Journal of Information Technology Education: Innovations in Practice **Volume 9, 2010**

Assessing Students' Structured Programming Skills with Java: The "Blue, Berry, and Blueberry" Assignment

Xihui Zhang
University of North Alabama, Florence, AL, USA

xzhang6@una.edu

Executive Summary

Java is an object-oriented programming language. From a software engineering perspective, object-oriented design and programming is used at the architectural design, and structured design and programming is used at the detailed design within methods. As such, structured programming skills are fundamental to more advanced object-oriented programming concepts.

Structured programming uses control statements to control the order of execution of a program. In Java, only three forms of control structures are used to implement an algorithm: sequence, selection, and repetition. Sequence structures are a built-in feature: statements run in the order they are listed in the program. Selection structures include single-selections, double-selections, and multiple selections. Repetition structures are implemented in three ways: *while* statement, *for* statement, and *do…while* statement. These simple control structures can be combined into more complex algorithms in only two ways – stacking and nesting.

The purpose of the "Blue, Berry, and Blueberry" assignment is to assess students' structured programming skills with Java. This assignment asks students to write a program that uses all three repetition structures to print, on each line, a number (1 to 100) and a word ("Blue" or "Berry" or "Blueberry", depending on whether the number is divisible by 3 or 5 or 15). The program is also required to report the total numbers of "Blue", "Berry", and "Blueberry".

The assignment was given to undergraduate students enrolled in an advanced object-oriented programming course using Java. The submissions for the assignment were graded using a five-criterion rubric. All in all, the class average score of the assignment was about 4.3 out of 5 (86.0%). The top two issues included formatting the output and counting the number of words.

To evaluate the students' perceptions of the assignment, an online survey was created, including two essay questions. The first question is: "Do you like the 'Blue, Berry, and Blueberry' assignment? Why or why not?" The second question is: "What are the most difficult challenges that you encountered while working on this assignment?"

Each question received 23 responses. In response to question one, more than 90% (21 out of 23) of the respondents said that they did like the assignment, mostly because it helped them to understand the concepts and provided an opportunity to use and compare different forms of selection structures (i.e., *if*, *if…else*, and *if…else if…else* selections)

Editor: Anthony Scime

and repetition structures (i.e., *while*, *for*, and *do…while* loops). In response to question two, most of the 23 students felt that the assignment was not all that difficult once they figured out what was expected. The top three most reported challenges include formatting the output, getting the total count for each word, and checking numbers divisible by 15 first. In summary, results from the instructor's grading report and student perceptions indicate that this assignment is effective for assessing students' structured programming skills with Java.

Keywords: Structured Programming, Object-Oriented Programming, Java.

Introduction

Java is an object-oriented programming language. Structured programming skills, however, are fundamental to more advanced Java topics. Compared with structured design and programming, object-oriented design and programming provides a more natural and intuitive way to describe the real world objects by creating classes and their associated instances (also called objects). The relationship between the two, from a software engineering perspective, is as follows: object-oriented design and programming is used at the high level (architectural) design; in contrast, structured design and programming is used at the low level (detailed) design within methods. As such, mastering structured programming skills is a stepping stone for anyone who wants to become proficient in object-oriented programming.

Structured programming uses control statements (e.g., selection statements and repetition statements) to control the order of execution of a Java program. A typical Java programming textbook (e.g., Deitel & Deitel, 2009) introduces these control statements in early chapters. Without a thorough understanding or a solid grip of the structured programming concepts and skills, students would quickly become confused and frustrated, leading to early withdrawals or ultimate failures.

Structured programming skills with Java include using three forms of control structures – sequence, selection, and repetition (see details in the section on Structured Programming with Java). Effective assessment of whether students have obtained these skills has been a challenging task because there are many control statements and many different ways to combine them.

This article describes the "Blue, Berry, and Blueberry" assignment, which can be used to effectively assess students' structured programming skills with Java. It includes a review of the history of structured programming, an overview of structured programming with Java, the specification of the program, the course background as well as the grading rubric and report, and the students' perceptions about the assignment.

