Wentworth Institute of Technology
COMP1000 – Computer Science I
Fall 2017, Kreimendahl

Programming Assignment 4

In this assignment you are to write programs to solve the following problems. As with all assignments, remember the following submission steps:

- Make sure your code passes at least all the provided JUnit tests
- Create and test Javadoc code documentation
- Save, commit, and push all code changes
- Confirm the latest code is visible via the "Files" section of your repository website
- Confirm that the repository is private, and that the instructor has Developer access

Not that for each problem, some JUnit tests that will be used for grading have not been provided as a part of the starter code. It is your responsibility to thoroughly test your code.

Problem a (`PA4a.java`)

Write a program that deals with inflation, which is essentially the rising cost of general goods over time. That is, the price of goods, such as a packet of peanuts, goes up as time goes by. So, you will write a program to gauge the expected cost of an item in a specified number of years. The program asks for the cost of the item, the number of years, and the rate of inflation. The output is the estimated cost of the item after that number of years, using the given inflation rate. The user enters the inflation rate as a percentage, for example 4.5. You will have to convert the percentage to a fraction (like 0.045), and then use a loop to estimate the item's price adjusted for inflation. Note that this is similar to computing compound interest on a credit card account or a mortgage. Also note that you must check each of the values provided by the user to make sure that they are reasonable. Finally, you have to print out the price with exactly two places after the decimal (for the cents) after your calculations are done

To adjust the price for inflation, you need to increase the price by the inflation rate each year. For example, if you have an item that is initially $10, with inflation rate of 10%, the adjusted prices will be:

- After 1 year: $10.00 ∗ (1 + 0.10) = $11.00
- After 2 years: $11.00 ∗ (1 + 0.10) = $12.10
- After 3 years: $12.10 ∗ (1 + 0.10) = $13.31

  …

In other words, to calculate the price after another year, you have to use the value from the current year, NOT the original price. To do this, you must use a loop. An example of what your program should output:

```
Enter the current price of the item: $10
Enter the number of years: 3
Enter the inflation rate as a percentage: 10
After 3 years, the price will be $13.31
```

You have been supplied JUnit tests for some simple valid examples, as well as negative price, negative year, and negative interest rate.

Problem b (`PA4b.java`)

Write a program that plays a guessing game with the user. Specifically, your program should randomly pick a number between 1 and 100. Then, ask the user for a guess. You should detect and tell the user if the guess is not a valid guess. Otherwise, tell the user their guess was too high or too low. The program should continue to prompt the user for new guesses until they get the correct number, telling them each time if the guess was too high or too low or invalid.

You have been supplied code to pick a random number between 1 and 100 each time you run your program. Here are a couple development/debugging strategies for this "target" variable:

- Print out the random number, to make sure your program is acting correctly – remember to remove/comment this before running unit tests/submitting.
- Temporarily set the random "seed" to a value, which will have the effect of always choosing the same random number – the unit tests have fixed seeds that you can use with known outcomes.
- Temporarily set the "target" variable to a fixed number, so you can test to see how your program responds in different testing situations.

Here's a sample run of a working version of the program:

```
Enter your guess (between 1 and 100): 50
Too high!
Enter your guess (between 1 and 100): 0
Invalid guess, try again!
Enter your guess (between 1 and 100): 101
Invalid guess, try again!
Enter your guess (between 1 and 100): 25
Too high!
Enter your guess (between 1 and 100): 12
Too high!
Enter your guess (between 1 and 100): 6
Too high!
Enter your guess (between 1 and 100): 3
Too low!
Enter your guess (between 1 and 100): 4
Too low!
Enter your guess (between 1 and 100): 5
You win!
```

You have been supplied with JUnit tests for a set of lucky guesses (i.e. the user immediately guesses the right answer), lucky guesses starting with invalid guesses, and a full game, including both invalid and valid guesses.