IMPROVING WEBIDE THROUGH DELIGHTFUL DESIGN AND GAMIFICATION

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Michael Hilton

March 2013

COMMITTEE MEMBERSHIP

TITLE:	Improving WebIDE through Delightful Design and Gamification
AUTHOR:	Michael Hilton
DATE SUBMITTED:	March 2013
COMMITTEE CHAIR:	David Janzen, Ph.D. Computer Science Department, Cal Poly, San Luis Obispo
COMMITTEE MEMBER:	John Clements, Ph.D. Computer Science Department, Cal Poly, San Luis Obispo
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Computer Science Department, Cal Poly, San Luis Obispo

Abstract

Improving WebIDE through Delightful Design and Gamification

Michael Hilton

WebIDE is a web-based online learning environment. WebIDE has been used successfully to teach CS0 and CS1 students Java and C concepts and software engineering best practices, specifically Test Driven Development. Previous WebIDE development has concentrated on developing functionality. The main goal of this effort is to improve two non-functional aspects of WebIDE. The first is to design a more delightful user interface. The second is to add a scoring mechanism that encourages students to develop best practices. The scoring mechanism rewards students who answer the question correctly on the first attempt, discouraging them from spamming the answer button. Our objective is to motivate the students to think before answering. The innovations are evaluated through a semi-controlled experiment that was conducted during the Fall quarter of 2012 at Cal Poly.

Acknowledgements

I would like to thank my committee members, Dr. John Clements and Dr. Franz Kurfess, for their support and input on the process. They both provided critical feedback that I feel helped me significantly improve on the weakest parts of my thesis.

Many thanks to Olga Dekhtyar (Cal Poly Department of Statistics) for her help with the assessment of the results of the experiment. Her help was invaluable in making sense of the data from the experiments.

I would like to thank Vanessa Forney and Lucas David for their help over the summer while I was developing the new interface for WebIDE. They provided a lot of great ideas as far as the initial layout, and provided a great team atmosphere.

The help that my neighbor, John Kelly, provided in proofreading was invaluable, and helped me tighten up the prose, and significantly improve the readability of this thesis.

I don't how to thank Dr. David Janzen for all the help he has given me throughout this process. Thanks for taking me on as a student even though it was your sabbatical. Thanks for allowing me to perform an experiment in the Fall 2012 csc123 class. You have truly been a inspiration to me!

My family deserves the most credit of all. Thanks to my lovely wife, Ann, and all her support, even when it looked like I was just surfing the internet. To my children, Elena and David, you both have been the best kids a dad could hope for. Thanks so much for providing our family with endless entertainment, as well as motivation for me to wrap things up in time to make it home for dinner.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

WebIDE was initially conceived as a tool to help students learn programming by presenting lessons in the easy to use, familiar environment of a web browser. There are many challenges facing first-year students and they often struggle with the tools that they need to use in order to learn basic programming concepts. WebIDE was developed to help students focus on the concepts that they are trying to learn, instead of struggling to learn how to use the tools. WebIDE accomplishes this goal is by letting students write code in the browser and then click the "run" button which sends the code to the server, where it is compiled and executed. The result is then returned and displayed in the browser.

In addition to providing students an easy way to develop code, WebIDE was developed to be a pedagogical tool to help students learn how to program in a Test Driven Learning (TDL) manner. While TDL has been shown to be beneficial to beginning students, it has proven to be difficult to convince them to adopt TDL behaviors.[16] WebIDE was designed to allow instructors to present lessons broken down into small steps, which can be locked so that they must be performed in a specific order. This functionality enables instructors to ensure

that the tests are being written before the code, if that is a constraint they wish to enforce.

The initial development of WebIDE was focused on functionality, and it demonstrated positive results.[8] However, like many software development projects, most of the initial development was focused on functional requirements, resulting in a user interface that was less then polished. The UI improvements described in this work are intended to increase the delightfulness of WebIDE for the students using it, and to encourage more professors to adopt WebIDE as a part of their courses.

In addition to increasing the delightfulness of the user interface, I added game mechanics to WebIDE, in an effort to encourage good practices among the students. The game mechanics consisted of a scoring mechanism that encouraged students to answer the question correctly on the first attempt. The reason for using this scoring mechanic is to help students not only write tests, but to put some thought into their answers before they submit them. The goal of this approach is to help students get into the mindset of Test Driven Development, where the first step is to stop and think about where you are going, as opposed to starting to write code without a clear goal in mind.

An experiment was conducted during the Fall 2012 quarter with two sections of Introduction to Computing - Mobile. Both sections were taught by Dr. David Janzen. The experimental section was chosen at random at the beginning of the experiment.

Much of the design and construction of the experimental versions of WebIDE used in this thesis was completed during the summer preceding the experiment. During that time I was able to participate in a mentoring program with two undergraduate students at Cal Poly. They were a large help in implementing many of the features described in this thesis, especially in the user interface design.

Chapter 2

Related Work

The related work for this effort spans a wide variety of topics. The various lines of research which intersect with the effort of WebIDE include Human Computer Interface, E-learning interface evaluation, Delight, and Gamification.

2.0.1 Human Computer Interface

In 1986, *The Psychology of Human-Computer Interaction*[2] was published which was the landmark book dealing with Human-Computer Interaction (HCI). The authors' purpose was to develop a scientific psychology that would help arrange interfaces that are easy, efficient, error-free and even enjoyable.

Another important work was a book written by Donald Norman titled *The Design of Everyday Things.*[23] His intended audience was not specifically software designers, but designers in general. His focus is on designing with the user in mind, so that the designer is thinking about how the user will interact with the object. Norman states that well-designed objects are easy to interpret and understand, while poorly designed ones can be difficult and even frustrating to use.

In order to help designers, Norman proposes seven principles for transforming difficult tasks into simple ones.

1. Use both knowledge in the world and knowledge in the head.
2. Simplify the structure of tasks.
3. Make things visible: bridge the gulfs of Execution and Evaluation.
4. Get the mappings right.
5. Exploit the power of constraints, both natural and artificial.
6. Design for error.
7. When all else fails, standardize.

These design principles are called User Centered Design (UCD). Having read Norman's book before beginning this project, I was able to incorporate UCD principles into the design of the new WebIDE interface. The concepts of UCD have been applied to many different areas, including games,[24] websites,[3] real-time battlefield visualization virtual environments,[14] mobile application development, [18] intelligent service robots,[11] digital libraries,[29] and e-learning.[1]

2.0.2 E-learning interface evaluation

With the rise of e-learning there has also been an increase in the study of e-learning interfaces. Zaharias [30] describes evaluating the usability of e-learning applications as a nontrivial task. However, he also points out that the usability of e-learning designs is directly related to their pedagogical value. It is the goal

of this project to increase the pedagogical value of WebIDE by increasing its usability.

Costabile et al. [4] argued that in addition to User-Centered Design (UCD) methods, Learner-Centered Design (LCD) methods are needed in order to develop new learning domains that are accessible and educationally effective. They propose that we do not need interfaces that support "doing tasks," but interfaces that support "learning while doing tasks." They argue that the usability of an e-learning application can significantly affect learning; therefore usability should be one of the main challenges of e-learning application developers. While WebIDE had a good functional core, little effort had been made to develop the usability of the system. Our goal was to increase the usability of WebIDE, in order to improve the ability of students to learn.

2.0.3 Delight

Software Design Manifesto

In 1991, Mitchell Kapor published a Software Design Manifesto.[19] He compared software design to the manner in which architects and construction engineers work together in the construction of a building, even though the architect has the overall responsibility for the building.[19] He posits that in a similar manner, software engineers and software designers should work together as well. He also discussed the notion of well-designed buildings that was formulated by the Roman architecture critic Vitruvius, which consists of firmness, commodity and delight. Kapor claimed that these same criteria also apply to well designed software.

Customer Satisfaction

The following year, Peter Denning wrote an article titled "What is Software Quality"[6] in which he proposes that the eponymous question should be reframed as "How do we satisfy the customers of our software?" He offers three levels at which a customer can declare satisfaction.

1. All basic promises were fulfilled.
2. No negative consequences were produced.
3. The customer is delighted.

Denning states that very few software systems have actually produced genuine delight. A few examples he gives are the Apple MacIntosh, Lotus 1-2-3, and the Quicken accounting system. He specifically describes that he chose these examples of delight because the users found they could complete more work in a faster manner with these systems than without them. He specifically mentions these software applications as exceeding the expectations of customers. As this article is targeted at businesses, Denning claims that one of the significant benefits of delighting the customer is the cultivation of fiercely loyal customers.

Usability Engineering and Agile Software Development

Sohabib et al.[28] wrote about the integration of usability engineering and agile software development. The authors claim that Extreme Programming delivers high quality software efficiently, but warns that the resulting software might fail to delight the customer, as usability engineering methods are not integrated into agile methods when developing UI intensive systems. In order to better delight users, they suggest several ways to augment Extreme Programming methods to build software that is more delightful.[22, 21, 25]

Other Terms for Delight

Karen Holtzblatt published an article "What Makes Things Cool? Intentional Design for Innovation"[15] which does not use the term delight, but covers similar concepts. The main concept that the author explores is what she titles the joy wheel and the joy triangle. It describes how creators can create joy in their end users. The joy wheel consists of core life motivators, which are Accomplish, Connection, Identity and Sensation. These are the motivations in the user that the designer should try to satisfy. How this plays out when the product is actually in use is described by the joy triangle. The joy triangle defines how the user will find joy while using the product. The joy triangle is compromised of the Hassle Factor, Direct into Action, and the Delta. Direct into Action is how quickly the user can use the product to accomplish their needs. The goal for Direct into Action is to let the user start using the system and accomplish their goal as quickly and directly as possible. The Hassle Factor defines the opposite parameter, the amount of hassle the user must deal with in order to use the product. One of my main design goals in reworking the interface for WebIDE was to minimize the Hassle Factor for the end user.

Hartmann, Sutcliffe, De Angeli [12] studied the link between aesthetics and usability. They concluded that positively perceived aesthetics creates a halo effect that causes users to experience higher satisfaction and system acceptance. However, they also found that in some cases increased aesthetics could lead to a decrease in usability, measured by both objective and subjective metrics. It is a challenge to balance good aesthetics with good usability, and it is important to not overemphasize one to the detriment of another.

2.0.4 Web-based Programming Learning Environments

There are a variety of Web-based learning environments to help students learn how to program. While none of them have a focus on Test Driven Learning similar to WebIDE, there are some areas in which they are aligned very nicely with WebIDE.

Codecademy.com

Codecademy[1] is an online programming learning tool. Codecademy offers users the option to learn JavaScript, HTML/CSS, Python, and Ruby. Unlike WebIDE, it does not offer students the option to program in Java, nor does it focus on TDL. However, Codecademy has a very clean and modern interface, which is very delightful. The redesign of the WebIDE user interface incorporates some of the aesthetic values of Codeacademy.

JavaWIDE

Jenkins et al.[17] discuss using Java Wiki Integrated Development Environment (JavaWIDE) in high schools, and in two and four-year colleges. JavaWIDE is used to facilitate collaborative learning of Java among students. Whereas WebIDE focus on individualized learning, this approach focuses on the collaborative aspects of learning. In this approach, the professor and students collaborate on development of Java code together via the collaboration features of JavaWIDE. The students' responses to this were positive. However, in comparison to WebIDE, this is a very time intensive method for the instructor, and would not provide flexibility as all students and professors had to be online at the same time.

