is currently studying the impact of various teaching techniques in computer science at UNO. If you check "YES" to this question, Mr. Stejskal may use your responses on this survey in this research. Any data you submit will be used only in combination with other students' data; no student will ever be named personally. If you check "NO", Mr. Stejskal will still consider your responses as he seeks to improve his teaching at Westside, but will not include your response in his research data.

- ○ Yes
- ○ No

When we first started learning about test-driven development (JUnit), what was your opinion of it? *
For example, did you think it would be helpful, or that it would be too hard? Why did you think that?

What is your current opinion of test-driven development? *
In particular, has learning more about JUnit changed your opinions?

What advantages do you see in test-driven development? *
Tell me some specific examples from your personal experience, if possible.

What disadvantages do you see in test-driven development? *
Again, give examples from your personal experience, if possible.

Any other comments about JUnit or test-driven development?

Survey #3 (given after Cops & Robbers project)

TDD Final Survey

This survey is divided into two parts. In the first part, you will be asked to give answers specifically relating to the GridWorld Cops & Robbers project and your use of JUnit in that project. The second part contains questions about your programming experiences and experiences with JUnit in general. As usual, all your answers will be anonymous. To help protect your anonymity, please do not include your name or other identifying information in your responses.

* Required

Does Mr. Stejskal have permission to use your survey responses in his research at UNO? *
Mr. Stejskal is currently studying the impact of various teaching techniques in computer science at UNO. If you check "YES" to this question, Mr. Stejskal may use your responses on this survey in this research. Any data you submit will be used only in combination with other students' data; no student will ever be named personally. If you check "NO", Mr. Stejskal will still consider your responses as he seeks to improve his teaching at Westside, but will not include your response in his research data.

○ Yes

○ No

Part 1 - Questions about GridWorld Cops & Robbers

If you're not finished with the project yet, please answer based on your work so far. You can skip questions if they're not relevant.

How long did you spend on each of these activities for GridWorld Cops & Robbers?

If you're not sure, just estimate as best you can.

Planning your project

Writing project code

Writing JUnit test code

Testing your project
This includes both testing with JUnit and testing by running your project.

How helpful were each of the following in completing the GridWorld Cops & Robbers project?

The written project description posted on Blackboard

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Discussing the project with others, such as classmates or Mr. Stejskal

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Writing JUnit tests for Bank and Robber

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Testing your project by running it

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Testing your project with JUnit tests

	1	2	3	4	5	
Not at all helpful	○	○	○	○	○	Very helpful

Please indicate to what extent you agree or disagree with the following...

For these questions, answer with regard to the GridWorld Cops & Robbers project only.

I understood JUnit well enough to create test cases for Bank and Robber.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Creating JUnit tests helped me think about what Bank and Robber should do.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I feel that my JUnit tests adequately tested my program code.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Part 2 - Questions about your programming background and about JUnit

Please answer these questions based on your overall experiences, rather than your experience with any specific project.

Before enrolling in AP CS, how much programming experience did you have?
(This includes Intro CS and, if you've taken it, C++, as well as any substantial programming experience outside of classes.)

Please rate yourself on how often you do each of the following in AP CS.

Begin projects as soon as they are assigned

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

Begin projects near the deadline

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

Plan your code before you begin writing

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

Develop and test code in small incremental portions

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

Develop test cases before writing solution code
(When writing test cases for a project.)

	1	2	3	4	5	
Very rarely	○	○	○	○	○	Very often

Please rate to what extent you agree or disagree with each of the following statements.

"TDD" stands for "test-driven development." In particular, this means writing test cases before writing the corresponding solution code. It also includes using automated tools (for us, JUnit) to perform program testing.

I consistently followed TDD in programming projects as taught in this course.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD was easy to understand.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD helped me write better test code.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD helped me write better program code.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD helped me understand project requirements.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD helped me better design my programs.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD took time that I could have better used elsewhere.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

TDD saved me time by helping me with design and/or debugging.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

For programming I may pursue beyond AP CS, I would consider using TDD.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

Finally, please respond to these questions in a little more detail.

You're almost there! Your thoughtful answers to these last few questions will help Mr. Stejskal improve this course in the future.

When we first started learning about TDD, what was your opinion of it?

What is your current opinion of TDD?

What advantages and/or disadvantages do you see in TDD, as compared to writing code and then testing it?

What differences do you think might apply to using TDD in a business project rather than in a school class?

The usage of TDD is somewhat different in industry and in academics. Why do you think this might be? Do you think that TDD would be more or less useful in the CS industry?

Do you have any other comments regarding JUnit or TDD?

Appendix D - Data from Student Surveys and Project Grades

Survey #1 Data

Students responded to the following Likert item questions on this survey:

- **1.** How hard did you find the linked list project, compared to other projects so far in this course? (1 = very easy, 5 = very hard)
- How helpful were each of the following items in understanding the requirements in the linked list project? (1 = not at all helpful, 5 = very helpful)
    - **2.** The written project description posted on Blackboard
    - **3.** In-class discussion of the project
    - **4.** Discussing the requirements with classmates
    - **5.** Writing the JUnit tests in class
    - **6.** Testing your code with the JUnit tests
    - **7.** Testing your code with the sample main() function
    - **8.** Reading the JavaDoc comments included in IntLinkedList.java
    - **9.** The list diagrams (blue note sheet handed out in class)
- Please indicate to what degree you agree or disagree with each of the following. (1 = strongly disagree, 5 = strongly agree)
    - **10.** I think that JUnit is a useful tool.
    - **11.** I would like to continue to use JUnit in future projects.
    - **12.** I understand JUnit well enough to write my own test cases.

These are the results of these items:

Table 4: Item responses from survey 1 (Linked List project)

	Question Number (questions listed above)											
Student	1	2	3	4	5	6	7	8	9	10	11	12
	Responses from 2012-13 survey											
Student 1	4	3	4	4	2	4	4	4	1	5	4	4
Student 2	4	3	3	4	5	5	5	4	5	5	5	3
Student 3	4	3	2	2	4	5	4	4	3	4	4	3
Student 4	2	4	4	4	4	4	4	2	4	5	5	4
Student 5	2	4	4	4	4	4	4	2	4	5	5	4
Student 6	4	4	5	3	5	5	5	5	5	5	5	3
Student 7	2	4	4	4	5	5	3	3	5	5	5	5
	Responses from 2013-14 survey											
Student 1	4	5	5	3	5	5	5	1	5	5	5	2
Student 2	3	4	5	4	5	5	5	5	5	5	5	4
Student 3	5	2	2	2	4	4	3	2	3	5	5	4
Student 4	4	3	4	3	4	3	4	3	5	5	5	2
Student 5	3	3	2	1	5	4	2	5	5	5	5	4
Student 6	4	4	3	2	5	4	3	3	3	5	4	4
Student 7	5	2	3	1	3	3	3	3	5	5	5	5
Student 8	4	4	4	4	5	4	5	2	5	3	4	2

The total number of each response given to the Likert items from the previous table is summarized below:

Table 5: Item summaries from survey 1 (Linked List project)

Item	2012-13 results					2013-14 results				
	1	2	3	4	5	1	2	3	4	5
How hard did you find the linked list project, compared to other projects so far in this course? *(1 = very easy, 5 = very hard)*		3		4				2	4	2
How helpful were each of the following items in understanding the requirements in the linked list project? (1 = not at all helpful, 5 = very helpful)										
The written project description posted on Blackboard			3	4			2	2	3	1
In-class discussion of the project		1	1	4	1		2	2	2	2
Discussing the requirements with classmates		1	1	5		2	2	2	2	
Writing the JUnit tests in class		1		3	3			1	2	5
Testing your code with the JUnit tests				3	4			2	4	2
Testing your code with the sample main() function			1	4	2		1	3	1	3
Reading the JavaDoc comments included in IntLinkedList.java		2	1	3	1	1	2	3		2
The list diagrams (blue note sheet handed out in class)	1		1	2	3			2		6
Please indicate to what degree you agree or disagree with each of the following. (1 = strongly disagree, 5 = strongly agree)										
I think that JUnit is a useful tool.				1	6			1		7
I would like to continue to use JUnit in future projects.				2	5				2	6
I understand JUnit well enough to write my own test cases.			3	3	1		3		4	1

Students were also asked which techniques they used to help verify the correctness of the linked list project. For this question, students were given a range of choices:

1. Used Mr. Stejskal's main() function to check the results
2. Modified the main() function to include my own tests
3. Added System.out.println() statements to check output
4. Used the debugger in NetBeans
5. Ran the project against the JUnit tests we created in class
6. Created my own JUnit tests
7. Other (specify)

These were students' responses to this question (items marked "Yes" were used by students):

Table 6: Verification techniques used by students on Linked List project

	Verification technique (as listed above)						
	1	2	3	4	5	6	Other…
	Responses from 2012-13 survey						
Student 1	Yes	Yes	Yes	Yes	Yes		Google
Student 2	Yes	Yes			Yes		
Student 3	Yes	Yes	Yes		Yes		
Student 4	Yes	Yes			Yes		
Student 5	Yes	Yes			Yes		
Student 6	Yes	Yes	Yes		Yes		
Student 7			Yes		Yes		
	Responses from 2013-14 survey						
Student 1	Yes			Yes	Yes		
Student 2	Yes				Yes		
Student 3	Yes			Yes	Yes		
Student 4	Yes		Yes		Yes		
Student 5	Yes				Yes		
Student 6	Yes				Yes		
Student 7		Yes			Yes		
Student 8	Yes		Yes		Yes		

Finally, there were two free-response questions on this survey: "Any other comments about JUnit in particular?" and "Any other comments about the linked list project?" Many of the students did not respond to these questions. These are the responses of the students who did respond:

- Any other comments about JUnit in particular?

  2012-13 student #2: I thought it was very useful.

  2012-13 student #7: How much more is there to learn about JUnit tests?

  2013-14 student #2: I think we should use JUnit test as often as we can to help us understand abstract programs much better by knowing what the output should be.

  2013-14 student #8: JUnit seems fairly helpful but I do not think that there was much instruction given on how to make them so it feels like another project itself.

- Any other comments about the linked list project?

2012-13 student #1: I honestly believe that it would be easier had I been able to use an older model of my own design. Having to change from something you've already done and use something else is a terrble thing to have to do.

2012-13 student #2: I really enjoyed it, it made me think about exactly what I was doing so I wouldn't mess up.

2012-13 student #3: It was a little confusing to me but once I got one method completed, the others were easy.

