Introductory Java Programming

Queen Mary
University of London
School of Electronic Engineering
and Computer Science

Course Code: EBU4201

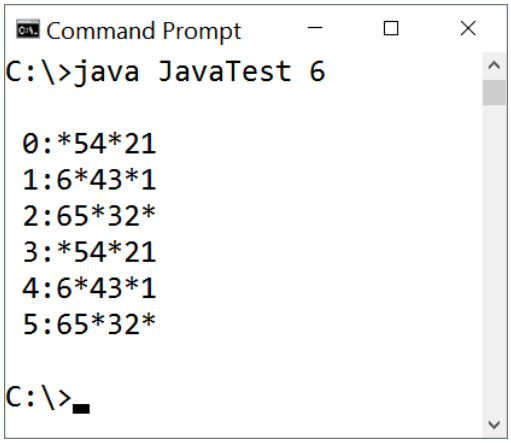
Lab Sheet 2: Java Basics / Javadocs / Methods

Java Basics:

1. A working Java program has been mixed up (like fridge magnets), as below:

```
else
}
public class JavaTest {
    public static void main(String[] args) {
System.out.print("*");
int loopUntil = Integer.parseInt(args[0]);
if (((i + j) % 3) == 0)
System.out.println();
System.out.println();
System.out.print(":");
}
}
for (int i=0; i < loopUntil; i++) {
}
System.out.print(" " + i);
for (int j=loopUntil; j > 0; j--) {
System.out.print(j);
```

Rearrange all the pieces above to create a working Java program that outputs the following:



2. Write a Java program called **`Pattern1`** using **nested loops** that prints a pattern, where the number should be taken from the command line argument. For example, running **`java Pattern1 5`** should produce the following output:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

3. Write a Java program called **`Pattern2`** using **nested loops** that prints a pattern, where the number should be taken from the command line argument. For example, running **`java Pattern2 5`** should produce the following output:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Methods:

4. Using your answers from ***Questions 2*** and ***3*** in this lab sheet, complete the code below. The class **`Patterns`** does the same things as what was required for ***Questions 2*** and ***3***. However, the code to print the patterns is not directly written in the **`main()`** method; instead, it is written in the two methods:

   `public void printPattern1(int n)` – **`n`** is the input number
   `public void printPattern2(int n)` – **`n`** is the input number

   The **`main()`** method of this class should create an instance of the class and then call the two methods above to print out the two patterns.

```
public class Patterns {
    public void printPattern1(int n) {
        // Write your code here (taken from your solution of Question 2).
    }
    public void printPattern2(int n) {
        // Write your code here (taken from your solution of Question 3).
    }
    public static void main(String[] args) { // Write your code here. }
}
```

Javadocs:

5. Download the file **`CountDownExample.java`** from the course website in QMplus (under the **COURSEWORK INFORMATION** section) and do the following:

   i) Compile and run the **`CountDownExample`** program.

   ii) Generate *Javadocs* for the **`CountDownExample`** program. In the directory where you stored the file **`CountDownExample.java`**, create a sub-directory called **`docCD`**[1] and then type the following command on the command line:

   `javadoc -d docCD CountDownExample.java`

   This command has created a whole set of different files in the **`docCD`** directory. Open up the file **`index.html`** in a web browser and take a look at what it has produced.

   iii) Alter the program **`CountDownExample`** so that it counts up, instead of counting down. When the program gets to the end of counting, it should now print the message **`All done!`** (rather than **`Time up!`**). Note that you will also need to change the *Javadoc* comments to your own comments. *Name your program* **`CountUpExample.java`**.

   iv) Generate *Javadocs* for the new program **`CountUpExample`**.

IMPORTANT: From now on, all your programs must contain both internal comments and *Javadoc* comments.

[1] Using **`mkdir docCD`** from the command line in your directory is faster than using a Windows file manager to do this!