Tutorial 1

COMP 1006/1406 Fall 2017

Objectives

Basic Java programming: control flow, arrays, input, output, and Strings.

A) Read the provided Tutorial00.java program. It gives examples of the basic Java that we will need to have so we can move forward in the course.

B) Modify the provided Tutorial01.java program. The comments in the java file tell you what you should be adding/modifying. Go to the **main** method and start there.

C) If you complete the tasks in the Tutorial01.java program, then you should finish the TicTacToe.java game (also provided). Add code so that a win is detected, a draw is detected. add code so that the game is displayed nicely (grid format).

D) Write a Java program that queries the user for a number, call is N, and then draws a an NxN grid of Qs. For example, if the user specifies 5 then the output would be

```
QQQQQ
QQQQQ
QQQQQ
QQQQQ
QQQQQ
```

Next, modify your code so that your program displays 4 triangles based on the input N. For example, if the number was 4, then the output should be

```
Q
QQ
QQQ
QQQQ

QQQQ
QQQ
QQ
Q

QQQQ
  QQQ
    QQ
      Q

      Q
    QQ
  QQQ
QQQQ
```

Some java reference material

Primitive Data Types

Java comes with eight (8) built-in primitive data types. These are

1. *byte*, *short*, *int*, and *long* (integer types)
2. *float* and *double* (floating point approximate real numbers)
3. *char* (single unicode character)
4. *boolean* (Boolean for logic)

For integers we usually use the *int* type, for approximate real numbers we will use the *double* type, and will use *char* and *boolean*.

Branching

Basic conditional execution is accomplished by *if* statement.

```
if(condition){
   statements
}
```

Here *condition* is a boolean expression. If If *condition* evaluates to *true*, then the block of code inside the curly braces is executed. Otherwise, execution goes to the next line after the closing curly brace.

More generally, we can a single *if* statement combined with zero or more *else if* statments followed by one or zero *else* statements.

```
if(condition1){
   statements1
}else if(condition2){
   statements2
}else if(condition3){
   statements3
}else{
   statements4}
}
statements5
```

Here, exactly ONE of statements1, statements2, statements3 or statements4 is executed. Statements5 is always executed.

There are two more ways that we can alter the flow of control of a program: using a *switch* statement and *throwing* exceptions. You will see the switch statement in Tutorial 2. Exception handling will be a topic we spend more detail in later in the course.

Repetition

There are three ways to repeat code (looping) in Java: the *while*, *do-while* and *for* loops.

A **while** loop will repeat zero or more times until *condition* evaluates to *false*.

```
while(condition){
  statements
}
```

A **do-while** loop will repeat one or more times until *condition* evaluates to *false*.

```
do{
  statements
}while(condition);
```

A **for** loop comes in two flavours in Java. The traditional for loop looks like

```
for(initialization; termination; increment){
  statements
}
```

For example, to print out the numbers from 0 to 9 using a for loop we would have

```
for(int i=0; i<10; i+=1){
  System.out.println(i);
}
```

Note that this achieves the same as

```
int i=0;
while(i<10){
  System.out.println(i);
  i+=1;
}
```

The *enahanced* for loop that more recent versions of Java have allows us to iterate over items in a list.

```
int[] numbers = {1,2,3,4}; // array of integers
for(int i : number){
  System.out.println(i);
}
```

This is essentially the same as

```
int[] numbers = {1,2,3,4}; // array of integers
for(int index=0; index<numbers.length; index+=1){
  int i = numbers[index];
  System.out.println(i);
}
```

Basic Output

Basic output to the *standard output stream* is done with the *print()* and *println()* functions. These functions (which will soon call **methods**) belong to the *out* object that lives in the *System* class.

For example,

```
System.out.print("hi");
System.out.println("there");
System.out.println("good-bye");
```

will display

```
hithere
good-bye
```

The *print()* function does not go to the next line when it is done (unless you tell it to). The *println()* always adds a newline character when it is done printing. You can always add your own newline characters manually, using the newline character escape sequence *\n* (slash n). For example, each line of the following do the same thing (print hi there and then newline).

```
System.out.print("hi there\n");
System.out.println("hi there");
```

User input

We will mostly use the *Scanner* class for user input from the keyboard.

```
import java.util.Scanner;

...


Scanner keyboard = new Scanner( System.in );
int = number;
System.out.print("enter an integer :");
int = keyboard.nextInt();
System.out.println("you entered " + int);
```

Have a look at the *Scanner* API. It shows all methods that you can use with the class.

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

Note: We need to be careful here. When we use a Scanner object to read an *int* but actually enter a string problems will happen. Try doing this when running your code.