2012-13 student #7: It was fun.

2013-14 student #1: Most of the difficulty came from one thing that I did wrong, so once that was fixed everything fell into place.

2013-14 student #8: I liked how we did not have to start from scratch and that it was just the main ideas of the project that we had to focus on.

Survey #2 Data

This survey contained five free-response questions. The questions on this survey are as follows:

1. When we first started learning about test-driven development (JUnit), what was your opinion of it?
2. What is your current opinion of test-driven development?
3. What advantages do you see in test-driven development?
4. What disadvantages do you see in test-driven development?
5. Any other comments about JUnit or test-driven development?

Responses to this survey are given by student. For each student, responses are numbered by question number as listed above.

Student 2.1:

#1: I don't like it, I never liked it. It's too time consuming when it's already easy to see and find where something goes wrong in a program.

#2: No, I do not like JUnit as stated above.

#3: The only advantage I see is if something works or does not work.

#4: In my projects I've been stressing out way to much to make a JUnit test along with the actual code.

#5: OMGWTFBBQ BURN JUNIT! KILLIT BANG!

Student 2.2:

#1: It was a new way to think about programming. Creating standards for your program and then program it so it can pass all the tests.

#2: It is still the same way as when I first learned about it.

#3: Doing this will eliminate bugs.

#4: It takes longer to program something.

#5: None.

Student 2.3:

#1: It was Okay for that it helps make the program more efficient but it is too hard to write more test program

#2: Yes because I noticed that the program became more efficient

#3: Reduced the error in the program

#4: Takes longer time

#5: N/A; JUnit is all good

Student 2.4:

#1: I thought it was unnecessary because we can test the program by running it.

#2: It's very useful because we don't test everything when running a program

#3: It lets us test for things we can test by running our program (testing if something throws an exception is an example) and tells us why things don't pass the tests.

#4: It takes a while to think of tests.

#5: They're useful

Student 2.5:

#1: I did not really get the purpose of J-unit because I didn't know the importance of testing programs, that being said, I thought it was easy to make because you expect certain results, majority of the time your results should be predictable.

#2: I think it is helpful in many projects a good example is In the linked list project; it helped me determine results, without it I would of struggled a lot more than I did.

#3: I know what parts of my code work and what parts do not, It helps me determine where to look and that could be in the junit test or in my code. It a helpful debugging tool for making abstract project esp. in linked list as I mentioned.

#4: That it does not fully certify that your code will work 100%, because my linked list worked for all of the J-Unit tests however it did not work as efficient as the linked list Implemented in java.

Student 2.6:

#1: I thought that it seemed hard, mostly because I had never done things like that before.

#2: It could be useful, but i'd rather just plan out on paper instead of writing tests.

#3: It makes you think about what you need to make your objects do.

#4: You can do what test driven programing can do with just a sheet of paper.

Student 2.7:

#1: I honestly thought it would be more work than it was worth. This was mostly because it seemed like you had to create another program to test your current program when you could just run the program to test it.

#2: I think it can be helpful for certain programs, but I don't think you need to use it for all programs.

#3: I think that it is a sort of design in that you know what you are going to create before you create it. It also always you to check all or mostly all circumstances of a particular situation, so that makes it useful.

#4: It can sometimes take a lot of work to make tests. Also, your program may change from the time you take the test and that makes the tests worthless.

Student 2.8:

#1: I thought the JUnit tests would be helpful to debug the program we had just coded but only to a certain degree. I thought that just looking at the code direct was more effective and timely because you have to code to tests as well as the main program. Also you may run into problems just coding the test and if they are incorrect then you may never know if the main program is entirely correct.

#2: I am still skeptical as to how effective JUnit tests are on small scale programs. Because the tests do not seem to be worth the time and effort for small programs. However throughout learning more about JUnit test I realize that they can be helpful to debug especially larger and/or complex programs. Because in larger programs it may take an extremely long time to pinpoint the issue and JUnit tests can do just that.

#3: I have found the main advantage to be pinpointing the problem. When I am creating a large program with multiple classes it can be rather difficult to pinpoint the issue and JUnit tests can help me do that. The bulk of my time, when programming, is spent debugging.

#4: One main disadvantage is that JUnit tests can be timely and inaccurate. With smaller programs I have found that even though the JUnit tests were smaller they still took a considerable amount of time to make. The time investment could have been worth making but at first I struggle to make the tests so they were often inaccurate. So in the tests seemed unnecessary.

#5: Overall I think the JUnit tests/test-driven development are valuable for a programmer to learn but I would suggest a more gradual transition into the unit. It could help to wait until the programs require JUnit tests and the programmers is more confident in making the tests. The unit does seem to be a core principle of coding that every programmer should learn.

Survey #3 Data

This survey contained both Likert items and free-response questions. The Likert data is presented first (response count totals are on the next page):

Table 7: Item responses from survey 3 (GridWorld Chase project)

	Student number								
Question	1	2	3	4	5	6	7	8	9
How helpful were each of the following in completing the GridWorld Cops & Robbers project? (1 = not at all helpful, 5 = very helpful)									
The written project description posted on Blackboard	5	5	5	5	4	3	4	4	3
Discussing the project with others, such as classmates or Mr. Stejskal	5	4	3	2	1	5	5	5	5
Writing JUnit tests for Bank and Robber	3	3	4	4	3	3	5	4	4
Testing your project by running it	5	2	5	5	4	5	3	5	5
Testing your project with JUnit tests	5	1	5	5	4	4	5	4	3
Please indicate to what extent you agree or disagree with the following, with regard to the GridWorld Cops & Robbers program.(1 = strongly disagree, 5 = strongly agree)									
I understood JUnit well enough to create test cases for Bank and Robber.	5	2	5	5	4	5	5	3	4
Creating JUnit tests helped me think about what Bank and Robber should do.	3	3	4	3	5	5	5	4	5
I feel that my JUnit tests adequately tested my program code.	5	4	4	4	5	4	5	4	3
Please rate yourself on how often you do each of the following in AP CS. (1 = very rarely, 5 = very often)									
Begin projects as soon as they are assigned	3	1	1	3	4	2	3	1	2
Begin projects near the deadline	3	1	5	3	1	4	4	3	4
Plan your code before you begin writing	2	5	3	3	5	2	5	2	3
Develop and test code in small incremental portions	1	1	5	5	3	3	4	4	2
Develop test cases before writing solution code	2	1	2	2	2	1	3	1	2
Please indicate to what extent you agree or disagree with the following... (1 = strongly disagree, 5 = strongly agree)									
I consistently followed TDD in programming projects as taught in this course.	3	1	3	3	4	3	5	3	3
TDD was easy to understand.	5	1	5	4	4	5	5	2	4
TDD helped me write better test code.	5	1	5	4	4	5	4	4	4
TDD helped me write better program code.	5	2	4	4	4	5	4	4	4
TDD helped me understand project requirements.	5	4	3	2	4	5	4	4	5
TDD helped me better design my programs.	5	5	5	5	5	5	5	5	5
TDD took time that I could have better used elsewhere.	2	5	2	3	5	2	2	3	3
TDD saved me time by helping me with design and/or debugging.	5	2	4	4	4	5	5	3	3
For programming I may pursue beyond AP CS, I would consider using TDD.	5	1	4	5	4	5	5	4	4

The number of students giving each response to the Likert items of the survey are summarized here.

Table 8: Item summaries from survey 3 (GridWorld Chase project)

	Response count				
Question	1	2	3	4	5
How helpful were each of the following in completing the GridWorld Cops & Robbers project? (1 = not at all helpful, 5 = very helpful)					
The written project description posted on Blackboard			2	3	4
Discussing the project with others, such as classmates or Mr. Stejskal	1	1	1	1	5
Writing JUnit tests for Bank and Robber			4	4	1
Testing your project by running it		1	1	1	6
Testing your project with JUnit tests	1		1	3	4
Please indicate to what extent you agree or disagree with the following, with regard to the GridWorld Cops & Robbers program.(1 = strongly disagree, 5 = strongly agree)					
I understood JUnit well enough to create test cases for Bank and Robber.		1	1	2	5
Creating JUnit tests helped me think about what Bank and Robber should do.			3	2	4
I feel that my JUnit tests adequately tested my program code.			1	5	3
Please rate yourself on how often you do each of the following in AP CS. (1 = very rarely, 5 = very often)					
Begin projects as soon as they are assigned	3	2	3	1	
Begin projects near the deadline	2		3	3	1
Plan your code before you begin writing		3	3		3
Develop and test code in small incremental portions	2	1	2	2	2
Develop test cases before writing solution code	3	5	1		
Please indicate to what extent you agree or disagree with the following... (1 = strongly disagree, 5 = strongly agree)					
I consistently followed TDD in programming projects as taught in this course.	1		6	1	1
TDD was easy to understand.	1	1		3	4
TDD helped me write better test code.	1			5	3
TDD helped me write better program code.		1		6	2
TDD helped me understand project requirements.		1	1	4	3
TDD helped me better design my programs.					9
TDD took time that I could have better used elsewhere.		4	3		2
TDD saved me time by helping me with design and/or debugging.		1	2	3	3
For programming I may pursue beyond AP CS, I would consider using TDD.	1			4	4

Students were also asked to answer five short-answer questions on this survey:

- How long did you spend on each of these activities for GridWorld Cops & Robbers?
  - **1.** Planning your project
  - **2.** Writing project code
  - **3.** Writing JUnit test code
  - **4.** Testing your project (this includes both testing with JUnit and testing by running your project)
- **5.** Before enrolling in AP CS, how much programming experience did you have? (This includes Intro CS and, if you've taken it, C++, as well as any substantial programming experience outside of classes.)

These are the responses that students gave to these items:

Table 9: Short-answer item responses from survey 3 (GridWorld Chase project)

	Item responses				
Student	Planning	Project writing	JUnit writing	Testing	Previous experience
1	5 seconds	2 hours	1 hour	5 minutes	Intro CS
2	2 hours	2 hours	3 hours	45 minutes	Only a basic code class
3	20 min	2-3 hours	30-40 min	15 min	intro cs
4	10 minutes	3 hours	30 minutes	20 minutes	Intro CS
5	2 to 3 days	3 to 4 days	.5 to 2 days	half day	2 years of experiences
6	5 minutes	60 minutes	30 minutes	5 minutes	2 years
7	about 10 minutes, just thinking nothing written.	one hour	30 minutes	15 minutes	intro CS
8	40 minutes	5 hours	1 hour	10 minutes	Intro CS and C++
9	5 minutes	1 hour	30 minutes	5 minutes	2 years

Finally, five free-response questions were presented to students on this survey:

1. When we first started learning about TDD, what was your opinion of it?
2. What is your current opinion of TDD?
3. What advantages and/or disadvantages do you see in TDD, as compared to writing code and then testing it?