[1] http://www.codecademy.com

Web-based IDE Research Agenda

Kats et al.[20] map out what they see as a research agenda for Web-based IDEs, based on the results of several pilot studies. The relevant part of their work to WebIDE was in the development of a Web-based e-learning tool called WebLab. They developed it based off of the ACE editor[2] and the Cloud9 IDE[3]. They used WebLab to teach functional programming to first year computer science students. The system they describe is very similar to WebIDE. They describe how students are presented with instruction and a window to enter code on the same page. The programs that the students write are then executed on the server, where student-defined tests and instructor-defined tests can be run against the code. They even mention safety features that will kill programs that have infinite loops, much like WebIDE.

When they lay out their research agenda, they describe three areas of improvement that would also apply to WebIDE. The first area is that of improving student coding style. They suggest evaluating the student code to check for style as well as correctness. Second, they suggest researching ways to detect fraud. While the familiarity of the web-browser is beneficial, it can also be used by the students as a tool for getting solutions from a search engine. Kats et al. suggest the development of monitoring tools for fraud detection that could uncover undesirable behavior and notify both students and instructors. Lastly, they state that by observing beginning students and analyzing using the resulting data, it would be possible to tailor courses to help students with the concepts that they struggle with the most.

[2]http://ace.ajax.org/
[3]https://c9.io/

2.0.5 Gamification

Gamification is a new term that still does not have a settled definition. Deterding et al. [7] proposed the definition as "the use of game design elements in non-game contexts." They also note that the commercial development of 'gamified' software promises new and interesting lines of inquiry and data sources for HCI studies.

Game Mechanics in e-Learning

In an article entitled "Gamification: Using Game Mechanics to Enhance eLearning," [26] Rick Raymer states that the purpose of adding game mechanics to e-learning is to increase user engagement. One important element according to Raymer is measuring progress. He states that providing feedback to users in games or e-Learning allows them to know how much progress they have made. The author claims that the most effective way to do this is graphically. He also says that progress should be measured on several levels. One of the enhancements made to WebIDE is a progress bar described later in this paper.

Gamification for College Students

An orientation application for college students was a subject of a gamification study. Fitz-Walter et al. [9] added gaming elements to a mobile application that would assist new students to become familiar with the university. It was interesting to note that game elements were positive if they did not remove functionality. The conclusion of the authors was that game elements should be utilized only if they enhanced the existing functionality, and did not restrict the functionality of the application. The following sections will describe the various user interface improvement including the header bar, lab progress, and various text fields.

Chapter 3

Design Approach

3.1 Design Approach for UI Improvements

The overall goal of the user interface improvements was to create a modern, delightful user interface for WebIDE. Since most of the previous work that had gone into WebIDE had been concerned with functionality, the user interface was in dire need of work. It has been shown that the user interface will affect how people perceive a site. [5] It was my hope that improving the user interface would improve user perception of WebIDE.

3.1.1 Header Bar

Figure 3.1: Previous Header Bar

The previous version of WebIDE had a header bar with the logo of WebIDE. The only way to login to WebIDE was via a login link in the header bar, but this option was available on only one page of the WebIDE site. This made it difficult

and unintuitive to the user as to how they should go about logging in. I felt that if the first task that a user needed to accomplish was unintuitive and confusing, this would create a poor impression of the site. As the old saying goes, "You only get one chance to make a first impression." Making the login process easier and more intuitive was therefore one of my first steps.

Figure 3.2: New Header Bar Before Login

The new header has been improved in several ways. The aesthetic differences can be seen by comparing Figure 3.1 and Figure 3.2. By making the top header darker, it helps to visually anchor the page. Additionally the login link was added to every page so the user doesn't have to search for the specific page that has the login link.

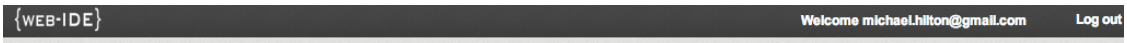

Figure 3.3: New Header Bar After Login

After the user logs in, they see the header bar in Figure 3.3. Once the user is logged in, their username as well as a Log out prompt are displayed on every page. This gives the user the information they need about the possible actions that they can take. If they click on their username, the link will take them to their user account page. If they click on the Log out option, it will log them out of the current session of WebIDE.

3.1.2 Lab Progress

One of the most significant changes to the UI was to change how the user navigates through the lab steps. Figure 3.4 shows the previous user interface.

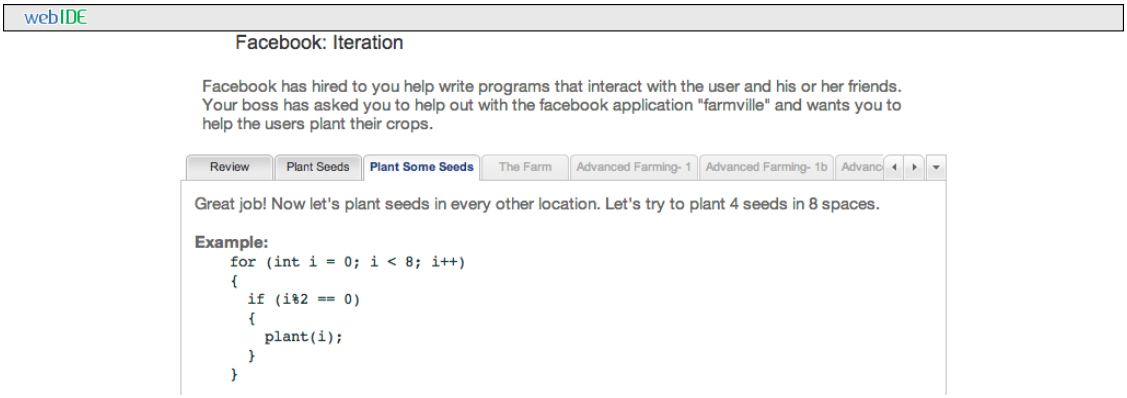

Figure 3.4: Previous Navigation

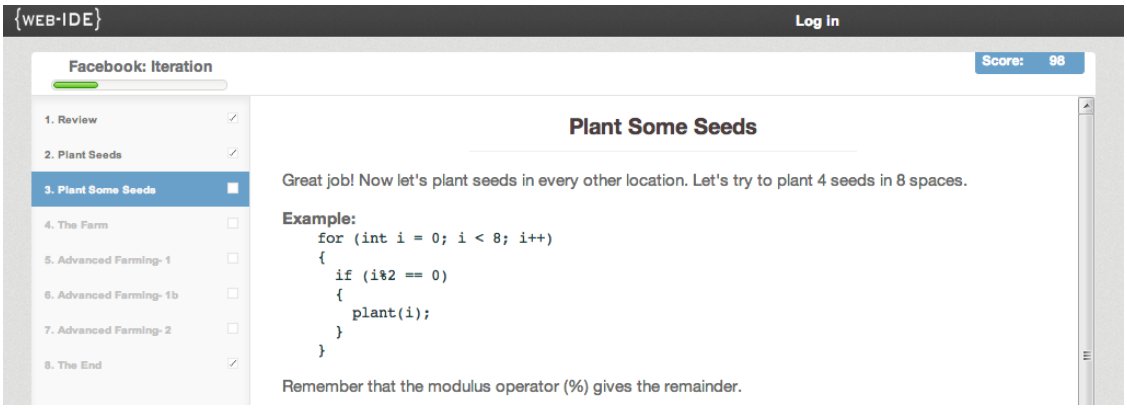

Figure 3.5: New Navigation

Each Lab is broken down into multiple steps. In Figure 3.4 the current lab is "Facebook: Iteration" and the current step is "Plant Some Seeds". The previous user interface used the default GWT Tabs element where each step in the lab had its own tab. Each step has three states: enabled, disabled and completed. The lab author has the ability to specify if a specific step should be locked until the previous step or steps have been completed. For a step to be completed, all the evaluators on that step must be successfully completed. In Figure 3.4, the first two steps are completed, the third is active, and the rest of the steps are locked.

Figure 3.5 shows the new user interface. There were several goals for the redesign of the navigation portion of the UI. The first goal was to help the user quickly and easily identify where they were in the lab and how to go to any other step that is available to them given their current progress. In addition to helping the user navigate through each step, another goal was also to enable better awareness on one's progress. One of the problems with the tab interface is that if there were more than a certain number of tabs, the tabs were off the screen and the user had to scroll left and right via the small buttons at the right of the tab bar. This made it difficult for users to know where they were in the lab and how many steps were left until the lab was complete.

Additionally it was desirable to give users a strong sense of completion and accomplishment in order to encourage them as they proceed through the lab. In order to achieve these goals, the concept of file tabs was re-evaluated and replaced with the concept of a list. Figure 3.5 shows the new navigation area in list form. The three states remain the same: enabled, disabled and completed. The objective of numbering the list of steps is to evoke the concept of a to-do list, with each step being marked off as it is completed. The current step is highlighted in blue to make it very obvious to the user which is the current step. By listing all of the steps, it makes it easy for the user to navigate from any step to any other step that is currently available to the user. The completed steps have been marked with a check box to make it obvious which steps are completed, and also to give the user a strong sense of accomplishment as each step is completed. In addition, a progress bar is located above the navigation area, providing a graphic representation of how far the user has progressed towards completing the entire lab.

3.1.3 Step Content Area

The most significant modification to the individual step content area is the changes to the user inputs. One of the main design goals for the user inputs was to improve user feedback. This included making it clear where input is required and giving positive reinforcement when the user gives a correct answer, as well as providing helpful feedback when the answer is incorrect in order to better help users correct their errors.

