Jason Yao CS 203

Ward 21/01/14

**Fortune Teller Report**

**Introduction & Use**

The Java Fortune Teller is program written in Java to gather input from a user, and to then use the information to output a “lucky number” and a customized fortune. To start it, a user should open the file in BlueJ, compile the program, and then start the program.

To compile the program, double-click on the Fortune class in BlueJ, and then click the Compile button on the top-left hand side of BlueJ. Exit out of the editor once the program has compiled.

In order to start the program, simply right-click the Fortune class in BlueJ, click the void main(String[] args) option, and then click ok. The program will start, and the user will simply need to follow the prompts that the program gives, by typing an answer, and then then hitting the return key.

**Program Description**

The program code is separated into “blocks” of code, with each block commented to show the purpose. The code blocks will be in the sequence that the user will see the output, and is designed to save the user’s input, then utilize the saved input to calculate a “lucky number” using the basic four operators (+, -, /, \*) and all of the numerical answers given by the user.

An important pieces of the program that should be noted are the while functions used, which were implemented in order to “catch” any illegal user input that either did not make logical sense, such as a GPA of 6.

|  |
| --- |
| while (gpa > 4)  {  System.out.println("Now, I know you think you're so smart, trying to trick me.");  System.out.print("But seriously, put in your actual GPA this time.");  gpa = keyboard.nextDouble();  } |

Some of the key algorithms implemented include:

-The Scanner class, which allowed the program to take a user’s keyboard input, which created a scanner object that captured keyboard input.

|  |
| --- |
| Scanner keyboard = new Scanner(System.in); |

-The print class, which allowed the program to display text to the user, allowing the program to give prompts to the user on the next needed keyboard input.

|  |
| --- |
| System.out.print("Hi"); System.out.println("Hi"); |

-variables and String declarations such as int, byte, double, String, and float, which were initialized to the value captured from a user’s keyboard. Numerical initializations utilized the int, double and float variables, in order to show how to use each one, while text-based answers were saved into Strings that could save words or phrases to memory.

When all the answers were gathered and stored in memory, the program will calculate a “lucky number” based on the following formula:

|  |
| --- |
| // The +0.5 is to round the subtotal, instead of truncating  int luckynum;  luckynum = (int)(monies/(relAge+(age-gamer))\*gpa+0.5);  int finalnum = Math.abs(luckynum); |

In which the initial number is calculated by “casting” the following operations, in which the user’s annual income (monies), oldest living relative’s age (relAge) , user’s age (age), the number of video games the user plays (gamer), and the user’s GPA (gpa) is used. An interesting thing to note is that in order to remove negative “lucky” numbers, the Math.abs(); function was used, in order to take the absolute value of the semifinal lucky number, and convert it to a positive value.

At the end, the program will thank the user by name, by recalling the String name, and printing out the message with System.out.print();.

**Testing & Evaluation**

Testing Method: The Java program was constantly tested after each block of code to see if the code was implemented correctly- if not, and errors were detected, the program was stopped, and then the code evaluated again to determine the cause of the error.

The process of testing would go as such: Code -> Compile -> Compiling error -> fix -> compile -> Run

Whenever the program ran into compiling errors, the section with the error would be checked to see what the issue was, and whether the issue could be fixed or not.

A problem that the program ran into was the issue of saving user input into Strings using the keyboard.nextLine(); command, which would end up skipping a line, due to the way the command puts a “marker” in the memory. In order to get around this, keyboard.nextLine(); had to be written above any future uses of the keyboard.nextLine(); command, in order to give the command a blank space to put a marker, resolving the issue.

The majour issue of this assignment was learning how to code a program, since it is a new environment that requires some time to acclimate to.

**Conclusion**

The most challenging aspect of coding this program was making sure that all used syntax was correct, remembering things such as the need to declare variables before using them, and not bang your head on the computer if the code does not work.

Some of the lessons I learned from completing the project is a healthy respect for programmers who have to code exceedingly difficult to understand things, just because of the amount of errors I was getting on this simple program was enough to make me want to toss my computer out the window after a while. Besides that, and learning patience, I’ve finally learned to code my first program, which is actually a really nice feel, especially near the end, when it was simple bug-fixing, and I had the feeling of knowing what I had to do.