4. What differences do you think might apply to using TDD in a business project rather than in a school class?
5. Do you have any other comments regarding JUnit or TDD?

Students' responses to these questions are given below, numbered by student and by question number as listed within each student's responses.

<u>Student 3.1:</u>

#1: It was putting the cart before the horse

#2: The horse can push the cart just fine

#3: TDD gives an outline as to what things should do.

#4: Designing a boss in a game to do a lot of damage in many ways.

#5: It's very useful.

<u>Student 3.2:</u>

#1: I thought it was going to help my programming a lot better

#2: It didn't help me at all. Other than knowing that I can use it to see the requirements.

#3: You knew your requirements

#4: N/A

#5: I was really excited for it but it never met my standards of liking. So I discarded and continued with the actual code rather than the tests.

<u>Student 3.3:</u>

#1: I though it was a waste of time.

#2: It is useful in big projects, or on projects where you jump into already written code, like grid world

#3: pros: saves some time, makes design a bit better; cons: time may be used better, only helpful for big projects

#4: It is useful in CS, and in other businesses as you have to plan ahead when doing anything large scale, be it a project or a business decision.

<u>Student 3.4:</u>

#1: I didn't think it was very useful as I could just write my code and test it in the main class instead of writing an entirely different set of code.

#2: I think it can be very helpful to know what you want your program to do, although the programs that we've made in class I haven't felt are big enough to really take full advantage of it.

#3: It allows you to know what you need to code before hand, and also can also test your program under many different scenarios very quickly. However, I don't think that for some of the projects we've done it was necessary as I could very easily test them in the main class/gui.

#4: I think it would be much more effective because you've got a lot of variables in business and being able to test them very quickly would be advantageous.

<u>Student 3.5:</u>

#1: It was good

#2: Nothing

#3: All I see is that it is quicker way to do and finish. That is the advantage.

#4: It would be more useful in the CS industry

<u>Student 3.6:</u>

#1: Programming a program would take forever.

#2: It takes more time developing a program and you need to know what your program is going to do before you start building it.

#3: If your code passes all the test, most likely your code is going to run the way you wanted it, but a downfall is that your test code needs to be strict and not loose in order to have your program to pass and work well.

#4: TDD would be useful in CS industry, but I feel a lot of companies might skip that step, just so they can throw a product out and then send out repair or update packages because that will earn more money for them.

#5: It should definitely be taught in every class, not just in CS, but in engineering classes. Think before you build.

<u>Student 3.7:</u>

#1: I thought that TDD was a bit over the top, probably because the program was simple when we first wrote them.

#2: I think TDD is useful, it allows me to easily understand what the program should do and lets me understand the program faster. TDD helps narrow down errors whether they are in the tests themselves or in your code.

#3: It can be unnecessary at times but I think if you really have a solid understanding on the program you are going to write, you should go without them. This is if your program is on a smaller scale only.

#4: It would show more usefulness in a business project because the code we write in class is examples of how it is used. A business would most likely have a large problem that would be hard to write without TDD in to my opinion.

#5: Junit would be good with a larger group project.

Student 3.8:

#1: I thought that it could be rather helpful in designing the main project.

#2: I think that it is a core part of writing code and should be taught alongside core curriculum. Although it should be fully understood before started otherwise it could potentially be a waste of time.

#3: The main advantage is that it helps for understanding and writing the program and potentially save time writing the program. I see no obvious disadvantage to TDD as long as it is fully understood.

#4: I think that it would make for a better structured more well rounded project and possibly more complete (covers topics that could have been missed). I think TDD would be different in industry because in academics TDD is used as a learning tool while in industry TDD is used to produce real world results. In industry one would use TDD as tool to further the industry and this would be, in my opinion. more effective in the CS industry because TDD could develop both the CS side and industry side.

#5: Overall TDD is an effective and useful tool. I believe that TDD can and should be used to help a programmer learn the core curriculum of computer science. However in my personal experience, I unfortunately had not fully understood Junit well enough to implement TDD in my coding process. When we first began creating Junit tests I saw it as a waste of time because I did not use it to help me code and it did take much longer to code than if I were to use TDD. I do not blame the curriculum, I just think that a greater emphasis should be put on understanding Junit.

Student 3.9:

#1: That is was a waste of time

#2: That is necessary and should be worked on first when writing a program.

#3: It takes more time when writing a program and harder to see that it completes what it needs to complete compared to testing it like a consumer.

#4: I don't think business's use TDD because they want to make money quickly and then send updates to patch the problems in it.

#5: All schools and businesses should use it and not just in computer programming classes.

Project Grade Data

Finally, student letter grades are presented for the 2013-14 school year (where TDL was used) and for the 2010-11 and 2011-12 school years (where TDL was not used). For each project, the letter grades only of students who submitted the project are given. Grades are in descending order and do not correspond student-to-student from one project to the next.

Table 10: Student performance on TDL projects over three school years

Project	Student Grades		
	2010-11 (no TDL), 10 students enrolled	2011-12 (no TDL), 9 students enrolled	2013-14 (TDL), 11 students enrolled
Heroes vs. Villains	A+, A+, A, A, A, B+, B, D+, D+	A+, A, B+, C+, C, C, C, D+	A+, A+, A, A, A, A, A, A, B, D+, D
Linked List	A+, A, A, B, B, D+, D, D, D	A, B, B, B, C, C, D	B, B, B, B, B, D, F
Shape Painter	A, A, A, A, B+, B, C	A+, A+, A+, A, B+, B, B, B, D	A+, A+, A+, A, B+ B, B, B, C+
Address Book	A+, A+, A, A, A, B+, C+, C, D+	A+, A+, A, A, B+, B+, C+	A+, A, A, B+, B+, B+, D+
GridWorld Chase	A+, A+, A, A, C+, C, D+, D	A+, A+, B+, B+, B, C+, C, C	A+, A+, A, A, B, D+

Appendix E - Descriptions of Projects using TDL in AP CS

AP Computer Science
Chapter 2 Programming Project

Heroes vs. Villains

In this project, your job is to write three classes. You'll create `Heroes` and `Villains` which battle each other, and a `Battle` class which holds the heroes and monsters as they fight.

Your `Hero` class should have the following members:

- Three member variables. One will hold the hero's name (a string), one will hold its hit points (which start at 20 and decrease from there), and one will hold the hero's strength (an integer between 1 and 10).
- A constructor which takes the hero's name and strength as parameters, and initializes the member variables.
- Three get methods (`getName`, `getStrength`, and `getHP`), which return the values of the hero's member variables.
- A method named `isKOd`, which returns true if the hero has been knocked out (its HP are 0 or less) and false otherwise.
- A method named `getHit`, which takes a `Villain` as a parameter. When a hero gets hit by a villain, the hero loses 2 HP, and prints a message like this: "Lex Luthor hits Superman! Superman has 18 HP left."
- A static member variable which keeps track of the total damage that's been dealt to all the heroes.
- A static method named `getTotalDamage`, which returns the amount of damage that has been dealt to all heroes combined.

Your `Villain` class should be similar to the `Hero` class. Here are the differences:

- Villains don't have a strength attribute, so their constructor only has one parameter, and they don't have a `getStrength` method either.
- A Villain gets hit by a hero, not a villain (obviously). When a Villain gets hit, it loses an amount of HP equal to the hero's strength. (So if Superman hits Lex Luthor and Superman's strength is 8, Lex Luthor loses 8 HP.)

Your `Battle` class should have the following features:

- Two member variables (one for heroes and one for villains), which should be arrays or `ArrayLists`. (You can choose which you prefer, but I strongly recommend `ArrayLists`, because they'll be a lot easier to use for the rest of the features of the class.)
- A constructor, which doesn't take any parameters, but creates your arrays or array lists.
- Methods named `addHero` and `addVillain`, which add a hero or a villain to the battle.
- Methods named `getHeroCount` and `getVillainCount`, which do what they say.
- Methods named `getHero` and `getVillain`, which take an integer index and return the corresponding hero or villain.

- A `doBattle` method, which causes the heroes and villains to do battle. Here's how the `doBattle` method works:
  - Pick one random hero and one random villain from the lists of heroes and villains.
  - The chosen hero and chosen villain hit each other. (This will cause messages to be printed by the `Hero` and the `Villain` that hit each other.)
  - If the hero is KO'd (its `isKOd` method returns true), then print a message like "Superman gets knocked out!", and remove the hero from the list of heroes. If the villain gets KO'd, do the same thing. (Note that it's possible that a hero and villain could knock each other out simultaneously.)
  - Repeat the above steps until either all the heroes or all the villains are KO'd.
  - At the end of the battle, report the winner (whichever side has members left standing), or if all the heroes *and* all the villains are KO'd, report that the battle was a draw.
  - Finally, report how much HP the heroes lost and how much HP the villains lost.

At the end of this assignment document, you can find a Main class that you can use to test your other three classes, and a sample output that shows one run of Mr. Stejskal's program using that main class. Keep in mind that, due to the use of random selections during the battle, the program's output will change somewhat from one run to the next.

Also, on Blackboard, Mr. Stejskal will post three JUnit test classes which you can use to help verify the correctness of your classes. (There is one test class for each of the three classes that you need to create.) *Warning*: the JUnit tests do not test all aspects of the three classes! In particular, the `doBattle` method is not tested by a JUnit test. Remember that failing a JUnit test is evidence that something is wrong in one of your classes, but passing all the JUnit tests (while a good sign) does *not* in itself guarantee the correctness of your code.

Here are a few hints to keep in mind as you work on this project:

- Although the classes in your project interact with each other, you can in large part work on one class at a time. In general, writing code in one class shouldn't have a big effect on the other classes.
- Remember that the various get methods in the `Hero` and `Villain` classes are there to be *used*. You will, for example, need to call `getName` on both heroes and villains within the `doBattle` method.
- It can be useful to create temporary variables. For example, here's the line of code I use to choose a hero from among the list of heroes:
  ```
      Hero h = heroes.get(r.nextInt(heroes.size()));
  ```
  (In this line, `r` is a `Random` object, and `heroes` is an ArrayList of `Hero` objects.)
- This isn't a particularly long program. My `Scene` class is 65 lines long, my `Hero` class is 51 lines, and my `Villain` class is 44 lines (but my code isn't sufficiently commented). Your program may be longer or shorter, of course, but if you find

yourself writing several hundred lines of code in one class, you might be working too hard.

