School of Computing Sciences

CMP-4008Y — Programming 1

LAB EXERCISES 4 — Selection and Repetition

Level of difficulty: (*) - easy; (**) - intermediate; (***) - decidedly tricky. Questions do not necessarily require the use of a computer. It is a good idea to read through the exercises before the laboratory session.

1. (*) You are required to implement a "guess the magic number" game. The program should generate a random magic number between 1 and 10. The user is then allowed three guesses to find the magic number, and should be prevented from entering a guess less than 1 or higher than 10. After each guess, the program should inform the user if their guess was too high, too low or correct. The user wins the game if they guess the magic number within the three guesses allowed. The program should allow the user to play one or more games, asking whether they would like another go after each game.

   Hints:

   - As we have seen, the `java.util.Random` class of the Java standard library implements a method called `nextInt()` that returns a different random integer value each time it is called.
   - This programming project is a more complicated than previous assignments. You should tackle it in small steps, testing thoroughly after each step. Make sure you have done the "problem solving" for each step, and have a clear picture in your mind of what you want the computer to do *before* attempting the coding. I would recommend a strategy along the following lines:

     (a) Start by writing a program that displays 100 random integers between 1 and 10 using the `rand` function.

     (b) Next, write a program that generates a random number between 1 and 10, and allows the user to make a guess at the value of this number. If the user's guess is too high, the program should print "too high!", if too low, the program should print "too low!", and if the user guessed the magic number correctly it should print "Correct!".

     (c) Modify the program using a loop so that the user is prevented from entering a guess less than 1 or greater than 10.

     (d) Modify the program, again using a loop construct, so that the user is allowed to make three guesses to find the magic number. The loop should terminate early if the user guesses correctly before the third guess.

     (e) Use an if-else statement to print "Well done, you have won!" if the user successfully guessed the magic number within the three guesses allowed, and "Bad luck, you have lost!", followed by the magic number, otherwise.

     (f) Modify the program so that the user can play more than one game. The user should be asked whether he or she would like another go after each game.

- The basic specifications can be implemented elegantly using one `for` loop, one `do-while` loop, one `while` loop, an `if-else` statement and one multi-way `if-else` statement. This project therefore gives an opportunity to display your understanding of all of the basic control structures.
- There are plenty of opportunities to extend the program, for instance you could use static methods or introduce different difficulty levels (say 4 guesses instead of 3), or computing the average success rate.

2. (**) It is possible to determine if a string is palindromic (it is the same read forwards as it is read backwards, e.g. "kayak") using the methods provided by `StringBuilder` objects. It is also possible to determine whether a string is palindromic using repetition and selection constructs. Write a program to determine whether a string is palindromic using both methods, and consider the advantages and disadvantages of each approach.

3. (***) Automated testing is very useful in software development, as automated tests are documented and repeatable, unlike manual testing. If you modify the code, you can easily re-run the automated tests to check that your modification has not introduced a new bug (known as a "*regression*"). Write a program to test your solutions to the previous task using a large number of randomly generated strings that are known by construction to be palindromic or non-palindromic.

4. (***) How would you iterate over all the Unicode characters in the following String?

```
1  String text = "penguin \uD83D\uDC27\uD83D\uDE42";
2
```

Gavin Cawley
12th October 2024