The History of Structured Programming

The idea of structuring a program with nested procedures or subroutines is not new (Wilkes, 1976). In 1949, Wheeler invented the term "closed subroutine" (Wilkes, 1976). In 1951, Wilkes, Wheeler, and Gill started to promote the "master routine calling subroutines and the subroutines calling other subroutines" style of programming, which could be regarded as a primitive form of structured programming (Wilkes, 1976). In 1968, Dijkstra published the "*Goto statement considered harmful*" article, which advocated abolishing the *goto* statement from all "higher level" programming languages. In 1972, Dijkstra published the "*Notes on structured programming*" article, advocating restricting program control to three structures: sequence, selection, and repetition. Dijkstra's work "triggered the structured programming movement" (Chen, 2002, p. 21) and "fundamentally altered human expectations and achievements in software development" (Mills, 1986, p. 58).

Structured programming advocates a "top-down" approach with a "divide-and-conquer" process to writing programs. In a "top-down" approach, programmers first formulate an overarching view

of the program under development, then divide it into smaller subunits. Each subunit is then refined in greater detail and if necessary, divided into even smaller sub-subunits. The previous step repeats until the specification of each and every lowest subunit can be easily translated into simple programming language syntaxes.

It is claimed that this "top-down" approach leads to "a disciplined method of programming with the following advantages" (Wilkes, 1976, p. 274): (1) The program, being modular in nature, is easy to understand and check. (2) There is a possibility of proving it correct. (3) It is easy to maintain and modify. By comparing two sets of program profiles, one set with and the other without using structured programming techniques, Elshoff (1977) shows that programs developed with structured programming techniques (1) have a more restrained flow of control, (2) are more readable and understandable, and (3) are expected to have a much longer lifetime. Indeed, over the years, we have learned that "structured programming produces programs that are easier than unstructured programs to understand, test, debug, modify and even prove correct in a mathematical sense" (Deitel & Deitel, 2009, p. 184).

Structured Programming with Java

In Java, by following Bohm and Jacopini (1966) to promote simplicity, only three forms of control structures are used to implement an algorithm: sequence, selection, and repetition. They are all single-entry/single-exit control statements, i.e., there is only one way to enter and only one way to exit each control statement. Deitel and Deitel (2009) provide a review of control structures which can be summarized as follows.

Sequence structures are a built-in feature: statements run in the order they are listed in the program. Selection structures include single-selections (e.g., *if* statement), double-selections (e.g., *if…else* statement), and multiple selections (e.g., *switch* statement). Repetition structures are implemented in three ways: *while* statement, *for* statement, and *do…while* statement.

It is easy to prove that (1) any form of selection can be implemented by combining *if* statements and (2) any repetition implemented by *for* statements or *do…while* statements can be implemented by *while* statements. Therefore, any form of control ever needed in a Java program can be expressed in terms of: sequence, *if* statement for selection, and *while* statement for repetition. And these simple control structures can be combined into more complex algorithms in only two ways – stacking (i.e., placing control statements one after another) and nesting (i.e., placing one control statement inside another).

In summary, Java programming language uses multiple forms of control structures, each control structure can be implemented by different statements, and all of them can be combined by either stacking or nesting or both. This may cause problems and uncertainties for students. For instance, students may have difficulties to decide which control structures or statements to use, how to combine them, and how to avoid logical errors. The purpose of the "Blue, Berry, and Blueberry" assignment is to provide students an opportunity to better understand and practice these structured programming concepts and skills with Java. Both stacking and nesting are needed to complete this assignment, as specified in the next section.

The "Blue, Berry, and Blueberry" Assignment

The assignment specification: Write a program that uses a *while* loop to print the integers 1 to 100, one number on each line, and set the number's field width as 3. When a number is divisible by 3, print "Blue" on the same line, and separate them by a space. Follow the same format, when a number is divisible by 5, print "Berry"; and when a number is divisible by 15, print "Blueberry". Note that a number divisible by 15 will also be divisible by 3 and 5. In this case, only "Blueberry" should be printed. A message, "The following is from a *while* loop," should be

printed on the first line. Three messages, one on each line, should be printed to report the total numbers of "Blue," "Berry," and "Blueberry" respectively. Finally, in the same program, add a *for* loop and a *do…while* loop: each accomplishes the same task as the *while* loop.