--- SUBMISSION INSTRUCTIONS - READ THESE ---

This program is due **Thursday, October 3** at **11:59 PM**. Submissions received after this time are late and will receive grade deductions as specified in the syllabus. To submit, send a FirstClass message to **Mr. Stejskal's dropbox**. Attach the files Hero.java, Villain.java, and Scene.java to your message. (You don't need to attach Main.java unless you have optional extra challenges which are demonstrated by your Main.java.)

Optional Extra Challenges
There are a number of ways in which you can make the battle more interesting. Some ideas:

- Add an agility statistic to heroes and villains. Agility represents the character's percentage chance of dodging an attack; for example, a character with an agility of 20 has a 20% chance of dodging and receiving no damage.
- Randomize the damage. Instead of having a strength 8 hero deal 8 damage to a villain, have the hero deal a random amount of damage in the range [1..8]. Similarly, have the villains randomly deal either 1 or 2 damage to heroes.
- Add critical hits. Give every attack a 10% chance of being a critical hit. Critical hits display an extra message ("It's a critical hit!") and deal twice the normal damage.

Note that if you implement any of these extra challenges, the JUnit tests that Mr. Stejskal has supplied will not all pass (at least not without modifications). Complete the base project and ensure that your unit tests pass before moving on to optional challenges.

Grading Rubric - Heroes vs. Villains

Correctness

- Program compiles without error as submitted, using either the main class provided or a main class submitted with the project (1 pt) ________
- Hero and Villain have all required attributes (1 pt) ________
- Hero and Villain getXXX methods are present and work (1 pt) ________
- Hero and Villain getHit methods work as specified (1 pt) ________
- getTotalDamage methods present and work (1 pt) ________
- Scene class has variables to store heroes and villains (1 pt) ________
- addHero and addVillain methods work correctly (1 pt) ________
- Battle picks random heroes and villains and has them hit each other (1 pt) ________
- Battle removes KO'd heroes/villains from the battle and announces a winner when the battle is over (1 pt) ________
- Battle reports the total damage taken by heroes and villains (1 pt) ________

Clarity

- All attributes have appropriate names (2 pts) ________
- Each class is organized in a logical order (2 pts) ________
- Indentation is consistent with program structure (2 pts) ________
- Program is commented, min 2 comments per class (2 pts) ________

Efficiency

- Program is not obviously inefficient (2 pts) ________

Total (max 20 points) ________

19-20 pts:	A	18 pts:	B+
16-17 pts:	B	15 pts:	C+
13-14 pts:	C		

Less than 13 pts is incomplete and must be redone

Bonus (optional extra challenge)

- First optional addition (________________________________) __________
- Second optional addition (________________________________) __________

Two or more bonus points increase your grade by ½ letter (including raising A to A+).

Heroes vs. Villains

Mr. Stejskal's main class:

```
package battlescene;

public class Main
{
    public static void main(String[] args)
    {
        Scene s = new Scene();
        s.addHero(new Hero("Superman", 8));
        s.addHero(new Hero("Batman", 5));
        s.addHero(new Hero("Spider-man", 6));

        s.addVillain(new Villain("The Joker"));
        s.addVillain(new Villain("Lex Luthor"));
        s.addVillain(new Villain("The Penguin"));
        s.addVillain(new Villain("Two-Face"));
        s.addVillain(new Villain("Catwoman"));
        s.addVillain(new Villain("The Riddler"));

        s.doBattle();
    }
}
```

And one sample run that comes when Mr. Stejskal runs his program with this main class:

```
Two-Face hits Superman!  Superman has 18 HP left.
Superman hits Two-Face!  Two-Face has 12 HP left.

The Riddler hits Spider-man!  Spider-man has 18 HP left.
Spider-man hits The Riddler!  The Riddler has 14 HP left.

The Joker hits Spider-man!  Spider-man has 16 HP left.
Spider-man hits The Joker!  The Joker has 14 HP left.

The Joker hits Superman!  Superman has 16 HP left.
Superman hits The Joker!  The Joker has 6 HP left.

Lex Luthor hits Batman!  Batman has 18 HP left.
Batman hits Lex Luthor!  Lex Luthor has 15 HP left.

Catwoman hits Superman!  Superman has 14 HP left.
Superman hits Catwoman!  Catwoman has 12 HP left.

The Penguin hits Spider-man!  Spider-man has 14 HP left.
Spider-man hits The Penguin!  The Penguin has 14 HP left.

Lex Luthor hits Superman!  Superman has 12 HP left.
Superman hits Lex Luthor!  Lex Luthor has 7 HP left.

Catwoman hits Batman!  Batman has 16 HP left.
Batman hits Catwoman!  Catwoman has 7 HP left.

Lex Luthor hits Spider-man!  Spider-man has 12 HP left.
Spider-man hits Lex Luthor!  Lex Luthor has 1 HP left.
```

```
The Joker hits Batman!  Batman has 14 HP left.
Batman hits The Joker!  The Joker has 1 HP left.

The Joker hits Batman!  Batman has 12 HP left.
Batman hits The Joker!  The Joker has 0 HP left.
The Joker gets knocked out!

Two-Face hits Batman!  Batman has 10 HP left.
Batman hits Two-Face!  Two-Face has 7 HP left.

The Penguin hits Batman!  Batman has 8 HP left.
Batman hits The Penguin!  The Penguin has 9 HP left.

Catwoman hits Batman!  Batman has 6 HP left.
Batman hits Catwoman!  Catwoman has 2 HP left.

The Riddler hits Superman!  Superman has 10 HP left.
Superman hits The Riddler!  The Riddler has 6 HP left.

The Riddler hits Batman!  Batman has 4 HP left.
Batman hits The Riddler!  The Riddler has 1 HP left.

Two-Face hits Superman!  Superman has 8 HP left.
Superman hits Two-Face!  Two-Face has 0 HP left.
Two-Face gets knocked out!

Catwoman hits Spider-man!  Spider-man has 10 HP left.
Spider-man hits Catwoman!  Catwoman has 0 HP left.
Catwoman gets knocked out!

The Penguin hits Spider-man!  Spider-man has 8 HP left.
Spider-man hits The Penguin!  The Penguin has 3 HP left.

The Penguin hits Spider-man!  Spider-man has 6 HP left.
Spider-man hits The Penguin!  The Penguin has 0 HP left.
The Penguin gets knocked out!

Lex Luthor hits Spider-man!  Spider-man has 4 HP left.
Spider-man hits Lex Luthor!  Lex Luthor has 0 HP left.
Lex Luthor gets knocked out!

The Riddler hits Spider-man!  Spider-man has 2 HP left.
Spider-man hits The Riddler!  The Riddler has 0 HP left.
The Riddler gets knocked out!

The heroes win!

The heroes did 142 damage to the villains.
The villains did 46 damage to the heroes.
```

AP Computer Science
Chapter 4 Programming Project

Linked List Implementation

This chapter's assignment is to implement your own linked list class. You don't have to do this from the ground up, though. You've already seen the class framework on Blackboard; you just need to fill in the class's methods.

This linked list class will work a little differently (and a little more simply) than `java.util.LinkedList`. In particular, this class will not use iterators. Instead, the class will keep track of its own "current position", where data can be inserted or removed.

The linked list class contains three `ListNode` members:

- `first` is the first node in the list.
- `curr` is the "current node" in the list. Insertions happen *before* `curr`. Deletions happen at `curr` (if a node is deleted, `curr` is the node to delete).
- `prev` is the node *before* `curr`. Why do you need it? You'll need to modify `prev`'s `next` pointer whenever a node is inserted or deleted (we'll see that today).

As noted above, most of the structure of the class is already written. Your job is to write the bodies of the class's methods. Each method which you have to write has a comment that describes the method's job, and the body contains this comment:

```
// WRITE ME
```

So look for the `// WRITE ME` comments to find the code you need to update in your project.

On Blackboard, in addition to the IntLinkedList Java file which we've already been working with, you'll find a main file that "exercises" most of the class's methods, as well as the output that this file is expected to produce if your linked list class is written correctly. You can use this main file, in conjunction with the JUnit tests we've created in class, to help check your implementation for correctness. (I've also posted the JUnit tests on Blackboard, in case you don't have a complete version of those.)

I have two hints for you as you work on this project:
(1) Use the "picture" diagrams I'll give you in class, or make your own. It's much easier to work out insertion and deletion if you keep track of what's going on using a picture.
(2) Remember that prev and curr must always be one node apart. Whenever you modify curr, you must also modify prev!

--- SUBMISSION RULES - READ THESE ---

This program is due **Wednesday, November 20** at **11:59 PM**. To submit, send a FirstClass message to Mr. Stejskal's drop box, with the subject line "Java: Linked List." Attach only the file IntLinkedList.java to this message (I don't need your Main.java or

IntLinkedListTest.java; I already have these, after all). Please do not copy and paste the code into the body of this message, or attach any other files (especially not .class files).

Optional extra challenges
For an optional extra challenge, you may make any or all of these additions to the linked list class in this assignment:

Challenge #1: Doubly linked list
The linked list in this project is a singly linked list. You could also make a doubly linked list. (Remember that in a doubly linked list, each node contains *two* references: one to the next element in the list and one to the previous element.) A doubly linked list is slightly harder to write `insert` and `remove` methods for, because you have twice as many references to update. However, you can move backward (i.e. write a `movePrevious` method) in a doubly linked list.

To make the linked list class handier, change it to a doubly linked list. Add a `movePrevious` method and a `hasPrevious` method to the class. If you make the list doubly linked, you can actually remove the `prev` member variable from the list class (since each node will now contain its own previous node reference, you don't need to track the previous node any more).

Challenge #2: Sort the list
Add a `sort` method that sorts the list in ascending order. The strategy you use to sort the list is up to you, but insertion sort seems the obvious plan. (Remember: don't sort the list until the `sort` method is actually called.)

Challenge #3: Generic list
The `java.util.LinkedList` class is more useful than this linked list class, because it can hold any type of data rather than just `int`. That class uses Java generics to allow the class to hold data of any type. However, we can use generics in our own classes as well.