Text Fields

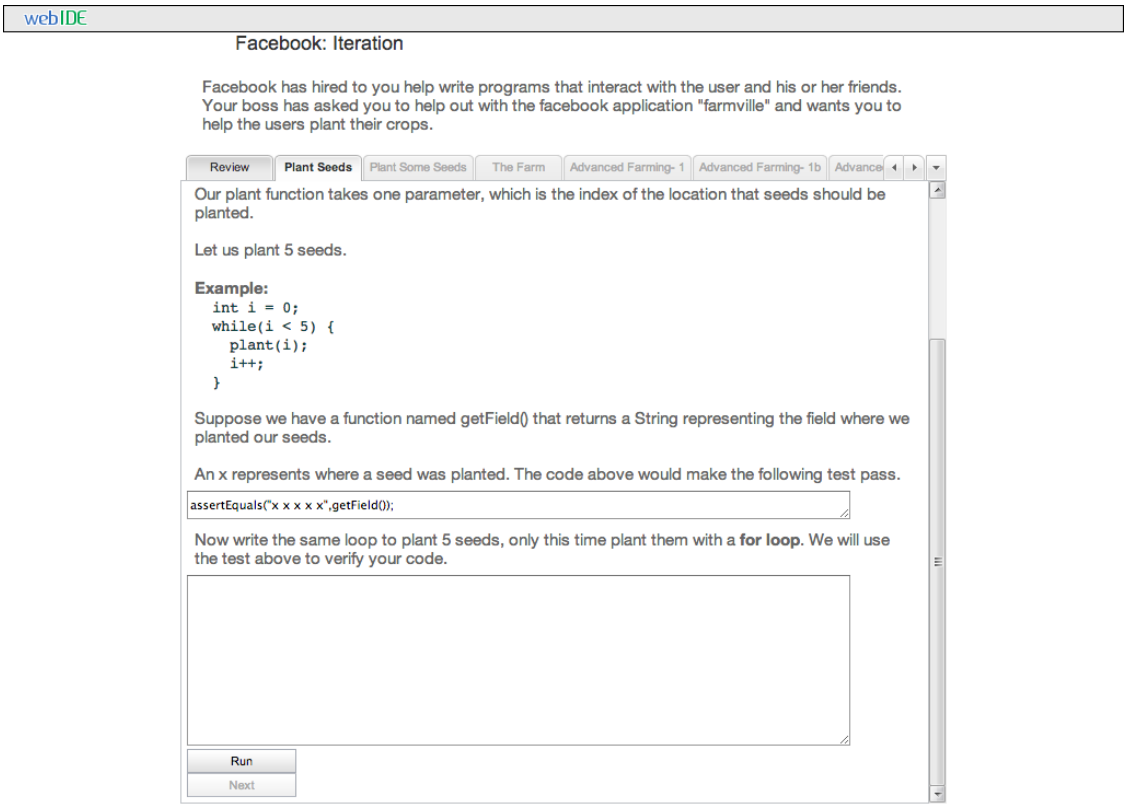

Figure 3.6: Previous User input

There were several design changes made to the text input area. Figure 3.6 and Figure 3.7 show the old and new text input areas. One design concern was

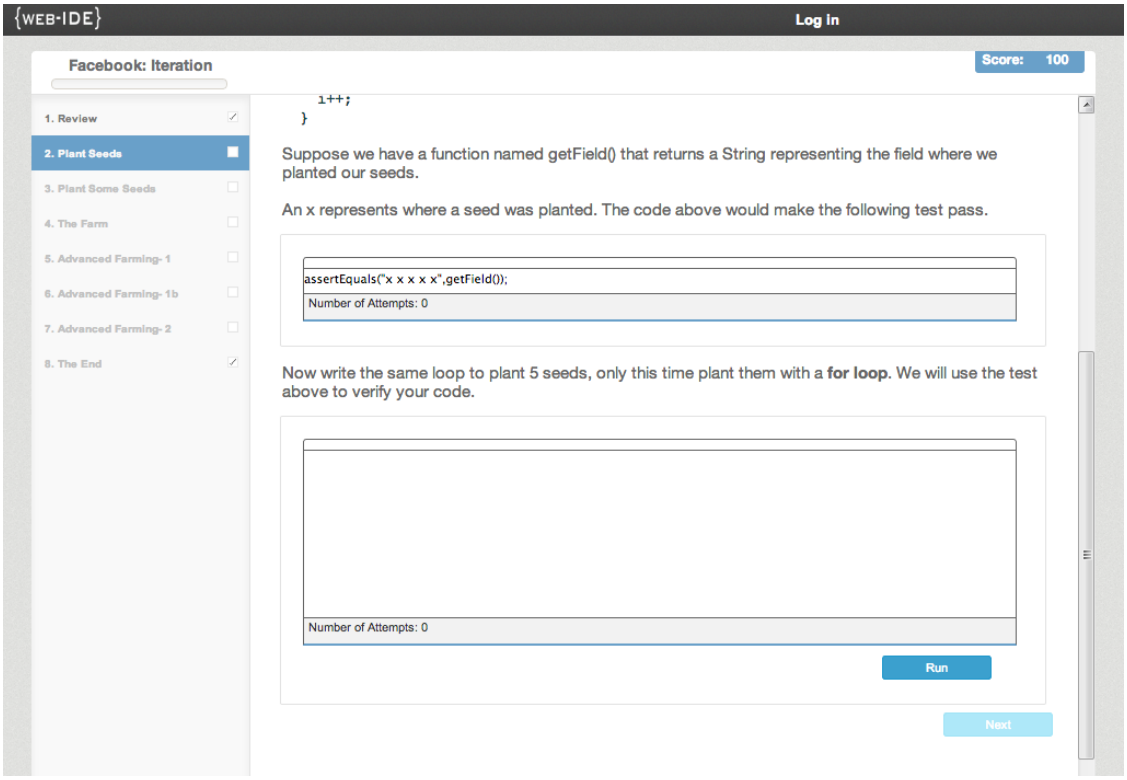

Figure 3.7: New User input

to increase the amount of communication with the user about their input. I wanted to more tightly couple feedback to the user with the immediate input received from the user. In addition, I wanted to provide information to the user about the number of attempts that they made in order to solve that specific problem. I wanted the design to communicate the purpose of the different design elements. The feedback flag was added just above the user input in order to clearly communicate the three possible states to the user: unattempted, incorrect, and correct. Due to the variance in the WebIDE specifications, sometimes the run button is included with the user input element, and sometimes it is specified to be in a different location by the lab author. If the user has specified a location for the run button, then the user's request is honored. If the user has not specified a location for the run button, then a thin border is placed around the user input, the

run button, and the feedback area. This border is used to convey the connection between the run button and the user input. Figure 3.7 shows an example of both scenarios. The first text area does not have an accompanying button, while the second text input area does have an accompanying button included in the outer container's border.

Incorrect Text Entry

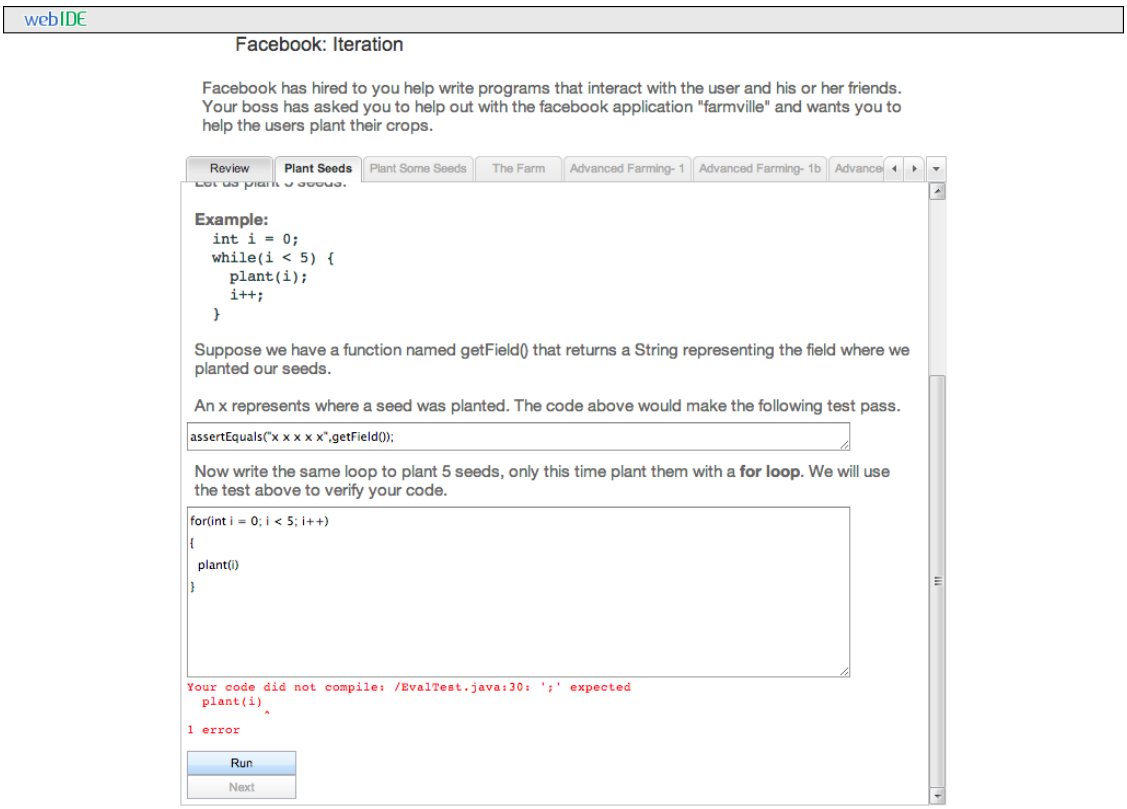

Figure 3.8: Previous User Input Incorrect

There were two main design concerns with the feedback to the user about incorrect answers. One concern was to ensure that the user not be confused about whether they answered the question correctly or not. If they answered the question incorrectly, it was important for that information to be immediately obvious.

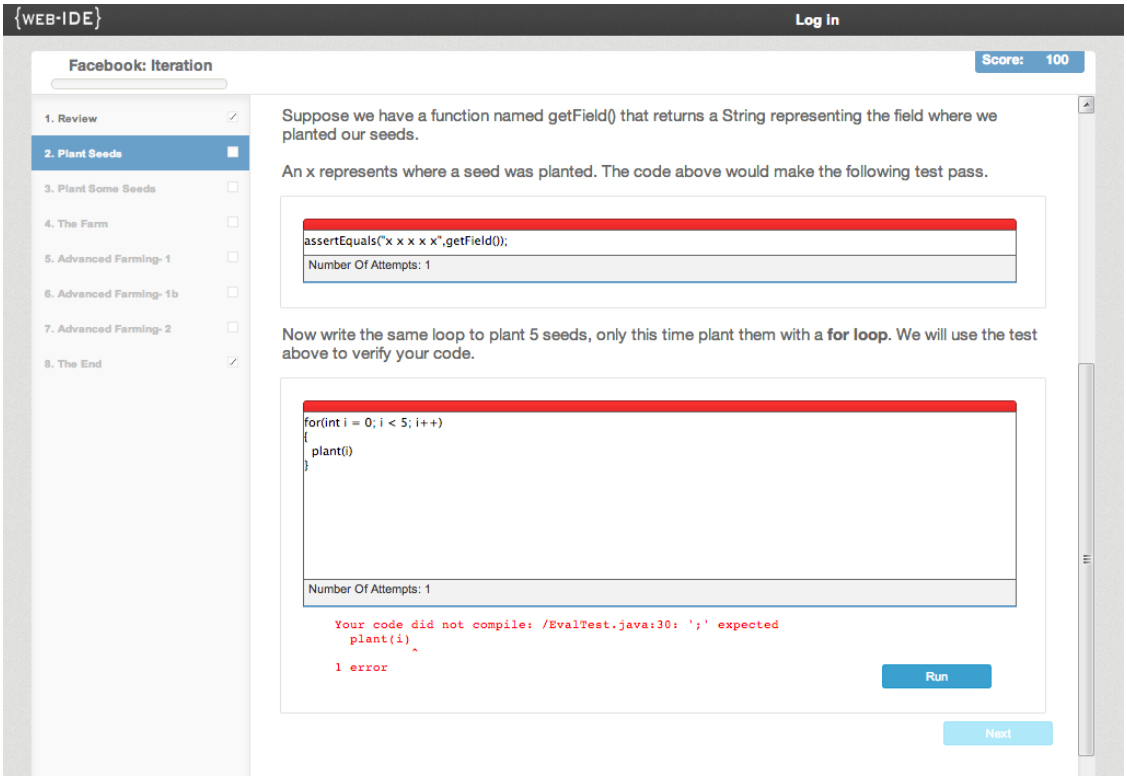

Figure 3.9: New User input Incorrect

It was also important to present a strong visual difference between correct and incorrect answers. The red flag was chosen to provide a strong visual confirmation that the answer is incorrect. Figure 3.8 shows the previous interface while Figure 3.9 shows the new interface after an incorrect answer has been given by the student.

