MULTIMEDIA UNIVERSITY

CCP6224 – Object-Oriented Analysis and Design
Lab 1 – Review of I/O and control structures revision

Lab outcomes

By the end of today's lab, you should be able to

- compile code to make use of I/O libraries
- recall and use programming control structures to achieve the desired output in command-line applications
- be comfortable in handling command-line Java application program code

Java I/O

Java displays content to the command-line output using the **`System`** class which is loaded by default in every Java program. To display a line of content to screen you use **`println`** which will also forward the content to a new line. To remain on the same line the **`print`** function is used instead. The print functions from the **`System`** class support escape characters (**`\b\n\t`**) as well as format specifiers (**`%d`**, **`%c`**, **`%f`**) from **`printf`** as well as string concatenation from **`cout`**. (Ref: http://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html)

NOTE: Java actually supports **`printf`** like C but recommends its own methods to prevent confusion. It introduces **`format`** which works identically as a replacement to **`printf`** but retains it for legacy purposes and support.

Displaying content on the screen [Recommended 15 mins]

Using a text editor, copy the following program code and save the file. The code shows the difference between the different display functions from the System class.

```
public class JavaIO{
      public static void main(String[] args){
        int x=10,y=30;
        System.out.println("This displays a line of text");
        System.out.print("This line appears underneath the first,");
        System.out.print(" and this appears next to that\n");

        System.out.printf("This line uses printf with %d specifier\n", 1);
        System.out.format("This line uses format with %d specifier%n", 1);

        System.out.println("x * y = " + x * y);
        System.out.println("x + y = " + x + y);
      }
}
```

Note: **`format`** *recommends* **`%n`** *but* **`\n`** *will work too!* **`%n`** *compiles to platform specific newline character*

Compile and run the code and answer the following questions.

1. What is the name of the class in this Java program?
2. What should the filename of the program be saved as?
3. What specifier replaces '%d' if you wanted to print the string "one" instead of a digit?
4. What is wrong with output in the last **`println`** statement?
5. How do you fix the last **`println`** statement?

Taking in user input [Recommended 15 mins]

User input requires the **`Scanner`** class which means the *import of java.util.* package* into the program. To use the **`Scanner`**, first create a **`Scanner`** class instance and assign to the desired source (keyboard input? file? web input? etc.). The code here takes input from keyboard:

```
Scanner scan = new Scanner(System.in);
```

Once the **`Scanner`** instance is ready, then use the provided methods from the class to obtain the desired input (**`nextDouble`**, **`nextInt`**, **`nextLine`** etc)
(Ref: http://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html)

Copy the program code below, compile and run to see the Scanner class in action.

```
import java.util.*; // or java.util.Scanner

public class ComputeAverage {
      public static void main(String[] args) {
            // Create a Scanner object
            Scanner input = new Scanner(System.in);

            // Prompt the user to enter three numbers
            System.out.print("Enter three numbers: ");
            double number1 = input.nextDouble();
            double number2 = input.nextDouble();
            double number3 = input.nextDouble();

            // Compute average
            double average = (number1 + number2 + number3) / 3;

            // Display result
            System.out.println("The average of " + number1 + " " + number2
             + " " + number3 + " is " + average);

      } // end main
} //end class
```

Control structures in Java – For reference

The control structures of Java, such as the for loop, while loop, if statement, case statement etc., have a syntax very similar to that of C. Only a very few additional features (like the declaration of the loop variable in a for-statement) are incorporated into Java. Try out the following simple programs to see how the control structures work.

The while loop:

```
public class Test {
   public static void main(String args[]){
      int x=10;

      while( x < 20 ){
         System.out.print("value of x : " + x );
         x++;
         System.out.print("\n");
      }
   }
}
```

The do...while loop:

```
public class Test {
   public static void main(String args[]){
      int x= 10;

      do{
         System.out.print("value of x : " + x );
         x++;
         System.out.print("\n");
      }while( x < 20 );
   }
}
```

Enhanced for loop in Java:

```
public class Test {
   public static void main(String args[]){
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ){
         System.out.print( x );
         System.out.print(",");
      }
      System.out.print("\n");
      String [] names ={"James", "Larry", "Tom", "Lacy"};
      for( String name : names ) {
         System.out.print( name );
         System.out.print(",");
      }
   }
}
```

The for loop:

```
public class Test {
   public static void main(String args[]){
      for(int x = 10; x < 20; x = x+1){
         System.out.print("value of x : " + x );
         System.out.print("\n");
      }
   }
}
```

The break keyword:

```
public class Test {
   public static void main(String args[]){
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ){
         if( x == 30 ){
            break;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }

}
```

The continue keyword:

```
public class Test {
   public static void main(String args[]){
      int [] numbers = {10, 20, 30, 40, 50};

      for(int x : numbers ){
         if( x == 30 ){
            continue;
         }
         System.out.print( x );
         System.out.print("\n");
      }
   }
}
```