For this challenge, update the linked list class to be generic, so that you can declare a `MyLinkedList<Pirate>`, for example. (Of course, if you do this challenge, it no longer makes sense to call the class **`IntLinkedList`**.) If you're interested in trying this challenge, search for "Java generics" on Google and friends for more information.

(A word of warning: Combining challenges #2 and #3 isn't the best of ideas, because if your list is generic, it's no longer obvious how to compare two objects for sorting. If you have two Pirates, for example, which comes first? This is a problem we'll address later this year, but for now, I advise just avoiding the issue.)

Grading Rubric - Linked List Implementation

Correctness

- Program compiles without error, using main class on Blackboard (1 pt)_________
- Program runs without throwing an exception at any point (1 pt) _________
- All unit tests on Blackboard pass (1 pt) _________
- `getCount` method is correct (1 pt) _________
- `moveFirst` and `getCurrent` methods are correct (1 pt) _________
- `addFirst` method is correct (1 pt) _________
- `hasNext` and `moveNext` methods are correct (2 pts) _________
- `set`, `insert` and `remove` methods are correct (2 pts) _________

Clarity

- All code is indented consistently and for clarity (2 pts) _________
- Any local variables and extra added member variables used are suitably named (1 pt) _________
- Methods that navigate the list (`getCount`, `moveFirst`, `moveNext`, `getCurrent`, `hasNext`) are commented as needed (at least two comments among the methods) (2 pts) _________
- Methods that modify the list (`addFirst`, `set`, `insert`, `remove`) are commented (at least one suitable comment *per method*) (2 pts) _________

Efficiency

- **All** methods are O(1) running time (this means that you shouldn't use a loop in any method!) (2 pts) _________
- Program is not obviously inefficient (1 pt) _________

Total (max 20 points) _________

19-20 pts:	A	18 pts: B+
16-17 pts:	B	15 pts: C+
13-14 pts:	C	

Less than 13 pts is incomplete and must be redone

Bonus points (optional extra challenges)

- First optional addition (________________________________) _________
- Second optional addition (________________________________) _________

Any one successful optional extra challenge from the assignment description increases your grade by ½ letter grade (including raising A to A+).

AP Computer Science
Chapter 5 Programming Project

Shape Painter

Over the past few days, we've been using inheritance to build Shape Painter. You've seen several ways in which inheritance has helped us to build this Java program. Another advantage of inheritance is that it helps make programs more easily extensible. In this project, you'll extend Shape Painter to add some additional features.

On Blackboard, you'll find Mr. Stejskal's version of Shape Painter. As you should know, right now the program only lets the user draw ellipses. Your job in this assignment is to give the user some additional shape options beyond the ellipse.

As a reminder, Shape Painter currently consists of four classes:

- `ShapeFrame` is the program's main GUI, handling the program's menu and user interface. This `JFrame`-derived class is also responsible for handling mouse events (clicks and drags).
- `ShapeComponent` is a custom component class that keeps a list of the actual shapes that make up the drawing, and is responsible for displaying them.
- `Shape` is a generic class that serves as the base class for all the various shapes that the program will draw. This class contains two methods which you'll need to override in derived classes. Right now, though, there's only one derived class...
- `Ellipse` is a class derived from `Shape`, representing an ellipse.

Recall that there are two methods that you'll need to override in your `Shape`-derived classes. The first method is `draw`, which takes a `Graphics` object as a parameter, and draws the appropriate shape on the `Graphics` object. The second method is `hitTest`. This method takes a point (x and y coordinates) in as parameters, and returns a `boolean` that tells whether that point is within the shape in question. (This method is used by `ShapePanel` to determine which shape, if any, the user has clicked on.)

In this assignment, you have two requirements:

- Add *at least two* additional shape options for the user beyond ellipses. One of these shapes should be a rectangle. The second (or third, or fourth...) is up to you, though; you might consider shapes such as triangles, diamonds, pluses, or stars.
- For each of your new shapes, create a JUnit test that tests your shape's `hitTest` method. You can find Mr. Stejskal's unit test for ellipses on Blackboard.

For each shape you add, you'll need to:

- Create a new class derived from `Shape`.
- Implement the two abstract methods from the `Shape` class.
- Add a new menu item to the Shape menu in `ShapeFrame`, so that the user can choose your shape for drawing.
- Add code to `shapesMouseReleased` in `ShapeFrame` to actually create your new shape.

- Create (and run) your JUnit test to help verify your shape's correctness.

Don't make your shapes *too* complex. Remember that you have to write code to (a) draw your shape and (b) determine whether an arbitrary point is in your shape. (Depending on the shape you choose to draw, this may require applying some geometry.)

Again, I'll point out that the code in the project that deals with ellipses can be used as a model for your own code. Also, make sure to check out Sun's JDK documentation, especially for the `java.awt.Graphics` class, for information on how to draw your shapes.

--- SUBMISSION RULES - READ THESE ---

This program is due **Wednesday, December 11** at **11:59 PM**. To submit, send a FirstClass message to Mr. Stejskal's drop box, with the subject line "Java: Shape Painter." Compress your *complete project* into an archive (control-click the project folder within your NetBeansProjects folder, and choose the Compress option from the popup menu), and attach the compressed project archive (.zip file) to your FirstClass message.

Optional additional challenges
There are several optional additional challenges you might consider adding to this assignment. You're not limited to the ones I list here, either; feel free to make your own additions.

Shape delete option: The user might want to delete one shape, without deleting the entire drawing. Provide the user the option to click on a specific shape and delete only that shape from the drawing. (This will probably work a bit like the existing code that changes a shape's color.)

Change shape order: In this challenge, you give the user the ability to modify the order of shapes in the drawing (bring shapes to the "front" or "back" of the drawing stack). Again, how you implement this is up to you, but it will probably be something like changing shape color.

Let the user select custom colors: The program has a list of colors from which the user can select. These colors are nice, but the user might want a color that's not on the list. Since it's not feasible for your program to list *every* possible color on the menu (keep in mind that Java can recognize 2^{24} different colors!), the solution is to give the user a "Custom color" menu option. Want to try this challenge? Start by looking up `JColorChooser` (a class that can display a color-selection dialog box) in the online Java documentation.

Random drawing: Add an option that tells the program to create one or more random shapes. What will you randomize? The shape, location, and color are all good candidates.

Drag box: It's hard to tell what shape you're drawing as you're drawing it, because there's no display of how big the shape being drawn will be. In this challenge, you draw a rectangle that shows where the new shape will appear as the user drags the mouse. This way, the user can tell where the new shape will appear when he/she releases the button. If you want to take this challenge on, start by reading about *mouse motion listeners* (mouse drag event handlers) here:

http://docs.oracle.com/javase/tutorial/uiswing/events/mousemotionlistener.html

One word of warning: while certainly not impossible, this extra challenge is a bit harder than the other optional extra challenges suggested here.

Grading Rubric - Shape Painter

Correctness

- Program compiles without error (1 pt) __________
- Menu options for three shapes are available (1 pt) __________
- Rectangle class derives from `Shape` correctly (1 pt) __________
- GUI allows user to draw rectangles (1 pt) __________
- Rectangles respond to clicks (correct `hitTest`) (1 pt) __________
- Appropriate JUnit test for rectangles (1 pt) __________
- Other shape class derives from `Shape` correctly (1 pt) __________
- GUI allows user to draw other shapes (1 pt) __________
- Other shapes respond to clicks (correct `hitTest`) (1 pt) __________
- Appropriate JUnit test for other shape (1 pt) __________

Clarity

- All class members have appropriate names (2 pts) __________
- Code is formatted reasonably (2 pts) __________
- Classes, methods, and members are named appropriately (2 pts) __________
- Code is commented appropriately (2 pts) __________

Efficiency

- Program is not obviously inefficient (2 pts) __________

Total (max 20 points) __________

19-20 pts:	A
18 pts:	B+
16-17 pts:	B
15 pts:	C+
13-14 pts:	C

Less than 13 pts is incomplete and must be redone

Bonus points (optional extra challenges)

- First optional addition (__________________________________) __________
- Second optional addition (__________________________________) __________

Two or more bonus points (one difficult optional challenge, or two easy challenges) increase your grade by ½ letter grade (including raising A to A+).

AP Computer Science
Chapter 7 Programming Project #1

Address Book

In this assignment, your job is to create an "address book" program that can keep track of people's addresses. The program is a GUI program using Swing. I've written some of the code (mostly GUI code) for you; your job is to write the code that will let the user add addresses to the book and sort addresses.

When you run the program, you'll see a table with six columns: last name, first name, address, city, state, and ZIP. (For those who are curious, the table is done with a `JTable` control.) There are two "sample" names/addresses already in the address book; you can remove these. I added them to give you a sample of how to add people to the table.

The code I've given you contains the following classes:

- `Person` - This class represents one person in the address book; it has member variables that correspond to each of the six columns in the table.
- `AddressFrame` - This is the GUI for the program. Right now, it contains some setup code, and one very important member variable: `model`. `model` is a `TableModel` (like a `ListModel`, but for `JTables` instead). Actually, it's not just any old `TableModel`; it's a `PersonTableModel` (which is described in a moment).
- `PersonTableModel` - This is the class that keeps track of the `Persons` in the address table. It contains two methods that you'll need for the project:
    - `void addPerson(Person p)` - This method adds the `Person` you pass to it to the table model. Whenever you create a `Person` object, you must pass that object to this method to actually make the `Person` show up in the table.
    - `void sort(Comparator<Person> comp)` - This method sorts the people in the table, using the `Comparator` object you pass to it. (So it's similar to `Collections.sort`; in fact, it uses the `sort` method internally to do its job.)

  Note that the table will automatically redraw itself whenever you call either of these methods, so you don't need to call `repaint()` or anything like that to get your people to show up.
- `PersonSortingTest` - This JUnit test class will help you test your `Comparators`.

You have three tasks that you must do for this project:

1. Create a way to add new people to the address book. You can do this any way you find appropriate. Some examples: you might add some extra controls to the GUI to allow the user to type in a name and address; you might use `JOptionPane` to ask for name, address, etc; or you might create a dialog box of your own. Take a look at your Ninja GUI project and other GUI programs from last semester for ideas on how to do this.
2. Allow the user to sort people by any of the following: name, city, state, or ZIP code. (This will require you to create four `Comparators`.) This means you will

have to add some sort of GUI control (such as a button or menu item) for each sort option. (*Ideally*, we'd design the GUI so that clicking on a column's header caused the data to be sorted on that column, but it turns out that this is kind of a pain in the backside to do.) When sorting by name, sort by last name, and if two people have the same last name, sort them by first name. Keep in mind that I *will* test this, so you should too... You don't have to sort by address (it's usually a pretty meaningless thing to do), although you can if you really want to.