Another design change was to move the run button in order to align it with the feedback text area. The feedback text was left aligned, while the run button was right aligned, and moved to the same horizontal line as feedback text area. This alignment creates a visual cue to let the user know that the feedback text is related to that specific run button. In this example the plant(i) is missing the ';'. The top box contains a unit test while the bottom box contains the code. The

run button will run the evaluator, and since the code is failing, it will set the flag to red, increment the number of attempts, and display a message to the user in the feedback area.

Correct Text Entry

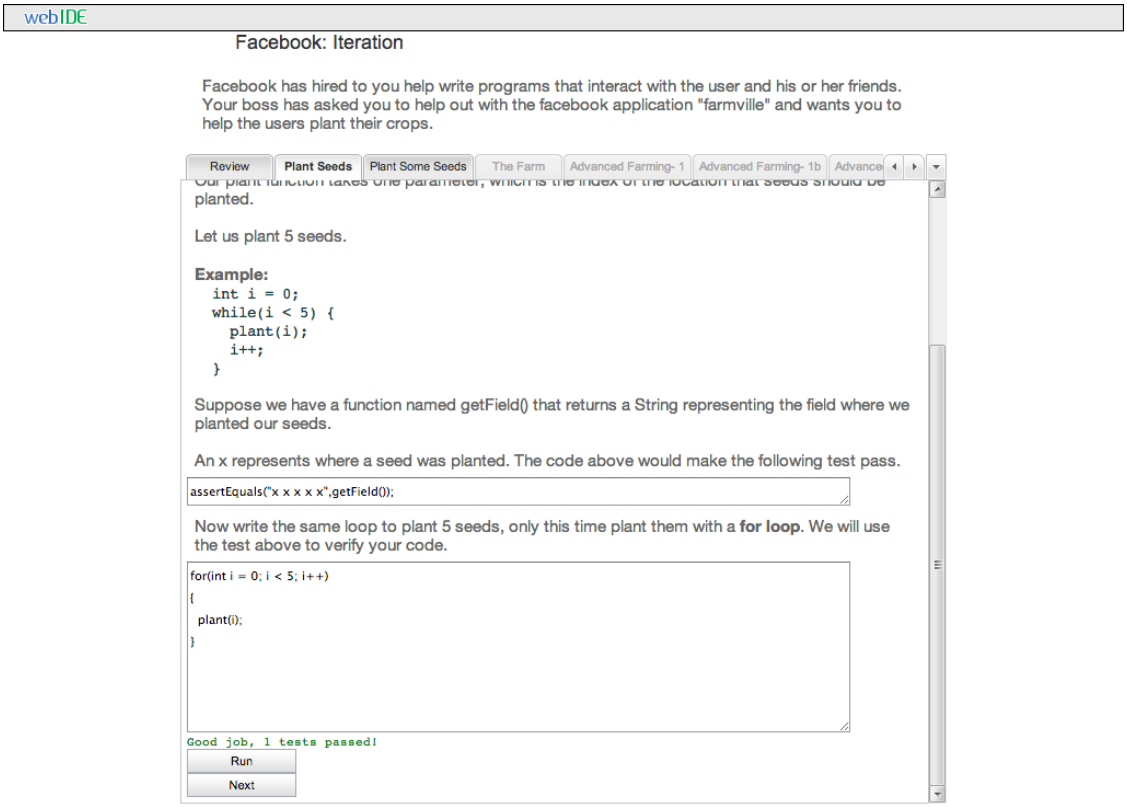

Figure 3.10: Previous User Input Correct Answer

Figure 3.11 shows the new interface when the user has correctly entered the code. For reference Figure 3.10 shows the previous version of the same interface. The primary design concern here was to give the user strong visual feedback that they answered that question correctly. By marking all the text boxes with a green flag there is a strong visual cue to the user that their answer is correct. The colors green and red were chosen as they are widely used to indicate success

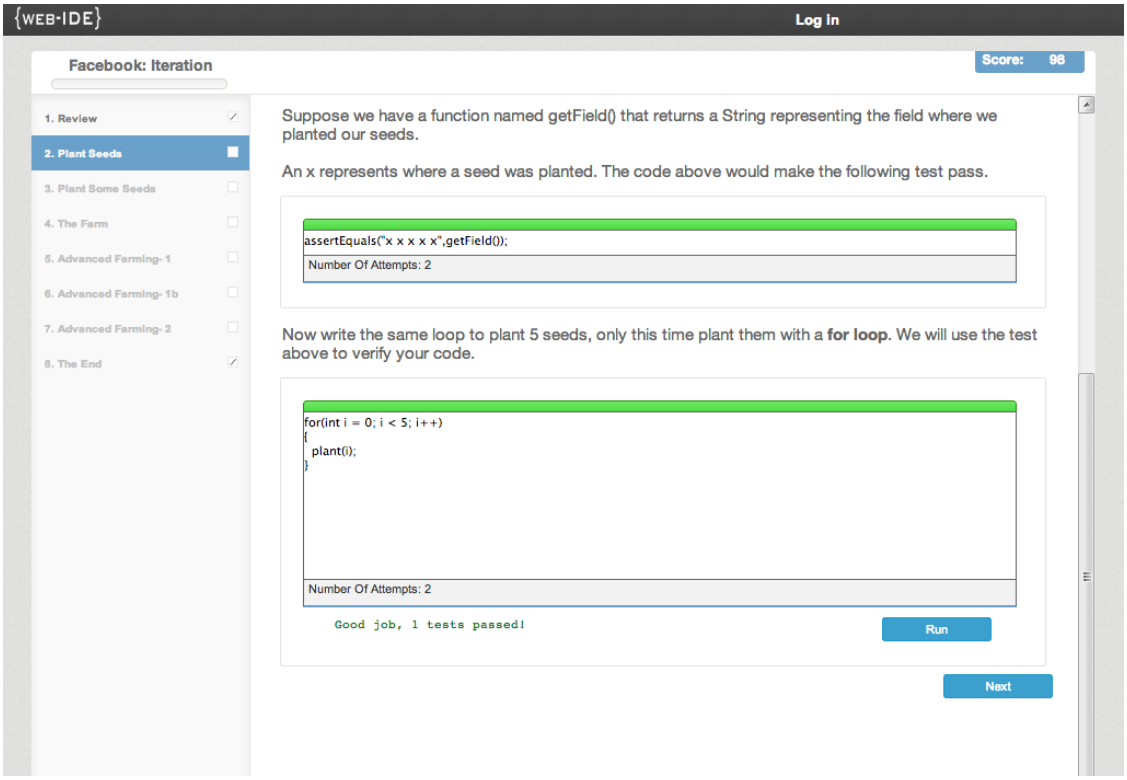

Figure 3.11: New User Input Correct Answer

and failure. This color use is also consistent with the JUnit framework, which the students will encounter later on.

Text Inputs

One option that the lab author had was that instead of specifying the larger, multi-line HTML text box, they could instead use the smaller, single-line HTML form element, text input field. Figure 3.12 shows an example of the text input. This text input would often be used to save space when multiple inputs needed to be evaluated at the same time. By putting a flag on each of the inputs, WebIDE is now able to tell the user which of the text boxes is correct and which is incorrect. Since these inputs are designed to be placed wherever the lab author indicates, it was important that these inputs be more compact than the much larger text

inputs. In order to accommodate this more compact design, a smaller, more compact flag was designed. This flag operates in a similar manner to the larger flags on the text inputs, but with a significantly reduced footprint. Figure 3.13 shows the new design for text inputs.

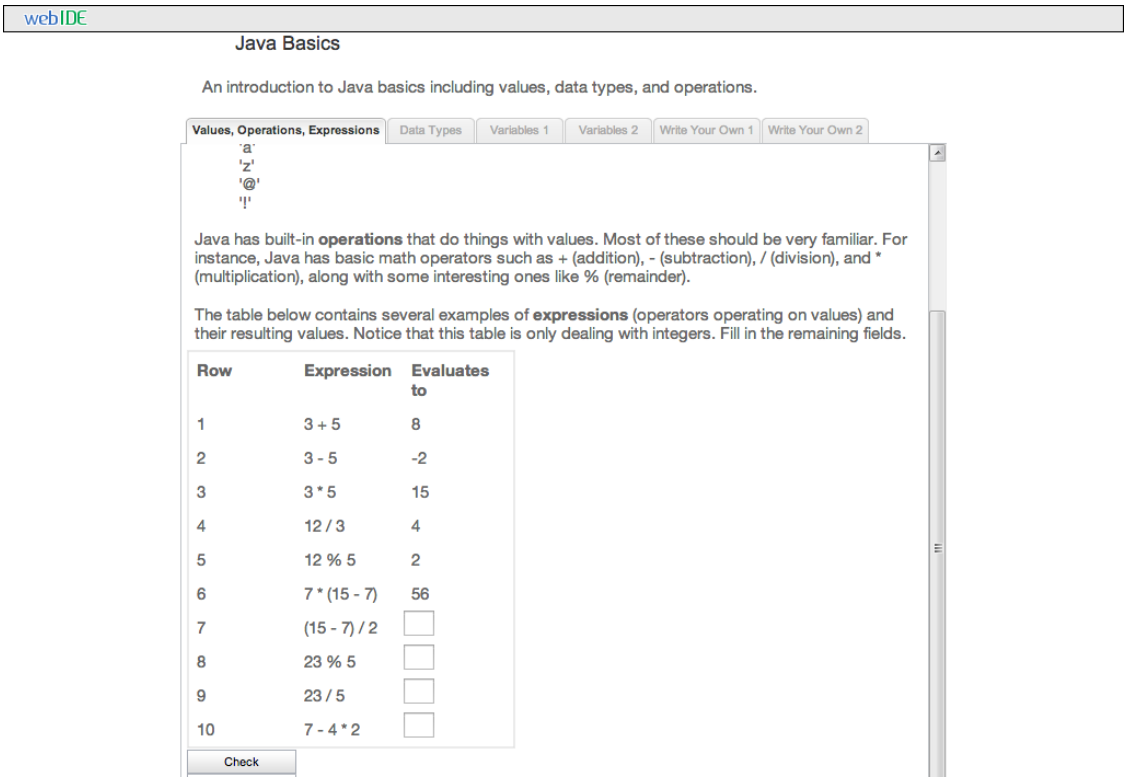

Figure 3.12: Previous User Multiple Input

Multiple Inputs Incorrect

Figure 3.14 shows the previous interface when a user has answered a question incorrectly. I felt that there were several problems with this interface and attempted to correct them. First there was a disconnect between the error message and the incorrect input that caused the error, especially when the run button is not in a location that logically aligns it with the user input. In order to give the user a stronger association between their answer and the returning informa-

