American College of Thessaloniki
November 2024

CS105 Course Project Report

Fall 2024 – Computer Science 105

Student: Petar Velkovski

Instructor: Dr. Alexandar Astaras

Contents

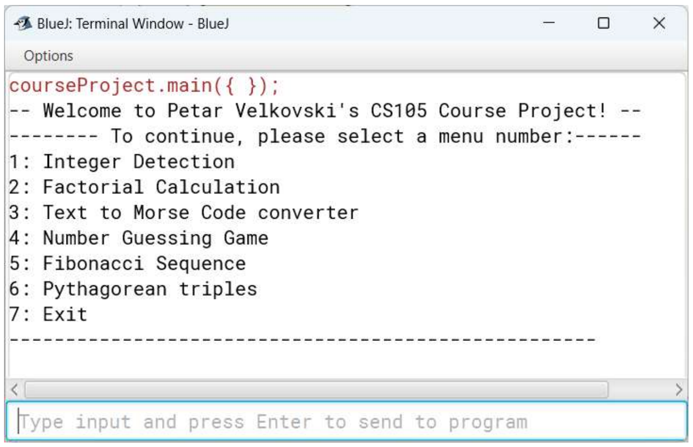
Introduction

For the final project of my CS105 class I was asked to create a seven-option program with a text user interface menu. The seven options needed to be 1: Integer Detection, 2: Factorial Calculation, 3: Morse Code translator, 4: A Number Guessing game, 5: A Fibonacci sequence generator, 6: Pythagorean triples calculator and 7: Exit. User input error trapping was implemented in all of the subprograms and the main menu using try... catch statements. This project was assigned to me on November 6th 2024, and I finished coding it on the November 26th. I spent an estimated 17 hours of coding and 3-4 hours of additional planning and researching.

Structure and Methodology

The main function itself is only a method call to the menu() method. I did this in order to be able to call the menu method from other parts of the program without having issues. I made a decision to use methods and method calling in the menu itself, as well as to implement a separate method named goingback() that executes at the end of all subprograms, in order to give the user the option to go back to the menu, or to exit the program. I used an online resource to study and learn about how to use methods and how to correctly implement them in my program. [1] The subprograms all function without logical errors or functional flaws. All of them have some form of invalid input error handling.

User Interface


BlueJ: Terminal Window - BlueJ
Options

```
courseProject.main({ });
-- Welcome to Petar Velkovski's CS105 Course Project! --
-------- To continue, please select a menu number:------
1: Integer Detection
2: Factorial Calculation
3: Text to Morse Code converter
4: Number Guessing Game
5: Fibonacci Sequence
6: Pythagorean triples
7: Exit
-----------------------------------------------------
```

Type input and press Enter to send to program

Figure 1 - Screenshot of the user interface of the main menu given from the terminal window of BlueJ.

Flowchart

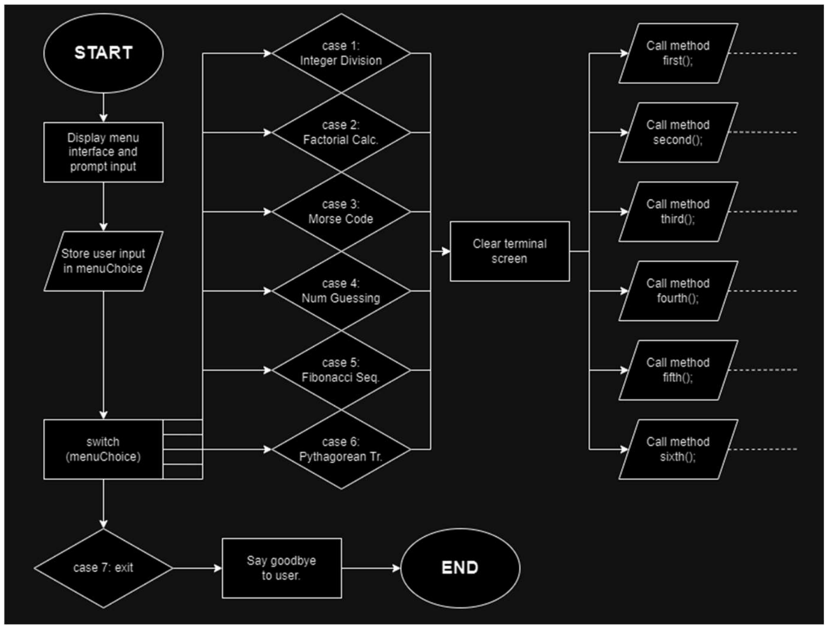


Figure 2 A top-level flowchart that shows the function of the main menu.

1 – Integer Division

As the first and one of the simplest subprograms I was assigned to code, I finished this in the same day the project was assigned. This method does not request input from the user, it uses predefined and preinitialized variables to control a while loop. The while loop counts a variable up to a limit (2000) and in every loop it checks whether the counter is perfectly divisible by all integers from 1 to 7 (including) without giving a remainder. This loop functions on 7 statements connected with the OR Boolean operators, ensuring that all 7 statements will be fulfilled. If all of the statements are true, i.e. the given number is divisible by all integers 1 through 7, the if block executes, printing out the number, adding it to a sum variable, adding to a variable keeping track of the amount of numbers found. After finishing the while loop until 2000, the method prints out the amount of numbers found, their sum and their average. As all other methods, it finishes with calling the method goingback() in order to present the user with a choice to either close the program or go back to the menu.