3. Modify the `PersonSortingTest` class to test your `Comparators`. I've added the framework for testing your sorting, but you'll need to actually use your `Comparators` to sort the array of `Person` objects that it creates, and then check that the objects are in the correct order. Two hints: (a) use the `Arrays.sort()` method, and (b) you should end up with four `@Test` functions when you're done.

--- SUBMISSION RULES - READ THESE ---
This program is due **Monday, January 27** at **11:59 PM**. To submit, send a message to Mr. Stejskal's FirstClass drop box, with the subject line "Java: Address Book". Compress your project folder into an archive, and attach that archive to the message.

Grading for this project will follow the grading rubric at the end of the assignment description.

Optional extra challenges
As usual, you may choose to enhance your assignment beyond the basic requirements with any of these (or other) *optional* challenges:

- Provide options for reverse-order sorting. You can use reverse-order comparators to do this fairly easily; look at the `Collections` class for a quick way to generate them.
- Allow the user to delete people from the address book, and/or to change information about people that are already in the book.
- Add an additional column or two to the address book. Phone number? Date you last heard from someone? Money this person owes you? There's lots of information you might want to remember about people...
- Save the data in the address book to a file and load saved data. (This would, obviously, be necessary if we want this program to be *useful*...)
- Creating a dialog box to ask for the information for a new address book entry looks nice, but it's also harder than the other available options. If you're interested in doing this, start by looking at the `JDialog` class online. Google can also provide examples of how to create and use your own dialog box, and Mr. Stejskal can also give some guidance if you choose to go this route.
- Make the column headers in the table clickable (so that clicking them sorts the contents by the data in that column). If you want to try this, you'll need to do some research. Start with a Google search for "JTable clickable column header".

Grading Rubric - Address Book

Correctness

- Program compiles without error (1 pt) ________
- UI allows user to create new entries (2 pts) ________
- New entries contain correct data in correct columns (1 pt) ________
- Four sorting options are available (1 pt) ________
- Each sorting option causes entries to be sorted (1 pt) ________
- City, state, and ZIP sorts are correct (1 pt) ________
- Name sort correctly sorts by last name, then by first (1 pt) ________
- JUnit tests correctly test all four sort orders, and all pass (2 pts) ________

Clarity

- GUI design makes sense and are usable (2 pts) ________
- GUI controls have appropriate variable names (1 pt) ________
- All comparator classes are appropriately named (2 pts) ________
- Comparator classes are in a sensible location (1 pt) ________
- Code is commented appropriately (min. 3 meaningful comments among your new code) (2 pts) ________
- Code formatting and indentation aids readability (1 pt) ________

Efficiency

- Program is not obviously inefficient (1 pt) ________

Total (max 20 points) ________

19-20 pts:	A
18 pts:	B+
16-17 pts:	B
15 pts:	C+
13-14 pts:	C

Less than 13 pts is incomplete and must be redone

Bonus points (optional extra challenges)

- First optional addition (________________________________) ________
- Second optional addition (________________________________) ________

Two or more bonus points increase your grade by ½ letter (including raising A to A+).

AP Computer Science
Chapter 8 Programming Project

GridWorld Cops and Robbers

As you've seen in this chapter, GridWorld allows you to make many different kinds of Actors, which can interact in interesting and sometimes complex ways. In this programming project, you'll create a cops-and-robbers chase scene, where the robbers try to steal money from banks, and the cops try to surround and capture the robbers.

For this project, you need to create *three* kinds of GridWorld actors:

- `Bank` should be derived from `Actor`. Banks don't move (anything else would be odd, yes?). In their `act` method, they accumulate money, and set their own color according to the amount of money in the bank. A Bank starts out with no money, and colored black. Each time the Bank acts, it adds $1 to its money, up to a maximum of $250. (Yes, these are small banks.) As the Bank's funds increase, it becomes more and more green - a Bank with the maximum $250 should be `Color(0, 250, 0)`. Banks also have a `rob` method, which removes all money from the bank (turning it black again).

- `Robber` should be derived from `Critter`. Robbers move randomly to an adjacent empty location (never to an occupied location). Robbers process actors as follows: if a robber is surrounded by *three or more* police officers (in the eight neighboring squares), then the robber is captured and removes itself from the grid. Otherwise, the robber robs all banks in squares adjacent to itself, and adds the stolen money to its stash. Robbers must also have a `getStash` method, which returns an integer telling how much money this robber has stolen.

- `PoliceOfficer` should also be derived from `Critter`. A police officer processes actors by looking at all the robbers (and only robbers) within *three squares* of itself in any direction. The officer chooses one of these robbers, randomly, as its "target" for this move. If there are no robbers within range, then the officer doesn't move and instead eats a donut. An officer with a robber within range has a 50% chance of moving one square *toward the target* (see later for help on how to do this) and a 50% chance of eating a donut. Officers can't enter occupied squares, so if the path to the robber is blocked by another Actor, the officer eats a donut. Whenever an officer moves, it turns blue. Whenever it eats a donut (in other words, whenever it doesn't move for any reason), it turns red.

In addition, for this project, you must create JUnit tests for `Bank` and for `Robber`. You should have a minimum of three test cases in your `Bank` test and four in your `Robber` test. (What should your test cases actually test? Think about the individual aspects of each actor's behavior. What behaviors should be tested for `Bank` and for `Robber`?)

I've already created unit tests for `PoliceOfficer`. You can use these unit tests to help test your `PoliceOfficer` class, as well as to help plan your `Bank` and `Robber` tests.

Finally, you also need to create a `Runner` class that demonstrates your bank, robber, and police officer classes in action. In your runner class, please create two banks, two robbers, and four police officers.

Remember to consider which methods you need to override in your derived classes. Neither `PoliceOfficer` nor `Robber` should override `act` (`Bank` should, since it will be derived directly from `Actor`). You may add additional member variables and/or methods as needed to your classes beyond the ones listed above, in order to help your classes do their jobs.

How do you figure out which direction to move a PoliceOfficer to chase a Robber? Remember GridWorld's `Location` class, and its handy methods? You might recall that one of these methods is called `getDirectionToward`. It takes one parameter (another `Location`), and computes the direction from `this` location to the other location.

So for example, when a police officer needs to know the direction to some other actor `a`, I could do this:

```
     int direction = getLocation().getDirectionToward(a.getLocation());
```

This line of code would return the direction from `this` toward `a`, rounded to the nearest 45°. You can then use this direction to plan the police officer's next move.

Avoid this common bug: Although a robber might determine in `processActors` that it should remove itself from the grid, you *cannot* actually remove it there. (Why not? The `act` method will continue to call the other `Critter` methods, which will cause a `NullPointerException` if the robber has already removed itself from the grid.) To avoid this bug, use a (member) variable to remember that the robber should remove itself, but don't actually use `removeSelfFromGrid()` on the robber until its `makeMove` method.

--- SUBMISSION RULES - READ THESE ---
This program is due **Wednesday, March 5** at **11:59 PM**. To submit, send a FirstClass message to Mr. Stejskal's drop box, with the subject line "Java: Chase". Compress your project folder into a .zip file, and attach that file to the message.

Optional Extra Challenges
As usual, there are some optional extra challenges for this project:

- *Actor graphics*. Include images for the `Bank`, `Robber`, and `PoliceOfficer` classes; otherwise, they'll look like generic Actors and Critters. (This challenge, by itself, is not enough to earn an A+; it's more of a for-fun idea.)

- *Smart robbers*. Rather than make a robber's movement random, have robbers attempt to move (a) away from police officers, (b) toward banks with lots of money, or (c) both. Note that a robber shouldn't try to move toward banks that are low on money, because this will usually cause a robber to just hang out around one bank indefinitely.

- *Spawning robbers*. A robber with a sufficiently large stash "splits" into two new robbers (which start out with no stash). How much is "sufficiently large"? Let the user choose: make it a static variable (shared among all robbers), and include a method in the Robber class to set this value. (Make sure to make the "split value" start out as something reasonable - in particular, 0 dollars is a bad idea, as you'll be overrun with robbers in short order.)

- *Bank guards*. A bank guard is an actor that walks in a circle around a bank (its movement is like `BoxBug`). The guard helps to defend the bank from robberies because robbers can't go through the guard to get to the bank. The bank guard will "eat" and capture robbers it happens to walk onto, but it won't capture any other actor (and it won't go out of its way to capture robbers, either). If there's a police officer or some other non-robber actor in the way of the bank guard's loop around the bank, then it will stand and wait for the actor to move. For this extra challenge, create a bank guard class, and place guards around one or more banks in the Runner class.

Grading Rubric - GridWorld Chase

Correctness

- Program compiles and runs without error as submitted (1 pt) ________
- Bank class is present and banks are created (1 pt) ________
- Banks accumulate money in act and change color as they do (1 pt) ________
- Robber class is present and robbers are created (1 pt) ________
- Robbers move correctly (1 pt) ________
- Robbers steal money from banks and add it to their stash (1 pt) ________
- PoliceOfficer class is present and officers are created (1 pt) ________
- Officers move correctly (1 pt) ________
- Officers turn blue whenever they move and red whenever they don't (1 pt) ________
- Robbers are captured when surrounded by 3 or more officers (1 pt) ________

Clarity

- Local and member variables are appropriately named (1 pt) ________
- Classes are arranged in a logical order (1 pt) ________
- Bank JUnit tests are present and cover expected behaviors (2 pts) ________
- Robber JUnit tests are present and cover expected behaviors (2 pts) ________
- Code is commented appropriately (min. 3 meaningful comments per class (2 pts) ________
- Code formatting and indentation aids readability (1 pt) ________

Efficiency

- Program is not obviously inefficient (1 pt) ________

Total (max 20 points) ________

19-20 pts:	A	18 pts:	B+
16-17 pts:	B	15 pts:	C+
13-14 pts:	C		

Less than 13 pts is incomplete and must be redone

Bonus points (optional extra challenges)

- First optional addition (________________________________) ________
- Second optional addition (________________________________) ________

Two or more bonus points increase your grade by ½ letter (including raising A to A+).

Appendix F - Sample Student-Created JUnit Tests

This section contains sample student-created JUnit tests for projects where students were required to write JUnit tests. (See Appendix E for descriptions of those projects.) In the interest of space, not all students' submissions have been reproduced. Different students' code is shown for different projects (thus "student 1" for Shape Painter does not correspond to "student 1" for Address Book).