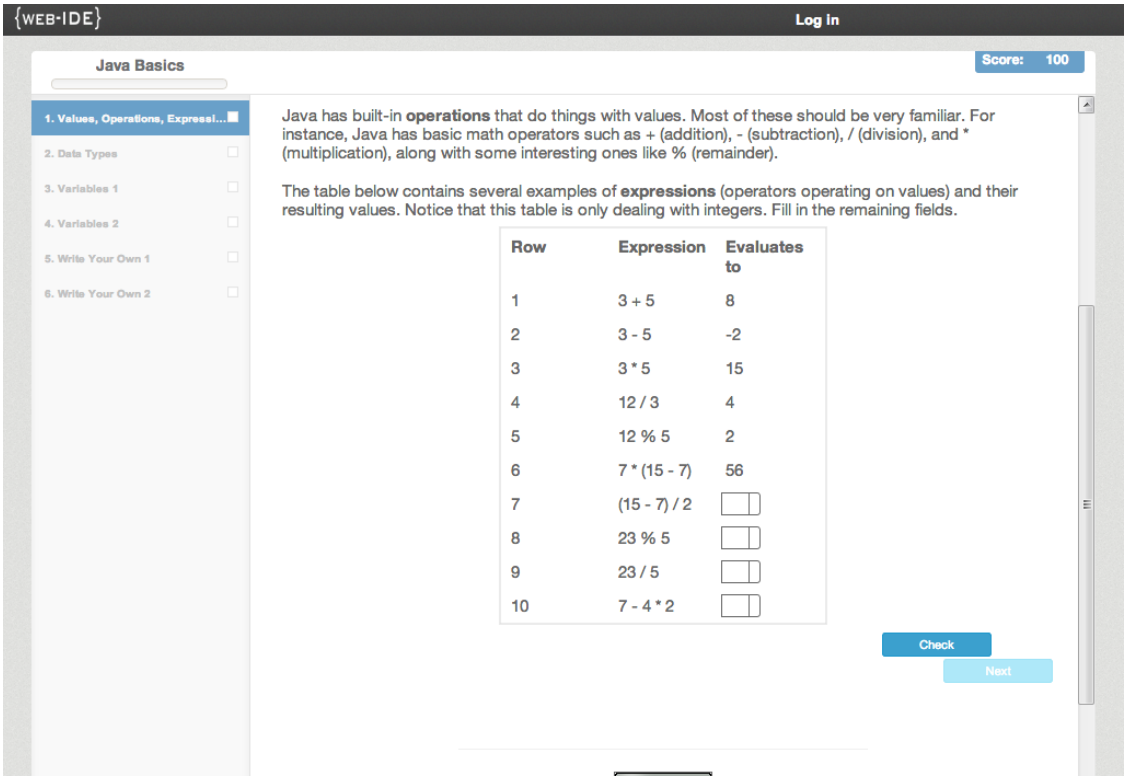

Figure 3.13: New User Multiple Input

tion from WebIDE, the response flags were added. Figure 3.15 shows the new interface with improvements. By adding the flag, it is now very obvious to the user which user inputs were evaluated on that run, as well as which user inputs were correct and which were incorrect. Also, by moving the response text to be inline with the check button, it is more apparent that that response text is from that evaluator button. I felt that this improved communication would result in a better overall experience for the end user.

Multiple Inputs Correct

Figure 3.16 shows the previous interface after the student has given all correct answers. I felt that one of the main issues was that when a user answered questions correctly, the feedback was very anti-climactic. I wanted the user to

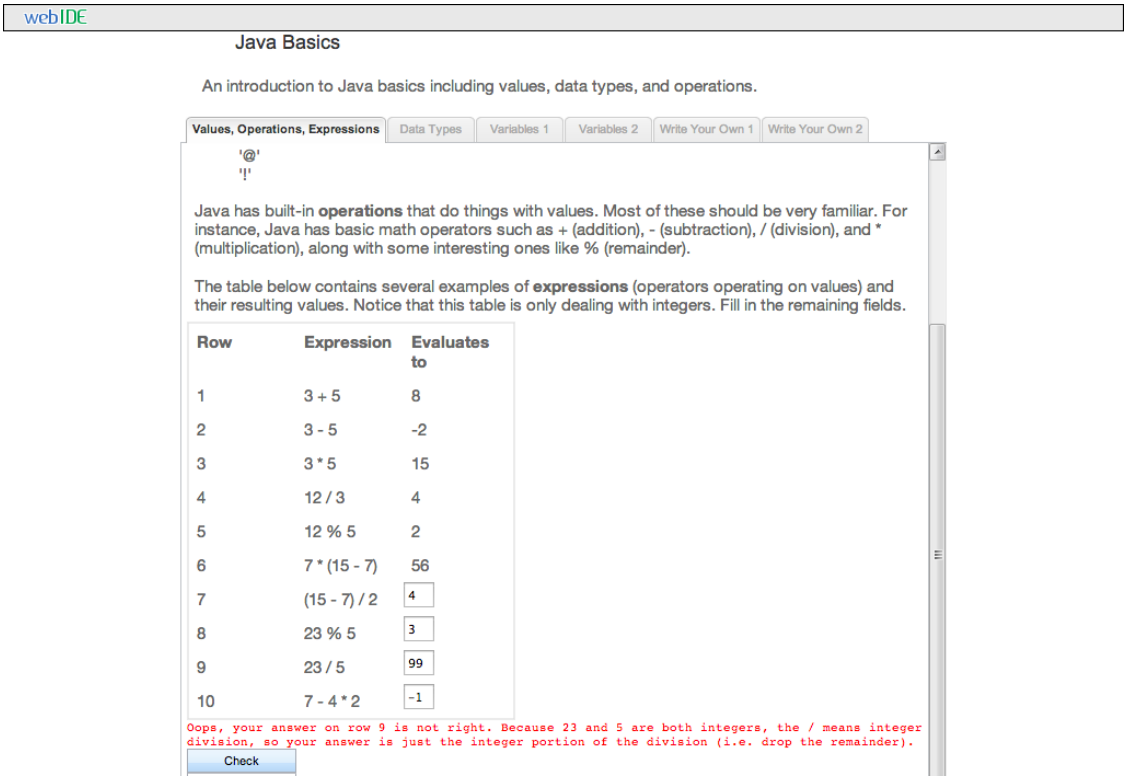

Figure 3.14: Previous User Multiple Input Incorrect

experience positive reinforcement when they answered a question correctly. To achieve this, I established two main criteria. The first was that it should be obvious to users that their input has been received and evaluated, and the second was that what they did was in fact correct. In Figure 3.17 the new interface is shown with positive user feedback. By setting the flags adjacent to the user input fields to green, it becomes clear that there has been a change in state, and that it is a positive state that the user is in. Also, by putting the flag immediately to the right of the user input field, the relationship between the user's input and notification of a correct result is coupled as tightly as possible.

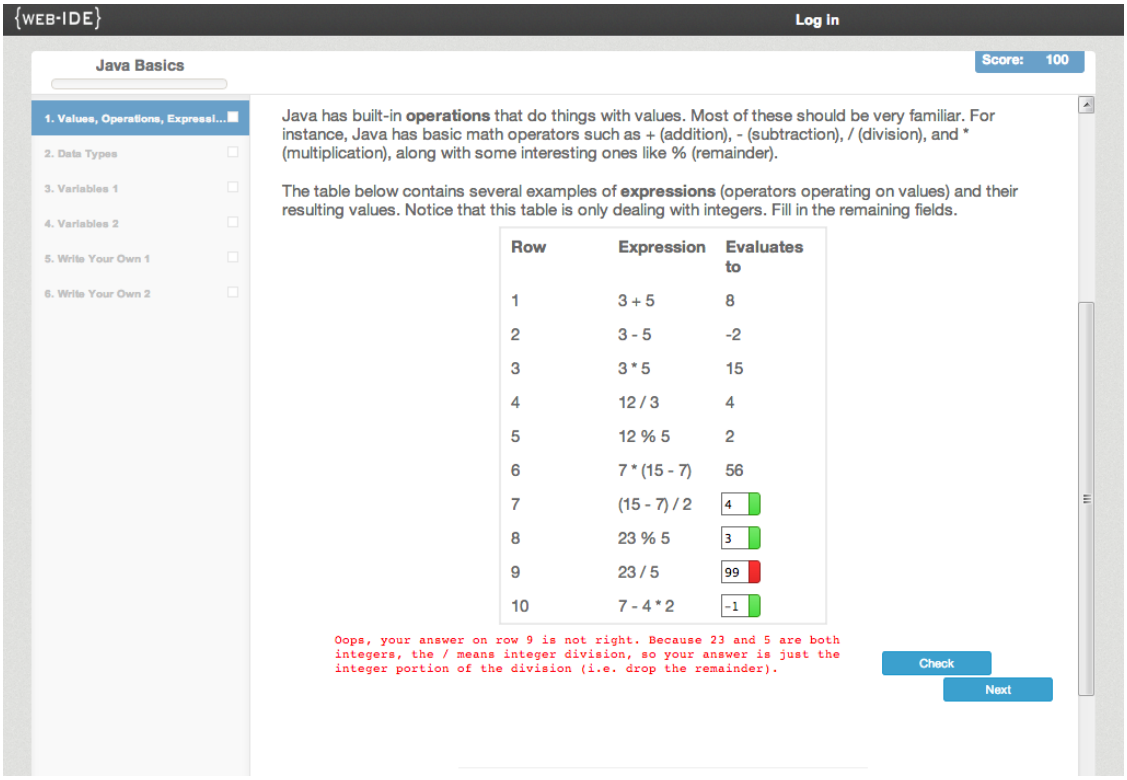

Figure 3.15: New User Multiple Input Incorrect

3.2 Design Approach for Gamification Element

Another goal for WebIDE was to encourage the students to think about their answers, and to encourage them to get the correct answer in as few tries as possible. One of the main advantages of Test Driven Development is that it encourages the developer to think about what they are going to do before they do it. The scoring mechanism was designed to encourage the behavior that we desired, which in this case was the student finding the correct answer in as few attempts as possible. In this section, I will present the design approach for the scoring element and the resulting scoring algorithm.