Learning objectives: Upon successful completion of the assignment, students will be able to: (1) analyze, design, and implement solutions to complex logical and computational problems, (2) understand the similarity and difference among the syntaxes and usages of *while*, *for*, and *do...while* loops, (3) choose among available selection statements those that are more appropriate to the situation, and (4) stack and nest selection and repetition control structures to solve problems more effectively.

Course Background, Grading Rubric, and Report

The assignment was given to undergraduate Computer Information Systems (CIS) and Computer Science (CS) students enrolled in an advanced object-oriented programming course using Java (comparable to a CS2 course). These students are generally upperclassmen who are required to have completed an introductory VB.NET programming course (comparable to a CS1 course) that focuses exclusively on structured programming concepts. These students are required to pass the introductory course with a grade of C or higher.

The contents covered in this advanced object-oriented programming course using Java can be roughly divided into three major groups: (1) structured programming such as control statements, methods, arrays, and collections, (2) objected-oriented programming such as inheritance and polymorphism, and (3) others such as exception handling, strings and characters, files and streams, and accessing databases with JDBC. The contents of this course are comparable to the contents suggested by Pendergast (2006). The "Blue, Berry, and Blueberry" assignment was given to the students after structured programming constructs were reviewed with Java syntax, but before object-oriented concepts were presented. Students had been given three other less complicated assignments before this one was introduced, focusing on practicing with input/output statements, selection structures, and repetition structures, respectively. The students were required to complete the assignment within a week, in or outside of a computer lab. The lab is a regularly scheduled part of the course, once every two weeks for one and a half hours.

The Appendix contains an example of code for the assignment that will get a perfect score. To simplify grading and ensure consistency, a grading rubric was developed for the assignment. The submissions for the assignment were assessed using five criteria, including (1) program compiles and runs, (2) all three loops are used, (3) word counts (e.g., number of Blues) are correct, (4) all numbers (i.e., 1-100) are shown, and (5) format (e.g., field width for the number is 3) is correct. For each criterion, either 1 (correct) or 0 (incorrect) was assigned.

The class average score of the assignment is about 4.3 out of 5 (86.0%). The top two issues include formatting the output and counting the number of words. Specifically, for the formatting criterion, 11 out of the 23 students did it incorrectly. In most cases, students added two spaces in front of a number instead of using "%3d" to set the number's field width. Apparently, students had not fully understood the meaning of field width, which was quite surprising. For the word counts criterion, 4 out of the 23 students failed to count the number of words correctly because they forgot to reset the counters back to zeros before executing next loop.

Student Perceptions

To evaluate the students' perceptions of the assignment, an online survey was created using services provided by SurveyMonkey.com. This ensures anonymity so that students would be more comfortable in sharing their thoughts and perceptions. The survey included two essay questions. The first question is: "Do you like the 'Blue, Berry, and Blueberry' assignment? Why or why

not?" The second question is: "What are the most difficult challenges that you encountered while working on this assignment?" The survey link was distributed by course announcement through the school's Blackboard Learning System.

Each question received 23 responses**.** In response to question one, more than 90% (21 out of 23) of the respondents said that they did like the assignment, mostly because it helped them to understand the concepts and provided an opportunity to use and compare different forms of selection structures (i.e., *if*, *if…else*, and *if…else if…else* selections) and repetition structures (i.e., *while*, *for*, and *do…while* loops). Some representative comments are as follows:

- I liked the assignment because it helped me understand the differences between the "for", "while", and "do-while" loops.
- Yes, I liked it because it made us use all the repetition statements. It made sure that we had a good idea of what to do when using all of the statements.
- I liked the Blueberry assignment as it gave us a chance to make a program using while, do… while and for loops. And I got to understand them in detail and how these loops work.
- Yes, I like the assignment. This assignment gives a very good practice of loops. You won't be able to do it without a thorough knowledge of loops.
- I liked the assignment. I liked that it combined 3 concepts (for, while, and do while loops) into one assignment. I felt like I learned a lot about the similarity and difference between all these because I was using them all to accomplish the same goal.