Sample tests submitted for Shape Painter

Student 1 submission

```
public class RectangleTest
{
    public RectangleTest()
    {
    }

    @Test
    public void testRectangleColor()
    {
        Rectangle r = new Rectangle(1, 2, 3, 4);
        r.setColor(Color.PINK);
        assertEquals(Color.PINK, r.getColor());
    }

    @Test
    public void testRectangleHitTest()
    {
        Rectangle r = new Rectangle(100, 100, 300, 200);

        assertTrue(r.hitTest(200, 150));
        assertTrue(r.hitTest(200, 199));
        assertTrue(r.hitTest(250, 125));
        assertTrue(r.hitTest(150, 150));
        assertFalse(r.hitTest(-1, -1));
        assertFalse(r.hitTest(99, 99));
        assertFalse(r.hitTest(301, 201));
        assertFalse(r.hitTest(100, 201));
    }
}

public class DiamondTest
{
    public DiamondTest()
    {
    }
    @Test
    public void testDiamondColor()
    {
        ShapeDiamond d = new ShapeDiamond(1, 1, 10, 10);
        d.setColor(Color.PINK);
        assertEquals(Color.PINK, d.getColor());
    }

    @Test
    public void testDiamondHitTest()
    {
        ShapeDiamond d = new ShapeDiamond(100, 100, 200, 200);

        assertTrue(d.hitTest(150, 150));
        assertTrue(d.hitTest(150, 101));
        assertTrue(d.hitTest(101, 150));
        assertTrue(d.hitTest(199, 150));
        assertFalse(d.hitTest(175, 190));
        assertFalse(d.hitTest(125, 190));
        assertFalse(d.hitTest(110, 110));
        assertFalse(d.hitTest(190, 110));
    }
}
```

Student 2 submission

```
public class TriangleTest
{

    public TriangleTest()
    {
    }

    @Test
    public void testTriangleHitTest()
    {
        int x1 = 100;
        int y1 = 100;
        int x2 = 200;
        int y2 = 200;

        int[] Xs = new int[3];
        int[] Ys = new int[3];

        //top middle point
        Xs[0] = x1 + ((x2 - x1) / 2);
        Ys[0] = y1;

        //bottom left point
        Xs[1] = x1;
        Ys[1] = y2;

        //botom right point
```

```
                Xs[2] = x2;
                Ys[2] = y2;

                Polygon p = new Polygon(Xs, Ys, 3);

                assertTrue(p.contains(101, 199));
                assertTrue(p.contains(199, 199));
                assertFalse(p.contains(151, 100));
                assertFalse(p.contains(99, 199));
                assertFalse(p.contains(201, 199));
            }
        }

        public class RectangleTest
        {
            public RectangleTest()
            {
            }

            @Test
            public void testRectangleColor()
            {
                Rectangle e = new Rectangle(1, 2, 3, 4);
                e.setColor(Color.PINK);
                assertEquals(Color.PINK, e.getColor());
            }

            @Test
            public void testRectangleHitTest()
            {
                Rectangle e = new Rectangle(100, 100, 200, 200);

                //test corners mostly
                assertTrue(e.hitTest(101, 101));
                assertTrue(e.hitTest(101, 199));
                assertTrue(e.hitTest(199, 101));
                assertTrue(e.hitTest(199, 199));
                assertFalse(e.hitTest(-1, -1));
                assertFalse(e.hitTest(99, 99));
                assertFalse(e.hitTest(295, 105));
                assertFalse(e.hitTest(200, 99));
            }
        }
```

Sample tests submitted for Address Book

Student 1 submission

```
        public class PersonSortingTest
        {
            public PersonSortingTest()
            {
            }
```

```
// A method marked with @Before is called before _each_ @Test
// method.  In this case, we use this method to create some
// Person objects and put them in a known unsorted order (so
// that we can check that they have been correctly sorted by
// each comparator).
@Before
public void scramblePersons()
{
    alicia = new Person("Alicia", "Jones",
            "123 Anywhere", "Austin", "TX", 23456);
    bob = new Person("Bob", "Dole", "345 Somewhere",
            "Russell", "KS", 67890);
    cindy = new Person("Cindy", "Smith", "234 Nowhere",
            "Augusta", "ME", 11111);
    don = new Person("Don", "Giovanni", "555 Everywhere",
            "Sacramento", "CA", 99876);
    elizabeth = new Person("Elizabeth", "Dole",
            "543 Elsewhere", "Omaha", "NE", 68124);
    fred = new Person("Fred", "Dole", "-1 Icywhere",
            "Anchorage", "AK", 54545);
    greg = new Person("Greg", "Louganis",
            "999 Waterwhere", "Honolulu", "HI", 43210);

    personArray = new Person[7];

    personArray[0] = greg;
    personArray[1] = fred;
    personArray[2] = elizabeth;
    personArray[3] = don;
    personArray[4] = cindy;
    personArray[5] = bob;
    personArray[6] = alicia;
}

@Test
public void testSortByName()
{
    // Sort the array by name using the PersonNameComparator
    Arrays.sort(personArray, 0, 7,
            new PersonNameComparator());

    assertArrayEquals(
            new Person[]
            {
                bob, elizabeth, fred, don,
                    alicia, greg, cindy
            },
            personArray);
}

@Test
public void testSortByCity()
{
    // Re-scramble the person array
    scramblePersons();
```

```
        // Sort the array by city using the PersonCityComparator
        Arrays.sort(personArray, 0, 7,
                  new PersonCityComparator());

        assertArrayEquals(
                  new Person[]
                  {
                       fred, cindy, alicia, greg,
                            elizabeth, bob, don
                  },
                  personArray);
    }

    @Test
    public void testSortByState()
    {
        // Re-scramble the person array
        scramblePersons();

        // Sort the array by state using PersonStateComparator
        Arrays.sort(personArray, 0, 7,
                  new PersonStateComparator());

        assertArrayEquals(
                  new Person[]
                  {
                       fred, don, greg, bob,
                            cindy, elizabeth, alicia
                  },
                  personArray);
    }

    @Test
    public void testSortByZip()
    {
        // Re-scramble the person array
        scramblePersons();

        // Sort the array by zip code using PersonZipComparator
        Arrays.sort(personArray, 0, 7,
                  new PersonZipComparator());

        assertArrayEquals(
                  new Person[]
                  {
                       cindy, alicia, greg, fred,
                            bob, elizabeth, don
                  },
                  personArray);
    }

    private Person alicia;
    private Person bob;
    private Person cindy;
    private Person don;
    private Person elizabeth;
    private Person fred;
    private Person greg;
    private Person[] personArray;
}
```