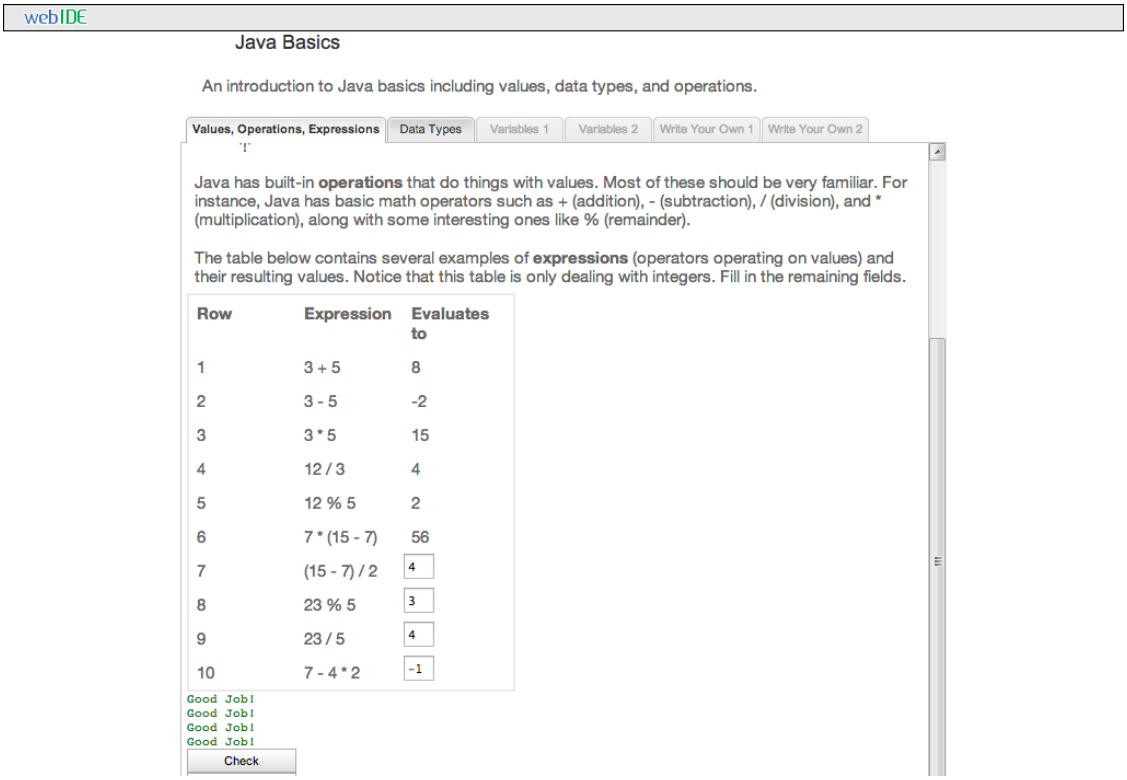

Figure 3.16: Previous User Multiple Input Correct

3.2.1 Design Approach for experiment UI

Two versions of the user interface were developed in order to conduct the experiment: the experimental user interface and the control user interface. Figure 3.18 shows the control user interface while Figure 3.19 shows the experimental interface. In order to study the effect of the scoring element on the students, both user interfaces were designed to be as similar as possible, with the one difference being the scoring element that we were attempting to study. The design decisions about the experimental user interface centered around communicating two pieces of information to the user. The first piece of information was that there was a score associated with their participation in the lab, and what that score currently was. The second piece of information was that WebIDE was tracking the number

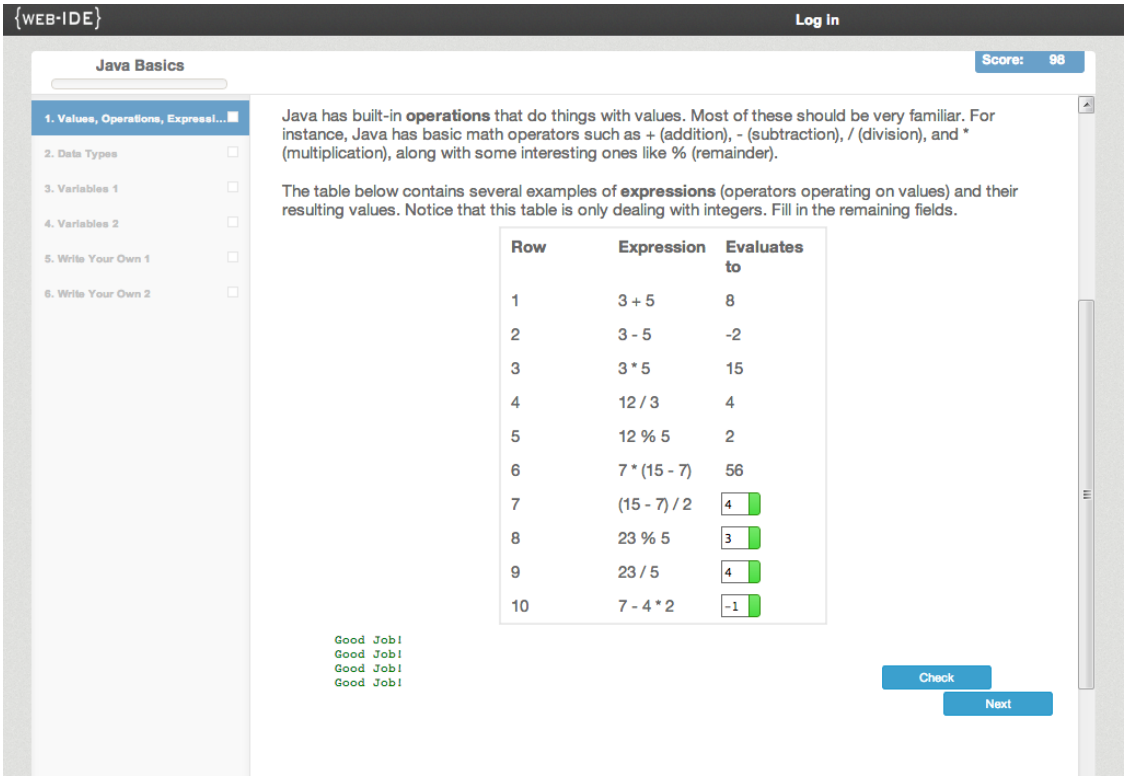

Figure 3.17: New User Multiple Input Correct

of attempts they made to answer each question. I also wanted them to be aware of the number of attempts they had made on the current question.

In order to be able to evaluate this experiment, the control group was presented with an almost identical user interface, except that the score elements where hidden from the user. There was nothing about the interface for the control group that indicated there was a score involved in their completing the lab, and there was no visual indication that the number of attempts were being tracked, nor were they informed of their current number of attempts.

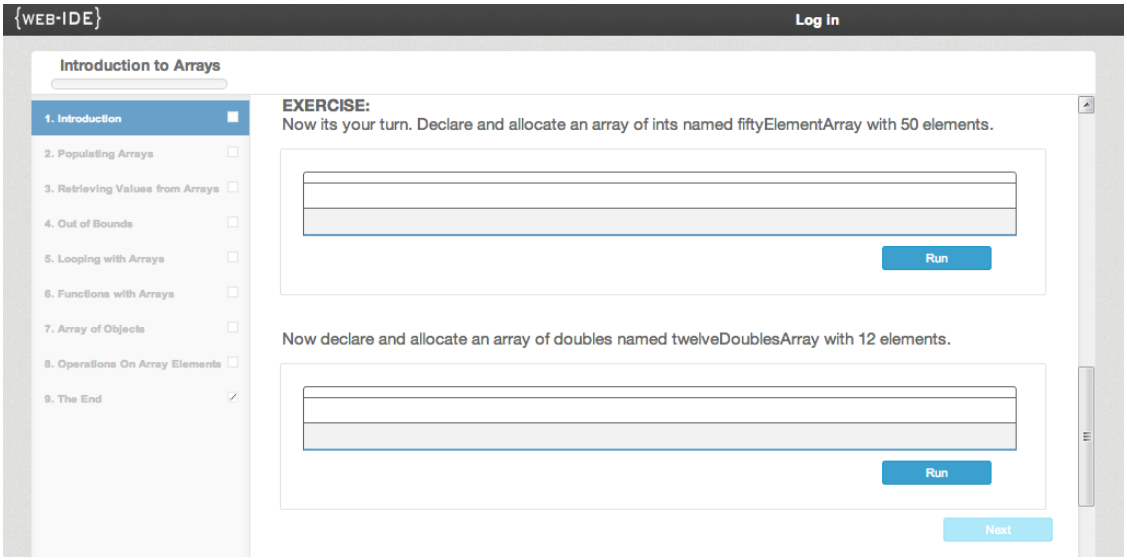

Figure 3.18: Control User Interface

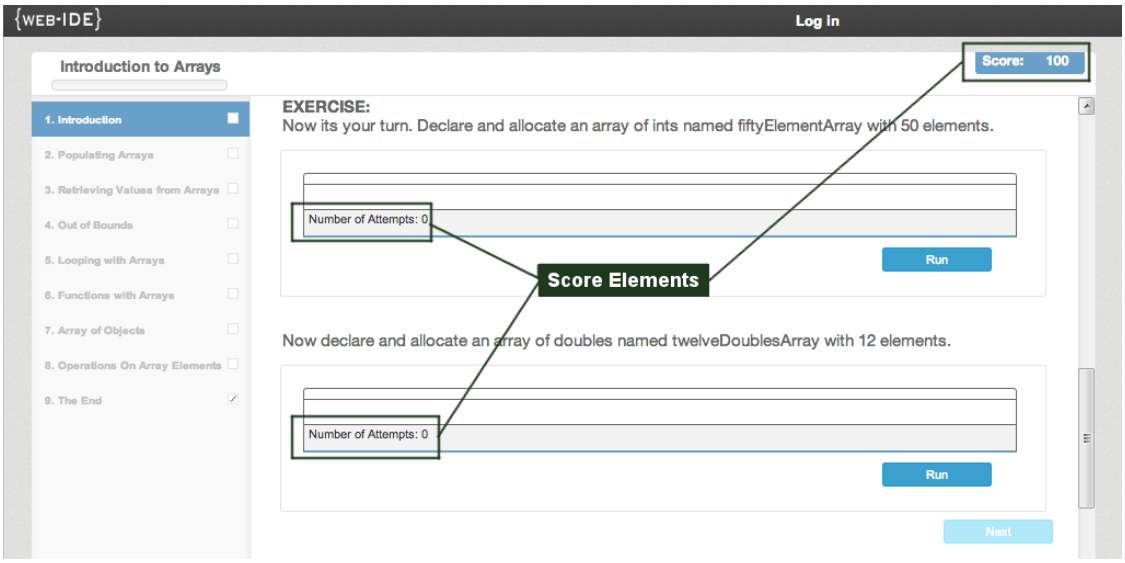

Figure 3.19: Experimental User Interface

3.2.2 Design Approach for Scoring Algorithm

As the goal of the scoring algorithm was to motivate students, the objective was for the score outcomes to be in a range that the students would find numerically familiar. For that reason, the desired score range was 60-100, in order to

closely resemble the traditional grading scale. The student would start with a score of 100, and as they progressed, they would lose points if they were not able to answer a question on the first attempt.

The score S is given by the formula

$$S = 100 - k \sum_{i=1}^{n} \min((a_i - 1), 4)$$

where n is the number of problems attempted, a_i is the number of attempts made on question i, and k is a scaling factor chosen to ensure that student grades remain in a reasonable range. In order to avoid excessively handicapping a student who is unable to figure out one question but did well on the others, the scoring algorithm only considers up to four attempts for each question. We set k as $10/p$, where p represents the total number of problems. This choice guarantees a lower bound on the student's score of

$$100 - (10/p)(4p) = 100 - 40 = 60$$

thus providing an intuitive connection to typical grading schemes. Due to the effect of k, the amount that the score decreases for each incorrect answer depends on the total number of problems in each lab.

Scoring example

Assume there is a lab that contains ten total evaluators. Table 3.1 describes what happens as the student progresses through the lab. The first evaluator is answered correctly on the first attempt, so their score is maintained at 100. The second evaluator is the one that the student finds the most difficult, and it takes a total of eight attempts for the student to answer the question correctly. It is important to note that the score will drop only for the first four attempts, so attempts number five, six, seven and eight will have no further effect on the

current evaluator	Answer	# of attempts	# of incorrect attempts	# of incorrect attempts counted	total # of attempts counted	Score
1	Correct	1	0	0	1	100
2	Incorrect	1	0	0	2	100
2	Incorrect	2	1	1	3	99
2	Incorrect	3	2	2	4	98
2	Incorrect	4	3	3	5	97
2	Incorrect	5	4	3	5	97
2	Incorrect	6	5	3	5	97
2	Incorrect	7	6	3	5	97
2	Correct	8	7	3	5	97
3	Incorrect	1	0	0	6	97
3	Correct	2	1	1	7	96
4	Correct	1	0	0	8	96
5	Correct	1	0	0	9	96
6	Correct	1	0	0	10	96
7	Incorrect	1	0	0	11	96
7	Incorrect	2	1	1	12	95
7	Correct	3	2	2	13	94
8	Correct	1	0	0	14	94
9	Correct	1	0	0	15	94
10	Incorrect	1	0	0	16	94
10	Incorrect	2	1	1	17	93
10	Correct	3	2	2	18	92

Table 3.1: Lab Scoring

student's score. When the student completes evaluators 4, 5 and 6 correctly on the first attempt, the score is maintained.