Two students didn't like the assignment: One thought it was too hard, and the other thought he/she had done a lot of work for it but did not feel he/she had accomplished anything useful.

In response to question two, most of the 23 students felt that the assignment was not all that difficult once they figured out what was expected. The top three most reported challenges include formatting the output, getting the total count for each word, and checking numbers divisible by 15 first. Some of the representative comments are as follows:

- It may just be something I overlooked but I had a hard time getting the numbers and words to print on the same line correctly.
- I had trouble getting the program to loop through and count how many times each were used.
- The most difficult part to me was getting the words to print out in the correct format.
- The biggest challenge for me was to figure out how to count the total number of Blues, Berries, and Blueberries.
- The most difficult challenge was to find out how 15 will be checked first so that it does not come again while we are checking division for 3 and 5, given that a number divisible by 15 is also divisible by 3 and 5.

Specifically, five of the 23 respondents admitted that formatting the output according to the requirements had been somewhat challenging; three of the 23 respondents acknowledged that they had trouble counting the total numbers of each word accurately (i.e., Blue, Berry, and Blueberry); and three of the 23 respondents stated that it took them quite a while to figure out how to check whether a number was divisible by15 first. This is consistent with the instructor's grading report.

Conclusions

This article described the "Blue, Berry, and Blueberry" assignment, which was used to assess students' structured programming skills with Java. The article reviewed the history of structured programming and provided an overview of structured programming with Java. It provided the assignment specification as well as the course background, grading rubric and report. It also collected survey data for evaluating student perceptions regarding the assignment. Results from the instructor's grading report and student perceptions showed that the assignment was effective for assessing students' structured programming skills with Java.

References

Bohm, C., & Jacopini, G. (1966). Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM, 9*(5), 366-371.

Chen, P. P. (2002). From goto-less to structured programming: The legacy of Edsger W. Dijkstra. *IEEE Software, 19*(5), 21.

Dijkstra, E. W. (1968). Goto statement considered harmful. *Communications of the ACM, 11*(3), 147-148.

Dijkstra, E. W. (1972). Notes on structured programming. In O. J. Dahl, E. W. Dijkstra, & C. A. R. Hoare (Eds.), *Structured programming* (pp. 1-82). New York, NY: Academic Press.

Deitel, P., & Deitel, H. (2009). *Java how to program* (8th ed.). Upper Saddle River, NJ: Prentice Hall.

Elshoff, J. L. (1977). The influence of structured programming on PL/I program profiles. *IEEE Transactions on Software Engineering, 3*(5), 364-368.

Mills, H. D. (1986). Structured programming: Retrospect and prospect. *IEEE Software, 3*(6), 58-66.

Pendergast, M. O. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education, 5*, 491-515. Retrieved from http://www.jite.org/documents/Vol5/v5p491-515Pendergast128.pdf

Wilkes, M. V. (1976). Software engineering and structured programming. *IEEE Transactions on Software Engineering, 2*(4), 274-276.

Wilkes, M. V., Wheeler, D. J., & Gill, S. (1951). *The preparation of programs for an electronic digital computer*. Cambridge, MA: Addison-Wesley.

