Introduction to Computer Science II

# Tutorial 1

## Objectives

Basic Java programming: loops, branching, arrays, Strings, and input/output.

---

**Submit any `.java` files that you modify or create.**

---

## Part One : Sample Code

Read the provided `SampleJavaCodeProgram.java`. Compile and run the program to see how the code works. **It is expeted that everyone will be able to write code like this very soon in the course. You will need to be able to write basic Java code in order to move forward.**

---

## Part Two : Printing Qs

Modify the provided `PrintQs.java` program. The program currently asks the user to enter an integer (call it the number N). Modify the program so that it will draw an NxN grid of Q's. For example, if you enter 7, the program will print to the screen

```
QQQQQQQ
QQQQQQQ
QQQQQQQ
QQQQQQQ
QQQQQQQ
QQQQQQQ
QQQQQQQ
```

There are 7 lines printed and each line has 7 Q's printed (without spaces). Use nested **for** loops and only print a single Q at a time. This example will call `System.out.print('Q')` 49 times and `System.out.print()` 7 times.

Running the program with the value 2 will print

```
QQ
QQ
```

---

## Part 3 : Babylonian Square Roots

In the provided `Babylonian.java` file, complete the static method that computes square roots of a number using the *Babylonian method*.

Wikipedia Link for Babylonian Square Root Method

It is an *iterative* method that keeps making better and better approximations to the square root of a number. The algorithm is terminated when the improvement in successive approximations becomes small.

Pseudocode for the Babylonian method to find the square root of a number N is as follows:

```
1. Let M1 := N/2 be our first guess
2. Let M2 := average of M1 and N/M1
3. If |M1-M2| < epsilon then stop and output M2
4. Otherwise, Let M1 := M2 and go back to step 2
```

Here, `|x|` is the absolute value of x and epsilon is a small positive number (like 0.0001). We use `:=` to denote assignment in the pseudocode above.

Python code for a function that implements this is as follows:

```python
def babylonian(N, epsilon):
   m1 = 0.5*N           # N/2
   m2 = 0.5*(m1+N/m1)   # average of m1 and N/m1
   while abs(m1-m2) >= epsilon:
      m1 = m2
      m2 = 0.5*(m1+N/m1)
   return m2
```

Your task is to translate this (pseudocode or python) into Java and complete the `babylonian()` method. A `main()` method is provided so that you can test your method.

Note that the `Math` class can provide useful functions to help with math. In particular, `Math.abs()` should be used in your code.

Link to Java's Math Class API

Once you have implemented the method properly, modify the `main()` method so that the program repeatedly prompts the user to enter a positive number (or "end", triggering the end of the program). Print the number, its square root using both `Math.sqrt()` and your Babylonian method, and the difference between them (like the original code did for the hard-coded number) if the input number is non-negative. Print a message like "Only enter positive numbers" if the number is negative. Exit the program when the user input is the string "end".

---

## Part Four :

Write a static method

```
public static char[] filter(char[] list, char target)
```

The method will take an array of `char`s as input and another `char`. It will then create a new array that has all the `char`s from the input array (in the same order) except that all instances of `target` are removed. For example,

```
char[] in = {'a', 'b', 'a' 'c', 'd'};
char[] out = filter(in, 'a');
// assert : out == {'b', 'c', 'd'}
out = filter(in, 'b');
// assert : out == {'a', 'a', 'c', 'd'}
```

Note: the `length` of the output array should only be as big as needed.

You can create Java class called `Part4` that has this method in it. Your class should have a `main()` method to test your method.

---

## Reading : The `String` and `StringBuilder` classes

Strings allows us to have text in our programs. They are *immutable* sequences of characters. Explore the API for the `String` class.

Link to Java's String class API

The API (Application Programming Interface) provides you with all methods that are in the given class. Use this to find methods that are equivalent to the following Python string behaviour:

```
word = "kitten "
character = word[4]      # get a single character at index position 4
substring = word[1:3]    # get a substring of length 2
stripped  = word.strip() # remove leading/trailing whitespace
WORD = stripped.upper()      # upper case version of word
WORD == 'KITTEN'         # check if two strings are the same
words = "cat dog eel"
word_list = words.split() # split by whitespace
words = ",".join(word_list) # recombine with comman between words
```

Write a short program whose `main()` method has the above code translated to Java and also prints each to the screen. You can name the class that has your main method anything you wish.

The `StringBuilder` class provides *mutable* sequences of characters. This class is especially useful when you need to iteratively build up or modify a string. Consider the two strings `text1` and `text2` in the following:

```
String text1 = "";
for(int i=0; i<size; i+=1){ //
   text1 = text1 + ">";
}

StringBuilder text_sb = new StringBuilder();
for(int i=0; i<size; i+=1){
   text_sb.append(">");
}
String text2 = text_sb.toString();
```

Both strings have the same characters in them. It is more *efficient* to construct `test2` though (especially if `size` is large). Find and then explore the `StringBuilder` API.

---

---

## Extra Practice :

In COMP 1005/1405, you likely wrote nested loops to draw triangles. Repeat this now using Java. That is, modify your `PrintQs` class so that your program displays an additional 4 triangles based on the input N. For example, if the input was 4, then the following four triangles of Q's should also be displayed on the screen

```
Q
QQ
QQQ
QQQQ

QQQQ
QQQ
QQ
Q

QQQQ
 QQQ
  QQ
   Q

   Q
  QQ
 QQQ
QQQQ
```

---

Create a java program (java class with a `main()` method) that translates the following python code to java. You can name your class whatever you wish for this.

```
def main():
    size = 10
    numbers = [0]*size
    for i in range(size):
        if i < 3:
            numbers[i] = i
        elif i < 6:
            numbers[i] = i + 10
        else:
            numbers[i] = i*10

    for i in numbers:
        print(numbers[i], end=',')

    print(numbers)
```