One quirk of the scoring algorithm is that since one answer is allowed for each question, the first incorrect answer does not cause the score to go down, but the second attempt, even if correct, will reduce the score. This is demonstrated in Table 3.1 by evaluator 3. The student's score is not reduced on the first incorrect answer, but with the second correct answer. In the future, the scoring algorithm could be improved by only tracking incorrect attempts.

Chapter 4

Evaluation

4.1 Evaluation Background

WebIDE was evaluated at California Polytechnic State University in San Luis Obispo in two different sections of the CSC 123 class. This class was titled "Introduction to Computing: Mobile." The two sections had 34 and 37 students respectively. This class was designed as an introduction to Computer Science. It is the first class students in the major are required to take. The class started with one day of instruction in Scratch,[1] a visual coding language.[27] Scratch is a programming language which allows for objects to be manipulated with drag and drop visual blocks. The blocks can perform an action such as moving a sprite, or control execution flow, such as repeat an action a certain number of times. The next instructional topic was App Inventor.[2] This was the topic for the next three and a half weeks. App Inventor is a freely available programming environment that was initially created at Google, but has transitioned to the MIT Center for Mobile Learning. App Inventor can be used to create Android apps using a visual

[1] http://scratch.mit.edu/
[2] http://appinventor.mit.edu/

drag-drop interface. It follows the same programming paradigm used by Scratch, with drag and drop programming blocks.[10] After App Inventor the rest of the course is spent teaching students to learn to program Android apps in Java. Table 4.1 summarizes the course topics by week.

Week	Topic
1	Programming Topics, Scratch
2	Functions, Lists, Iteration, App Inventor
3	Presentations, More App Inventor
4	Testing Project 1 (App Inventor), Presentations
5	Java Basics
6	More Java, Java Loops and Arrays
7	Eclipse, Android overview
8	Project 2 (Android) Presentations, More Java, More Android
9	Java/Android practice
10	Project 2 (Android) Workshop
11	Project 2 (Android) Presentations

Table 4.1: Class Topics

The course was being offered for the third time. The goal for this class was to improve first year success and retention.[13] This year the material was the same as previous years, but it was being presented in a new format. For the Java portion of the class, the instructor recorded a series of video lectures, which the students were assigned to watch before coming to class. The class time was then used to do the WebIDE labs and quizzes. Over the course of the class the students completed a total of nine labs, as well as two large projects. The Labs and Quizzes combined for 25% of the students' grade, and the Projects were worth a total of 40% of the final grade.

At the beginning of the quarter, a pre-quiz was administered to determine how much Java experience the students had. Since Java knowledge was not a

prerequisite for this class, it was not unexpected that a large group of students scored zero on the pre-quiz. While the differences between the two groups were not statistically significant, one can see in Figure 4.1 that the control group had better scores than the experimental group. The control group was selected by a coin flip before the results of the quiz were known.

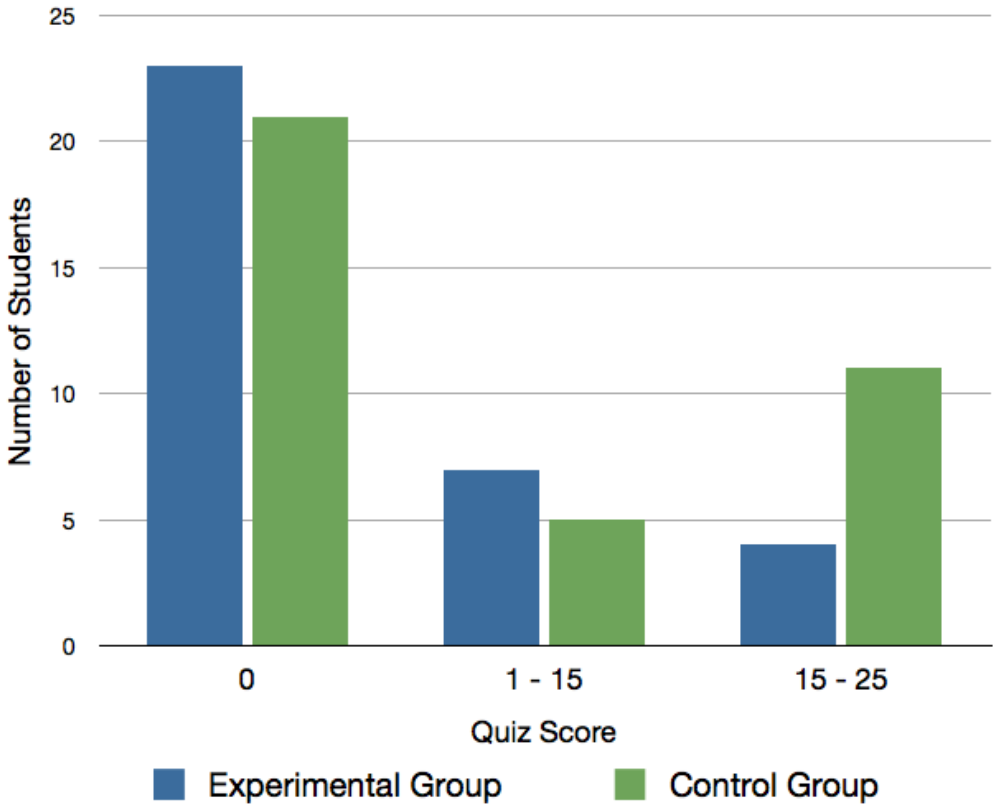

Figure 4.1: Java Prequiz Scores

4.2 Evaluating delightfulness of user interface

In order to evaluate whether the changes resulted in a user interface that was more delightful to the users than the original user interface, the students

were surveyed about WebIDE to record their impressions. In the previous year, the same survey was given to the students of one section of CSC123 who used the previous version of WebIDE. The survey was administered online through surveymonkey,[3] and was completed by the students after each WebIDE lab. For the delightfulness experiment, the survey results from 2011 were treated as the control group, and the results from both sections of the 2012 class were the experimental group. The students were told that their answers would not affect their grade, and their responses were recorded with a standard Likert scale. There were some questions that were not relevant to this discussion that dealt with the difficulty of the lab materials, but there were two questions whose response was tracked for this experiment. Table 4.2 shows the relevant questions.

Statement	Possible Answers				
This lab was fun.	1 - Strongly Agree	2	3	4	5 - Strongly Disagree
I liked this lab.	1 - Strongly Agree	2	3	4	5 - Strongly Disagree

Table 4.2: Survey Questions

Table 4.3 shows the average responses of the students to the two relevant questions on the survey. In the Likert scale, the smaller values are more positive impressions. One can see that the results from the experimental group are slightly better across the board than the results from the Control group. The results were evaluated with a two sample T-Test. For the first question $p = 0.648$ while $p = 0.443$ for the second question. As these values were $> .05$ the difference is not statistically significant. We cannot therefore assert that the new interface is statistically preferred by the students. However, it does seem to lean this way, so further experiments may result in more conclusive results. One possible reason for the difference is that the students who used the new interface were not shown the old interface. If they had been shown both interfaces and asked which one

[3] http://www.surveymonkey.com/

they liked more, there might be a more significant difference in how they felt about each interface.

	Lab was Fun		Liked Lab	
Lab Topic	Exp	Control	Exp	Control
Java Basics	2.01	2.27	1.83	2.14
Selection	2.01	2.27	1.83	2.14
Methods	2.014	2.27	1.83	2.14
Classes	2.51	2.52	2.51	2.48
Loops	2.25	2.22	2.46	2.19
Arrays	2.25	2.59	2.46	3.06
Intro to Android	2.93	3.38	2.78	3.14

Table 4.3: Student Opinions in Average Likert Scale

4.3 Game Scores

Figure 4.2: WebIDE lab Scores

In addition to tracking the user impressions of the user interface, another experiment was conducted that compared user behavior. This experiment was conducted using only the two sections from Fall 2012. One section was chosen

Lab Name	Mean Control Score	Mean Exp. Score	p value
Java Basics	98.03	99. 29	0.012
Java Int Operations	96.54	98.18	0.005
Facebook: Iteration	82.19	88.87	0.000
Facebook:Selection	96.27	98.22	0.012
Introduction to Arrays	87.85	92.12	0.003
Android: Hello, World!	87.68	92.79	0.000
Android: Rock-Paper-Scissors	85.33	91.43	0.002

Table 4.4: WebIDE Scores

at random to be the control group, while the other section was the experimental group. In this instance, the control group was not given any scoring information, while the experimental group was provided information on how they were being scored based on the number of attempts it took them to solve each problem as described in Section 3.2.1. Table 4.4 shows how each section scored on each lab. The scores were evaluated with a Two-Sample T-Test and the p values are also included in Table 4.4. The test used for significance was that of $p < 0.05$. Using this criterion, every lab score was determined to be significantly higher for the experimental group vs. the control group. Figure 4.2 shows the same information in chart form. From this data we can determine that the game mechanism had a significant effect on the behavior of the students, and that the scoring mechanism was effective in changing behavior as users interacted with WebIDE.

4.4 Comprehensive Lab Score

At the end of the WebIDE portion of the course, a comprehensive lab quiz was given to the students that covered all the topics covered by the WebIDE labs. The results of that quiz are shown in Table 4.5. While the experimental group did score better on the comprehensive quiz on average, unfortunately there was

a very high standard deviation in the results. When the Two-Sample T-Test was performed, the p value = 0.822. We cannot therefore assert that the students who were in the experimental group performed better on the quiz than the control group. While it does look like they might have performed slightly better, more studies are needed to determine conclusively if this scoring mechanism will produce better results. Since the results were in a region of statistical uncertainty, we can at least say that the gaming elements were not proven to be detrimental to the students.

Section	Mean	Std Dev
Control	69.3	18.4
Experimental	70.3	16.7

Table 4.5: Comprehensive Quiz Scores

4.5 Threats to validity

4.5.1 Persistence Issues

In the initial experimental version, there was an issue of some students' input being saved while the input of other students was not. On further inspection, it turned out that a field in the database was a varchar(256). If a student attempted to save an input string longer then 256 characters, it would cause an error that would result in the data not being persisted. This was an issue only for the first two labs, after which the problem was resolved by changing the field from a varchar(256) to a mediumtext.

4.5.2 Cheating

Another threat to validity is a student that was observed cheating. The instructor observed that the student had figured out that the same labs were available at the www.web-ide.org site, even though the students were instructed to use another url configured specifically for this experiment. By completing steps of the lab and finding the correct answer out on the www site, and then inputting that into the experiment site, he was able to score a 100 on the lab. While cheating is in no way condoned, at least we can observe that the scoring mechanism motivated the student enough to find a way to improve his score. Because the labs were completed in-class while the instructor was observing students and answering questions, we believe this behavior was rare and perhaps even limited to this one individual.