Appendix. The Code

```
// Blueberry.java
// Display numbers and their corresponding words
// Using selection and repetition structures
public class Blueberry
{
        public static void main ( String [] args )
        {
                int number = 1; // starting number
                int blueCount = 0; // counter of "Blue"
                int berryCount = 0; // counter of "Berry"
                int blueberryCount = 0; // counter of "Blueberry"

                // display message for the while loop
                System.out.print( "The following is from a while loop: \n" );

                // display numbers and their corresponding words using while loop
```

```
while ( number <= 100 )
{
        // display the number and "Blueberry" when it is divisible by 15
        if( number % 15 == 0 )
        {
                System.out.printf( "%3d %s\n", number, "Blueberry" );
                blueberryCount++;
        }
        // display the number and "Blue" when it is divisible by 3
        else if ( number % 3 == 0 )
        {
                System.out.printf( "%3d %s\n", number, "Blue" );
                blueCount++;
        }
        // display the number and "Berry" when it is divisible by 5
        else if ( number % 5 == 0 )
        {
                System.out.printf( "%3d %s\n", number, "Berry" );
                berryCount++;
        }
        // otherwise, only display the number
        else
        {
                System.out.printf( "%3d\n", number );
        }

        number++;
} // end while loop

// display the total numbers of "Blue", "Berry", and "Blueberry" respectively
System.out.printf( "There are %d Blues.\n", blueCount );
System.out.printf( "There are %d Berries.\n", berryCount );
System.out.printf( "There are %d Blueberries.\n", blueberryCount );

// display message for the for loop
System.out.print( "The following is from a for loop: \n" );

// reset all counters
blueCount = 0;
berryCount = 0;
blueberryCount = 0;

// display numbers and their corresponding words using for loop
for ( number = 1; number <= 100; number++ )
{
        // display the number and "Blueberry" when it is divisible by 15
        if( number % 15 == 0 )
        {
                System.out.printf( "%3d %s\n", number, "Blueberry" );
                blueberryCount++;
        }
```

```
                // display the number and "Blue" when it is divisible by 3
                else if ( number % 3 == 0 )
                {
                        System.out.printf( "%3d %s\n", number, "Blue" );
                        blueCount++;
                }
                // display the number and "Berry" when it is divisible by 5
                else if ( number % 5 == 0 )
                {
                        System.out.printf( "%3d %s\n", number, "Berry" );
                        berryCount++;
                }
                // otherwise, only display the number
                else
                {
                        System.out.printf( "%3d\n", number );
                }
        } // end for loop

        // display the total numbers of "Blue", "Berry", and "Blueberry" respectively
        System.out.printf( "There are %d Blues.\n", blueCount );
        System.out.printf( "There are %d Berries.\n", berryCount );
        System.out.printf( "There are %d Blueberries.\n", blueberryCount );

        // display message for the do...while loop
        System.out.print( "The following is from a do...while loop: \n" );

        // reset the starting number and all counters
        number = 1;
        blueCount = 0;
        berryCount = 0;
        blueberryCount = 0;

        // display numbers and their corresponding words using do...while loop
        do
        {
                // display the number and "Blueberry" when it is divisible by 15
                if( number % 15 == 0 )
                {
                        System.out.printf( "%3d %s\n", number, "Blueberry" );
                        blueberryCount++;
                }
                // display the number and "Blue" when it is divisible by 3
                else if ( number % 3 == 0 )
                {
                        System.out.printf( "%3d %s\n", number, "Blue" );
                        blueCount++;
                }
                // display the number and "Berry" when it is divisible by 5
                else if ( number % 5 == 0 )
                {
```

```
                    System.out.printf( "%3d %s\n", number, "Berry" );
                    berryCount++;
                }
                // otherwise, only display the number
                else
                {
                    System.out.printf( "%3d\n", number );
                }

                number++;
            } while ( number <= 100 ); // end do...while loop

            // display the total numbers of "Blue", "Berry", and "Blueberry" respectively
            System.out.printf( "There are %d Blues.\n", blueCount );
            System.out.printf( "There are %d Berries.\n", berryCount );
            System.out.printf( "There are %d Blueberries.\n", blueberryCount );
        } // end main
} // end class Blueberry
```

Biography

Xihui Zhang is an Assistant Professor of Computer Information Systems in the College of Business at the University of North Alabama. He earned a Ph.D. in Business Administration with a concentration on Management Information Systems from the University of Memphis. His teaching and research interests include technical, behavioral, and managerial aspects of Information Systems. He has published numerous articles in refereed journals and conference proceedings. He serves on the Editorial Review Board for several academic IS journals such as *Journal of Computer Information Systems* and *Journal of Information Technology Education*. Additional information about him can be found at http://sites.google.com/site/xihuizhang/.