School of Computing Sciences

# **CMP-4008Y** — Programming 1

## **LAB EXERCISES** 2 — Primitive Data Types, Operators, Expressions and Casts

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. (*) `Math.random()` is a static method in Java that returns a random number from zero up to but not including one, of type `double`. So the line

   ```
   int i = (int)(10 * Math.random() + 1);
   ```

   should result in a random integer from one to ten (we will see a different way of generating random numbers in lecture 5). Write a program that rolls an imaginary dice ten times, and output the results. The line

   ```
   System.out.println(i);
   ```

   can be used to print out the variable `i`. This can be implemented using a `do-while` loop, which we have already seen in the lectures, but a `for` loop (which we will see in lecture 7) would be neater.

2. (*) Modify your program so that it rolls a twenty sided dice and continues until a twenty is rolled, displaying each result. Make sure your program outputs the final roll of twenty.

3. (*) Given the declarations `double a = 7.5, b = 5.25, c = 1.25`, use brackets and casts to modify the expression `a + b/c`, such that it evaluates to:

   (a) 12.5

   (b) 10.2

   (c) 11.7

   (d) 11.2

   (e) 11.5

   this may be achieved by adding only parentheses, `()`, and type conversions (casts), such as `(int)`.

4. (*) Strings can be concatenated with `+`. What output do you think the following code will produce?

   ```
   System.out.println(10+2+3);
   System.out.println("10"+2+3);
   System.out.println("10"+(2+3));
   System.out.println("10"+'2'+'3');
   System.out.println("10"+('2'+'3'));

   ```

5. (*) A triangular number is obtained by adding all the integers from one up to a given number. So the seventh triangular number is $1+2+3+4+5+6+7$. Write a program to calculate and output the first fifty triangular numbers.

6. (**) Modify your program so that it prints out all the *even* triangular numbers less than 2000. Hint: This is another chance to use the `%` operator and an `if` statement, seen in lecture 2 will also be needed.

7. (**) `Math.pow(a, b)` is a method for raising `a` to the power of `b`. Like `Math.random()`, it returns its answer as a `double`. Using this method, write a program that outputs the last digit of $3^{20} - 2^{25}$. (Hint: The answer should be 9.)

8. (**) The program below produces the wrong answer to the previous question. *Why* does it gives the wrong answer? What is the easiest way to modify it to give the correct answer?

```java
int firstPow = (int)(Math.pow(3, 20));
int secondPow = (int)(Math.pow(2, 25));
int lastDigit = (firstPow - secondPow) % 10;

System.out.println("The last digit is " + lastDigit);

```

9. (**) What output do you expect from the following code? Explain why the code fragment behaves as it does.

```java
double d = 2.0/3.0;
float f = 2.0f/3.0f;

if (d == f)
{
   System.out.println(d + " == " + f);
}

if (d != f)
{
   System.out.println(d + " != " + f);
}

```

10. (**) The following code fragment is intended to display the difference between two integers, but it doesn't give the expected answer. Explain the result and modify the code fragment so that it gives the correct answer.

```java
int neg  = -1234567890;
int pos  = +1234567890;
int diff = pos - neg;

System.out.println(pos + " - " + neg + " = " + diff);

```

11. (***) Consider the following code fragment:

```java
int x = 0;

if (x == -x)
{
   System.out.println(x + " == -" + x);
}

```

In this case, it isn't unduly surprising that `0` and `-0`, are equal, and the code fragment will print "`0 == -0`" on the console as expected. There are at least two other values that will result in something being displayed, one of which is obviously `-0`, but what is the other?

12. (**) If the code fragment in the question above were modified so that `x` were declared as a `byte`, rather than an `int`, there would only be two values of `x` that would result in something being displayed on the console, namely `x = 0` and `x = -0`. Explain why that is the case.

13. (***) The maximum `char` value is 0xFFFF (although that value is a non-character in Unicode). What happens if you try a larger number (for example, the slightly smiling face emoticon has the hexadecimal value 0x1F642)?

Gavin Cawley
24th September 2024