4.5.3 Instructor

The instructor was aware of this experiment as it was being run. He also has a vested interest in the success of WebIDE, as he is the project lead. While he attempted to remain as unbiased as possible, it is possible that his bias may have affected the outcome of the study. One step that was taken to counter any possible bias is that all the quizzes were graded as a group, both the control and experimental sections, without taking notice to which section that student belonged.

4.5.4 Time of Day

Some of the difference between the sections could be attributed to the different time of day of the classes. One of the classes was in the morning, while the other class was held after lunch. In addition to the possibility of class time affecting

the body rhythms of the students, the instructor noticed that the students in the morning section didn't seem to rush through the material in order to leave early, whereas that behavior was observed in the afternoon section.

Chapter 5

Challenges and Future Work

5.0.5 Challenges

ACE editor

One challenge consisted of attempting to incorporate the ACE editor into WebIDE. ACE editor is an embeddable code editor written in JavaScript.[1] The ACE editor offers much of the same functionality provided by a dedicated code editor, but in a browser interface. Some of the features I would like to have been able to include in WebIDE are line numbers, automatic code coloring, and automatic indenting. This added functionality would not only be helpful to the students as they wrote the code, but it would prepare them for how things would look once they moved on to using application Integrated Development Environments (IDEs), such as Eclipse. However, it was a significant challenge getting the ACE editor to work correctly with GWT, which is the framework which was used to develop WebIDE. ACE is a javascript component, and in order to pre-populate it with text, the editor must first be initialized with a javascript call. However, for the javascript call to happen, the ACE object must

[1] http://ace.ajax.org/

be added to the DOM. Before the page in WebIDE is added to the DOM, all the elements on the page, including the ACE editor must be initialized. This obviously creates a circular dependency which I was unable to break. There is currently a second version of WebIDE under development, and I hope that with the new architecture changes, we will be able to incorporate the ACE editor.

Answer Flags

One significant challenge was that of setting a flag on every text box that was evaluated in order to let the user know if the last attempt was correct or incorrect. The largest challenge was identifying which text box was associated with which evaluation. In the current WebIDE architecture, any evaluator may have zero to many user inputs associated with it. Also, a user input may have zero or more evaluators, although it usually has at least one. The goal was to give feedback to users as tightly coupled to the input as possible. Also, since there could be more than one input associated with an evaluator button, if one of the inputs was correct, and another of the inputs was incorrect, it was important to only flag the incorrect answer as being wrong, while flagging the correct answer as being correct. Additionally, since there can be more than one evaluator run at a time, it would be possible to have return values of correct, incorrect, and correct as the result of three evaluators run on a single text box. In order to correctly flag the user inputs, a data structure was developed that would keep track of all the attempts against all the user inputs for that specific step. For each evaluator run, all the user's inputs are stored in the data structure with the results from that evaluator run. If the evaluator returns false, it is stored in the data structure. If the evaluator returns true, before it is stored, there is a check to make sure that a true is not overwriting a false. This way all the evaluation

information is stored and every user input on the page can be flagged with the correct status.

Tracking Number of Attempts

Another challenging aspect of the work was tracking the number of times each student attempted a specific question. This was a challenge because not only did the data need to be persisted into the database, it also needed to be restored if a user reloaded a lab in progress. Additionally, the score needed to be computed for each lab. In the end the number of attempts was persisted in the database, and then the score was recalculated whenever it was needed.

5.0.6 Future Work

Based on the experience gathered during this effort, I would like to propose a few directions in which future work on WebIDE can proceed.

Diverse Scoring Algorithms

Since WebIDE is designed to be a pedagogical tool, it would be very nice to allow lab authors to specify how they would like the lab to be scored. If lab authors were able to define a scoring algorithm, then any lab author could conduct experiments by varying the scoring algorithm. Up to now, WebIDE has been focused primarily on Test Driven Learning. However, with author-specified scoring, WebIDE could be adapted to run almost any scoring based experiment. This would allow greater flexibility when it came to evaluating different learning paradigms.

Cooperative Competition

One future improvement to WebIDE would be to enhance the learning experience by combining cooperation and competition. This could be achieved by grouping students into teams that would compete against each other. One potential way to do this would be to have different sections of a class compete against each other. This would foster a sense of cooperation within each section as students worked together to increase their overall score, but also there would be the competitive aspect of trying to beat the other section. In order for this to be possible, a leader board mechanism would need to be developed to display the scores from each team, so the students could be aware of how the competition is going. In addition, a team mechanism would need to be developed so an instructor could group students into teams to compete against each other.

Creative Engagement

Additional development could include continuing to develop the gamification aspects of WebIDE. In addition to the competitive aspects of a gaming mechanism, there could be ways to allow users to engage WebIDE in a more creative manner. One possible way this could be developed is by creating a code gallery that would allow students to publish their solutions to certain problems. Once a student had finished a lab, they would be able to view the code gallery and view other students' solutions to the same problem. This would most be most effective for the larger more complex problems. Which elements of any given lab would be eligible for uploading to the code gallery would be up to the author of the lab. This would allow students to attempt to solve problems in a creative way, as well as helping them to learn how to read other authors' code, which would be a beneficial skill to have.

Chapter 6

Conclusion

WebIDE has been shown to be beneficial to beginning students learning how to program, specifically by introducing them to Test Driven principles early in their learning trajectory. However, the existing interface design was less than user friendly, and various aspects of WebIDE were improved as a result of this project. These improvements resulted in differing levels of success, as measured by various evaluation methods

The user interface improvements resulted in an interface that was more cohesive and better designed than the original. The students seemed to like it better, but we were not able to definitively prove this finding. It is my hope that this redesigned interface will encourage other educators to use WebIDE and increase its adoption rate.

I was able to show that by adding the gamification elements to WebIDE we were able to significantly impact how the students approached working on, and answering, the labs. While we were not able to show that this significantly improved test scores, it did not prove to be a negative for the students either. One

of the benefits of a tool like WebIDE is the ability to collect metrics on student learning in a way that would have been impossible using traditional pedagogical methods. It is my hope that WebIDE will continue to be a tool that instructors use and that this will result in improvements in the science of computer science instruction.

Bibliography

[1] T. Berns. Usability and user-centred design, a necessity for efficient e-learning. *International Journal of The Computer, the Internet and Management*, 12(2):20–25, 2004.

[2] S. Card, T. Moran, and A. Newell. *The psychology of human-computer interaction*. CRC, 1986.

[3] M. Corry, T. Frick, and L. Hansen. User-centered design and usability testing of a web site: An illustrative case study. *Educational Technology Research and Development*, 45(4):65–76, 1997.

[4] M. Costabile, M. De Marsico, R. Lanzilotti, V. Plantamura, and T. Roselli. On the usability evaluation of e-learning applications. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 6b–6b. IEEE, 2005.

[5] A. De Angeli, A. Sutcliffe, and J. Hartmann. Interaction, usability and aesthetics: what influences users' preferences? In *Proceedings of the 6th conference on Designing Interactive systems*, DIS '06, pages 271–280, New York, NY, USA, 2006. ACM.

[6] P. Denning. What is software quality. *Communications of the ACM*, 35(1):13–15, 1992.

[7] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM.

[8] T. Dvornik, D. Janzen, J. Clements, and O. Dekhtyar. Supporting introductory test-driven labs with webide. In *Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CS Conference on*, pages 51 –60, may 2011.

[9] Z. Fitz-Walter, D. Tjondronegoro, and P. Wyeth. Orientation passport: using gamification to engage university students. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, OzCHI '11, pages 122–125, New York, NY, USA, 2011. ACM.

[10] J. Gray, H. Abelson, D. Wolber, and M. Friend. Teaching CS principles with app inventor. In *Proceedings of the 50th Annual Southeast Regional Conference*, ACM-SE '12, pages 405–406, New York, NY, USA, 2012. ACM.

[11] A. Green, H. Huttenrauch, M. Norman, L. Oestreicher, and K. Eklundh. User centered design for intelligent service robots. In *Robot and Human Interactive Communication, 2000. RO-MAN 2000. Proceedings. 9th IEEE International Workshop on*, pages 161–166. IEEE, 2000.

[12] J. Hartmann, A. Sutcliffe, and A. De Angeli. Towards a theory of user judgment of aesthetics and user interface quality. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 15(4):1–30, 2008.

[13] M. Haungs, C. Clark, J. Clements, and D. Janzen. Improving first-year success and retention through interest-based CS0 courses. In *Proceedings of the*

43rd ACM technical symposium on Computer Science Education, SIGCSE '12, pages 589–594, New York, NY, USA, 2012. ACM.

[14] D. Hix, J. Swan, J. Gabbard, M. McGee, J. Durbin, T. King, et al. User-centered design and evaluation of a real-time battlefield visualization virtual environment. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 96–103. IEEE, 1999.

[15] K. Holtzblatt. What makes things cool?: intentional design for innovation. *interactions*, 18(6):40–47, 2011.

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

[17] J. Jenkins, E. Brannock, T. Cooper, S. Dekhane, M. Hall, and M. Nguyen. Perspectives on active learning and collaboration: Javawide in the classroom. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 185–190, New York, NY, USA, 2012. ACM.

[18] E. Kangas and T. Kinnunen. Applying user-centered design to mobile application development. *Commun. ACM*, 48(7):55–59, July 2005.

[19] M. Kapor. Bringing design to software. chapter A software design manifesto, pages 1–6. ACM, New York, NY, USA, 1996.

[20] L. C. Kats, R. G. Vogelij, K. T. Kalleberg, and E. Visser. Software development environments on the web: a research agenda. In *Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software*, Onward! '12, pages 99–116, New York, NY, USA, 2012. ACM.

[21] J. Kollmann, H. Sharp, and A. Blandford. The importance of identity and vision to user experience designers on agile projects. In *Agile Conference, 2009. AGILE'09.*, pages 11–18. IEEE, 2009.

[22] J. Lee and D. McCrickard. Towards extreme(ly) usable software: Exploring tensions between usability and agile software development. In *Agile Conference (AGILE), 2007*, pages 59 –71, aug. 2007.

[23] D. Norman. *The design of everyday things*. Basic books, 2002.

[24] R. Pagulayan, K. Keeker, D. Wixon, R. Romero, and T. Fuller. User-centered design in games. *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 883–906, 2003.

[25] J. Patton. Hitting the target: adding interaction design to agile software development. In *OOPSLA 2002 Practitioners Reports*, pages 1–ff. ACM, 2002.

[26] R. Raymer. Gamification: Using game mechanics to enhance elearning. *eLearn*, 2011(9), Sept. 2011.

[27] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Commun. ACM*, 52(11):60–67, Nov. 2009.

[28] O. Sohaib and K. Khan. Integrating usability engineering and agile software development: A literature review. In *Computer Design and Applications (ICCDA), 2010 International Conference on*, volume 2, pages V2–32 –V2–38, june 2010.

[29] N. Van House, M. Butler, V. Ogle, and L. Schiff. User-centered iterative design for digital libraries: The cypress experience. *D-lib Magazine*, 1996.

[30] P. Zaharias. A usability evaluation method for e-learning: focus on motivation to learn. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1571–1576, New York, NY, USA, 2006. ACM.