2 – Factorial Calculation

This subprogram asks the user to input 5 integers between 0 and 12 and calculates the factorial of the integer which is the closest to the group's average. I started by defining the InputStreamReader and BufferedReader as the input system that I would need to use. The main algorithm to do the calculations is a do-while loop that prompts the user to enter a number, parses the input as a double, and does an if-else check to see whether the number is in range. If it is, it gets added to a temporary sum and the amount of numbers inputted gets increased by 1. The while statement for the do-while loop is until the amount of valid numbers is less than or equal to 5. When the do-while loop finishes successfully, the average is calculated and printed, and then a for loop executes that calculates the factorial of the number closest to the average. As all other programs, this one ends by calling the goingback() method.

3 – Morse Code Calculator

This subprogram may have been the one that took me the most time to create and plan, as it uses hash tables, a data structure the CS105 course material does **not** cover, and I had to learn how to use them by myself. I do realize there may have been easier ways to do this, however I chose to do this as I believe using hash tables would be the most efficient way. The method starts with defining the InputStreamReader and BufferedReader, as we expect input, and after an introduction to the user, the method defines two hash tables with strings as the keys and the values. This is because both the alphabet set and the morse values need to be expressed with strings. Using Hashtable.put(key, value) I populate both hash tables used for converting from text to morse and vice versa. After that, a do-while loop executes prompting the user for an input, and the input lines are contained within a try... catch statement, in order to catch non-numeral user input errors.

After the user chooses a conversion type, an if-else statement executes with three options: 1 – Text to Morse, 2 – Morse to text, and anything else just gets flagged as invalid input.

The text to morse converter functions by reading the input for the text, storing it in a string, and creating a char array from the string. A string builder is defined for the morse translation. After that, a for each loop executes, counting the amount of characters in the char array, and an if-else block appends the correct character from the hashtable using the lookup command TTMorse.get(char). If the character doesn't exist in the predefined hashtable, the loop appends "//" to indicate an invalid character. After the loop, the string builder is printed.

The morse to text converter uses a do while loop to constantly ask for morse values which get appended to a string builder, until one of the sentinel values "end" or "-1" get entered, after which the loop gets broken, and the translation is printed.

As all other methods this one finishes with calling the goingback() method.

4 – Number Guessing Game

The entirety of the Number Guessing Game is just a do-while loop which reads an integer, checks whether it is correct with the random number generated, and if not, whether it is higher or lower. It tells the user this information and counts the guesses. It has 4 custom messages for the amount of tries the user needs in order to guess it. As all other methods this one finishes with calling the goingback() method.

5 – Fibonacci Sequence

This method took me a while to do as I decided to do my own research on which algorithm to use to calculate the Fibonacci sequence, as well as fulfil the requirements given. The block of code handling input catches non-positive inputs and inputs that aren't numbers and gives an appropriate response. I used a for loop for the first part, to calculate the needed Fibonnaci number of the inputted term, and then print it. After that, I reset the same variables and use them in a while loop to print out the entire sequence. This is not the most effective way to calculate it, as more sophisticated algorithms use arrays to do this quickly. [2]

However, I did not yet know how to properly use arrays, but if I were to code this portion of the project anew, I would use the array method cited. As always, this method finishes with calling the goingback() method.

6 – Pythagorean Triples

This last method works by automatically calculating Pythagorean Triples up to a set limit (120). I used a two dimensional array for the triples, and Booleans to check for duplicates and multiples. A first block of three for loops generate all possible triples, and after that, a second block of code checks for duplicates, multiples and prints the triples. Temporary intermediate variables are used twice in this second block of code, in order to simplify and make the comparisons in the if statements easier. Unique triples are meant to stand out in the printed list, so they are printed with a triple indent. After finishing, the method ends with calling the goingback() method, as always.

Conclusion

To take everything into account, this course project was a nice challenge, and enough time was given, in order to plan and code all of the entire project. I feel satisfied with my performance, as it is the largest project I have coded by far. I learned new things, like hash tables, arrays and properly using methods and method calling. It did give me a glimpse of my potential future as a software engineer, especially considering that continuous learning and creating is a crucial part of this career and industry.

References

[1] W3Schools, "Java Methods," W3Schools, 2024. [Online]. [Accessed 11 2024].

[2] C. Cubukcu, Z. B. G. Aydin and R. Samli, "Comparison of C, Java, Python and Matlab programming languages for Fibonacci and Towers of Hanoi algorithm applications," *Boletim da Sociedade Paranaense de Matemática,* vol. 41, p. 3, 2022.

[3] D. A. Astaras, "CS105 Course Lecture Slides," Thessaloniki, 2018.

[4] K. Sierra and B. Bates, Head First Java, O'Reilly Media, 2005.

[5] Javatpoint, "Java StringBuilder Class," Javatpoint, 2024. [Online]. Available: https://www.javatpoint.com/StringBuilder-class. [Accessed 11 2024].