Student 2 submission

```
public class AddressBookTest
{
    public AddressBookTest()
    {

    }

    /*
     * To test Creating of Person object.
     */
    @Test
    public void testCreatePerson()
    {
        Person p = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        assertEquals("S*****", p.getFirstName());
        assertEquals("N*****", p.getLastName());
        assertEquals("1234 166th Ave", p.getAddress());
        assertEquals("Omaha", p.getCity());
        assertEquals("NE", p.getState());
        assertEquals(68116, p.getZip());
    }

    /*
     * To test NumerFormatException error.
     */
    @Test(expected = NumberFormatException.class)
    public void testNullPerson()
    {
        Person p = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", Integer.parseInt(""), 0);
    }

    /*
     * To test Name comparator class.
     */
    @Test
    public void testPersonNameComparator()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                 "2234 166th Ave", "Bmaha",
                 "BE", 68115, 0);
        Comparator comp = new Person.NameComparator();
        //System.out.println(comp.compare(p1, p2));
        assertEquals(true, comp.compare(p1, p2) > 0);
        assertEquals(true, comp.compare(p1, p1) == 0);
        assertEquals(true, comp.compare(p2, p1) < 0);
    }

    /*
     * To test City comparator class.
     */
    @Test
    public void testPersonCityComparator()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                 "2234 166th Ave", "Bmaha",
                 "BE", 68115, 0);
        Comparator comp = new Person.CityComparator();
        //System.out.println(comp.compare(p1, p2));
        assertEquals(true, comp.compare(p1, p2) > 0);
        assertEquals(true, comp.compare(p1, p1) == 0);
        assertEquals(true, comp.compare(p2, p1) < 0);
    }

    /*
     * To test State comparator class.
     */
    @Test
    public void testPersonStateComparator()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                 "2234 166th Ave", "Bmaha",
                 "BE", 68115, 0);
        Comparator comp = new Person.StateComparator();
        //System.out.println(comp.compare(p1, p2));
        assertEquals(true, comp.compare(p1, p2) > 0);
        assertEquals(true, comp.compare(p1, p1) == 0);
        assertEquals(true, comp.compare(p2, p1) < 0);
    }

    /*
     * To test Zip comparator class.
     */
    @Test
    public void testPersonZipComparator()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Comparator comp = new Person.ZipComparator();
        //System.out.println(comp.compare(p1, p2));
        assertEquals(true, comp.compare(p1, p2) > 0);
        assertEquals(true, comp.compare(p1, p1) == 0);
        assertEquals(true, comp.compare(p2, p1) < 0);
    }

    /*
     * To test adding person objects to Address Table.
     */
    @Test
    public void testPersonTableModel()
    {
        Person p1 = new Person("S*****", "N*****",
                "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
    }

    /*
     * To test Sort by LastName, FirstName of Address Table.
     */
    @Test
    public void testPersonTableModelNameSort()
    {
        Person p1 = new Person("S*****", "N*****",
                "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.NameComparator(), false);
        assertEquals("B*****", model.getPerson(0).
                getFirstName());
        assertEquals("B*****", model.getPerson(0).
                getLastName());
        assertEquals("V*****", model.getPerson(3).
                getFirstName());
        assertEquals("V*****", model.getPerson(3).
                getLastName());
    }

    /*
     * To test Sort by LastName, FirstName of Address Table in
reverse order.
     */
    @Test
    public void testPersonTableModelNameRevSort()
    {
        Person p1 = new Person("S*****", "N*****",
                "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.NameComparator(), true);
        assertEquals("V*****", model.getPerson(0).
                getFirstName());
        assertEquals("V*****", model.getPerson(0).
                getLastName());
        assertEquals("B*****", model.getPerson(3).
                getFirstName());
        assertEquals("B*****", model.getPerson(3).
                getLastName());
    }

    /*
     * To test Sort by City of Address Table.
```

```
 */
@Test
public void testPersonTableModelCitySort()
{
    Person p1 = new Person("S*****", "N*****",
            "1234 166th Ave", "Omaha",
            "NE", 68116, 0);
    Person p2 = new Person("B*****", "B*****",
            "2234 166th Ave", "Bmaha",
            "BE", 68115, 0);
    Person p3 = new Person("V*****", "V*****",
            "3234 166th Ave", "Vmaha",
            "VE", 68113, 0);
    Person p4 = new Person("P*****", "P*****",
            "4234 166th Ave", "Pmaha",
            "PE", 68111, 0);
    PersonTableModel model = new PersonTableModel();
    model.addPerson(p1);
    model.addPerson(p2);
    model.addPerson(p3);
    model.addPerson(p4);
    assertEquals(4, model.getRowCount());
    model.sort(new Person.CityComparator(), false);
    assertEquals("Bmaha", model.getPerson(0).getCity());
    assertEquals("Omaha", model.getPerson(1).getCity());
    assertEquals("Pmaha", model.getPerson(2).getCity());
    assertEquals("Vmaha", model.getPerson(3).getCity());
}

/*
 * To test Sort by City of Address Table in reverse order.
 */
@Test
public void testPersonTableModelCityRevSort()
{
    Person p1 = new Person("S*****", "N*****",
            "1234 166th Ave", "Omaha",
            "NE", 68116, 0);
    Person p2 = new Person("B*****", "B*****",
            "2234 166th Ave", "Bmaha",
            "BE", 68115, 0);
    Person p3 = new Person("V*****", "V*****",
            "3234 166th Ave", "Vmaha",
            "VE", 68113, 0);
    Person p4 = new Person("P*****", "P*****",
            "4234 166th Ave", "Pmaha",
            "PE", 68111, 0);
    PersonTableModel model = new PersonTableModel();
    model.addPerson(p1);
    model.addPerson(p2);
    model.addPerson(p3);
    model.addPerson(p4);
    assertEquals(4, model.getRowCount());
    model.sort(new Person.NameComparator(), true);
    assertEquals("Vmaha", model.getPerson(0).getCity());
```

```
        assertEquals("Pmaha", model.getPerson(1).getCity());
        assertEquals("Omaha", model.getPerson(2).getCity());
        assertEquals("Bmaha", model.getPerson(3).getCity());
    }

    /*
     * To test Sort by Zip of Address Table.
     */
    @Test
    public void testPersonTableModelZipSort()
    {
        Person p1 = new Person("S*****", "N*****",
                "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.ZipComparator(), false);
        assertEquals(68111, model.getPerson(0).getZip());
        assertEquals(68113, model.getPerson(1).getZip());
        assertEquals(68115, model.getPerson(2).getZip());
        assertEquals(68116, model.getPerson(3).getZip());
    }

    /*
     * To test Sort by Zip of Address Table in reverse order.
     */
    @Test
    public void testPersonTableModelZipRevSort()
    {
        Person p1 = new Person("S*****", "N*****",
                "1234 166th Ave", "Omaha",
                "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                "2234 166th Ave", "Bmaha",
                "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.ZipComparator(), true);
        assertEquals(68116, model.getPerson(0).getZip());
        assertEquals(68115, model.getPerson(1).getZip());
        assertEquals(68113, model.getPerson(2).getZip());
        assertEquals(68111, model.getPerson(3).getZip());
    }

    /*
     * To test Sort by State of Address Table.
     */
    @Test
    public void testPersonTableModelStateSort()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                 "2234 166th Ave", "Bmaha",
                 "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                 "3234 166th Ave", "Vmaha",
                 "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                 "4234 166th Ave", "Pmaha",
                 "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.StateComparator(), false);
        assertEquals("BE", model.getPerson(0).getState());
        assertEquals("NE", model.getPerson(1).getState());
        assertEquals("PE", model.getPerson(2).getState());
        assertEquals("VE", model.getPerson(3).getState());
    }

    /*
     * To test Sort by State of Address Table in reverse order.
     */
    @Test
    public void testPersonTableModelStateRevSort()
    {
        Person p1 = new Person("S*****", "N*****",
                 "1234 166th Ave", "Omaha",
                 "NE", 68116, 0);
        Person p2 = new Person("B*****", "B*****",
                 "2234 166th Ave", "Bmaha",
                 "BE", 68115, 0);
        Person p3 = new Person("V*****", "V*****",
                "3234 166th Ave", "Vmaha",
                "VE", 68113, 0);
        Person p4 = new Person("P*****", "P*****",
                "4234 166th Ave", "Pmaha",
                "PE", 68111, 0);
        PersonTableModel model = new PersonTableModel();
        model.addPerson(p1);
        model.addPerson(p2);
        model.addPerson(p3);
        model.addPerson(p4);
        assertEquals(4, model.getRowCount());
        model.sort(new Person.StateComparator(), true);
        assertEquals("VE", model.getPerson(0).getState());
        assertEquals("PE", model.getPerson(1).getState());
        assertEquals("NE", model.getPerson(2).getState());
        assertEquals("BE", model.getPerson(3).getState());
    }
}
```

Sample tests submitted for GridWorld Cops and Robbers

Student 1 submission

```
public class BankTest
{
    private ActorWorld world;
    private Bank bank;

    public BankTest()
    {

    }

    @Before
    public void setUp()
    {
        // Put a bank in the middle of the grid
        world = new ActorWorld();
        bank = new Bank();
        world.add(new Location(5, 5), bank);
    }

    //Tests the bank's money after it is added to when it is
    //empty and full
    @Test
    public void testAct()
    {
        bank.act();
        assertEquals(bank.getMoney(), 1);

        bank = new Bank(255);
        assertEquals(bank.getMoney(), 255);
```

```
            bank.act();
            assertEquals(bank.getMoney(), 255);
        }

        //Tests if the money equals 0 after the bank is robbed
        @Test
        public void testRob()
        {
            bank.rob();
            assertEquals(bank.getMoney(), 0);

            bank = new Bank(89);
            bank.rob();
            assertEquals(bank.getMoney(), 0);
        }
    }

    public class RobberTest
    {
        private ActorWorld world;
        private Robber robber;

        public RobberTest()
        {

        }

        //Code before the tests actually start
        @Before
        public void setUp()
        {
            world = new ActorWorld();
            robber = new Robber();
            world.add(new Location(5, 5), robber);
        }

        //Tests if the robber has all the moeny in the bank and the
        //bank has no money after the robber robs the bank
        @Test
        public void testRobBank()
        {
            Bank bank = new Bank(255);
            world.add(new Location(4, 5), bank);

            world.step();
            assertEquals(bank.getMoney(), 0);
            assertEquals(robber.getMoney(), 255);
        }

        //Tests if the robber will remove itself after it is
        //surrounded by 3 robbers and if the bank remains full if the
        //robber is to remove itself
        @Test
        public void testGetCaught()
        {
```

```
        PoliceOfficer officer1 = new PoliceOfficer();
        world.add(new Location(4, 5), officer1);
        PoliceOfficer officer2 = new PoliceOfficer();
        world.add(new Location(5, 4), officer2);
        PoliceOfficer officer3 = new PoliceOfficer();
        world.add(new Location(6, 5), officer3);
        Bank bank = new Bank(255);
        world.add(new Location(5, 6), bank);

        world.step();
        for (int x = 0; x < 10; x++)
        {
            for (int y = 0; y < 10; y++)
            {
                assertFalse(world.getGrid().get(
                    new Location(y, x)) instanceof Robber);
            }
        }

        assertEquals(bank.getMoney(), 255);
    }
}
```

Student 2 submission

```
public class BankTest
{
    public BankTest()
    {
    }

    private ActorWorld world;
    private Bank bank;

    @Before
    public void setUp()
    {
        // Put a police officer in the middle of the grid
        world = new ActorWorld();
        bank = new Bank();
        world.add(new Location(5, 5), bank);
    }

    @Test
    public void testAct()
    {

        //Tests for 0 money at beginning
        assertEquals(0, bank.getMoney());
        bank.act();
        assertEquals(1, bank.getMoney());
        for (int i = 0; i < 50; i++)
        {
            bank.act();
        }
        assertEquals(51, bank.getMoney());
    }

    @Test
    public void testRob()
    {
        for (int i = 0; i < 100; i++)
        {
            bank.act();
        }

        assertEquals(bank.rob(), 100);

    }

    public void testColorChange()
    {
        assertEquals(Color.BLACK, bank.getColor());

        bank.act();
        bank.act();

        assertEquals(new Color(0, 2, 0), bank.getColor());

        for (int i = 0; i < 500; i++)
            bank.act();

        assertEquals(new Color(0, 250, 0), bank.getColor());

    }
}

public class RobberTest
{
    private ActorWorld world;
    private Robber robber;

    public RobberTest()
    {
    }

    @Before
    public void setUp()
    {
        world = new ActorWorld();
        robber = new Robber();
        System.out.println("Setting up robber");
        world.add(new Location(5, 5), robber);

    }

    @Test
    public void testGetCaught()
    {
        world.add(new Location(4, 5), new PoliceOfficer());
        world.add(new Location(4, 4), new PoliceOfficer());
        world.add(new Location(5, 6), new PoliceOfficer());
        world.step();
        assertEquals(null, robber.getLocation());
    }

    @Test
    public void testIfRobsBank()
    {
        Bank bank = new Bank();
        world.add(new Location(4, 5), bank);
        for (int i = 0; i < 10; i++)
        {
            bank.act();
        }
        System.out.println("Testing robber rob");
        robber.act();
        assertEquals(10, robber.getStash());
    }

    @Test
    public void testRockInWay()
    {
       //surrounds the robber with rocks and makes
        //sure that he does not move.
        world.add(new Location(4, 5), new Rock());
        world.add(new Location(4, 4), new Rock());
        world.add(new Location(5, 4), new Rock());
        world.add(new Location(6, 4), new Rock());
        world.add(new Location(4, 6), new Rock());
        world.add(new Location(6, 6), new Rock());
        world.add(new Location(5, 6), new Rock());
        world.add(new Location(6, 5), new Rock());

        for (int i = 0; i < 50; i++)
            world.step();

        assertEquals(new Location(5, 5), robber.
                getLocation());
    }

    public void testMovesEveryAct()
    {

        for (int i = 0; i < 50; i++)
        {
            Location l = robber.getLocation();
            world.step();
            assertFalse(robber.getLocation() == l);
        }
    }
}
```