The if statement:

```
public class Test {
   public static void main(String args[]){
      int x = 10;
      if( x < 20 ){
         System.out.print("This is if statement");
      }
   }
}
```

The if…else statement:

```
public class Test {
   public static void main(String args[]){
      int x = 30;

      if( x == 10 ){
         System.out.print("Value of X is 10");
      }else if( x == 20 ){
         System.out.print("Value of X is 20");
      }else if( x == 30 ){
         System.out.print("Value of X is 30");
      }else{
         System.out.print("This is else statement");
      }
   }
}
```

Nested if…else statement:

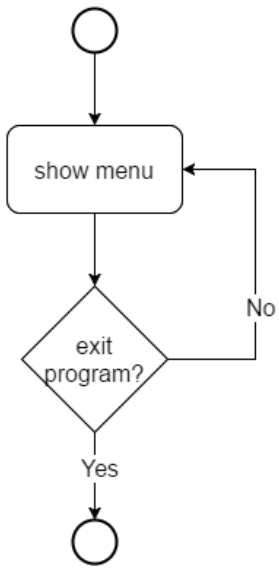
```
public class Test {
   public static void main(String args[]){
      int x = 30;
          int y = 10;

      if( x == 30 ){
         if( y == 10 ){
         System.out.print("X = 30 and Y = 10");
      }
   }
}
```

The switch-case statement:

```
public class Test {
   public static void main(String args[]){
      char grade = args[0].charAt(0);

      switch(grade)
      {
      case 'A' :
         System.out.println("Excellent!");
         break;
      case 'B' :
      case 'C' :
         System.out.println("Well done");
         break;
      case 'D' :
         System.out.println("You passed");
      case 'F' :
         System.out.println("Better try again");
         break;
      default :
         System.out.println("Invalid grade");
      }
      System.out.println("Your grade is " + grade);
   }
```

Using control structures to create looping menus

A looping menu returns to a fixed point in execution until a specific termination criterion is met. This usually involves encompassing an entire code block in a do-while loop so the program continues to loop until the block satisfies an exit criterion.



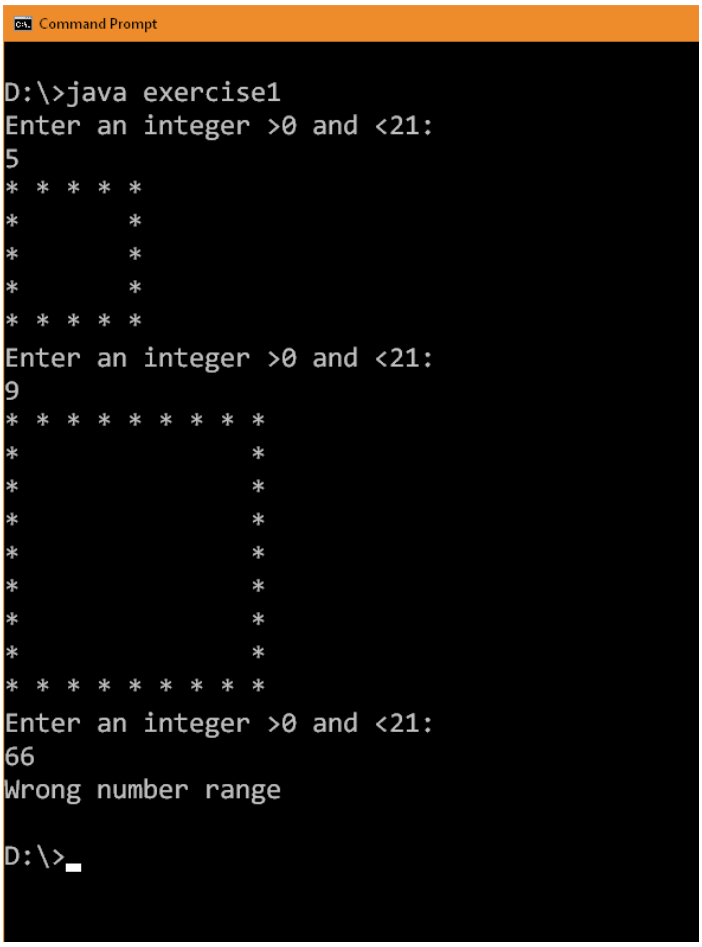
A simple example of the loop shown in the flowchart is shown below

```
import java.util.*; // or java.util.Scanner
import java.io.*;

class TestLoop {
     public static void main(String[] args) {
          // Create a Scanner object
          Scanner input = new Scanner(System.in);
          Boolean quit = new Boolean(false);
          while (!quit){ // can be replaced with "true"
               System.out.println("Enter y to quit any other key
                                                  to continue]: ");
               char in = input.next().charAt(0);
               if (in=='y')
                    quit = true; // or use "break"
          }

     } // end main
} //end class
```

Exercise [Recommended 60-90 mins]

Exercise 1

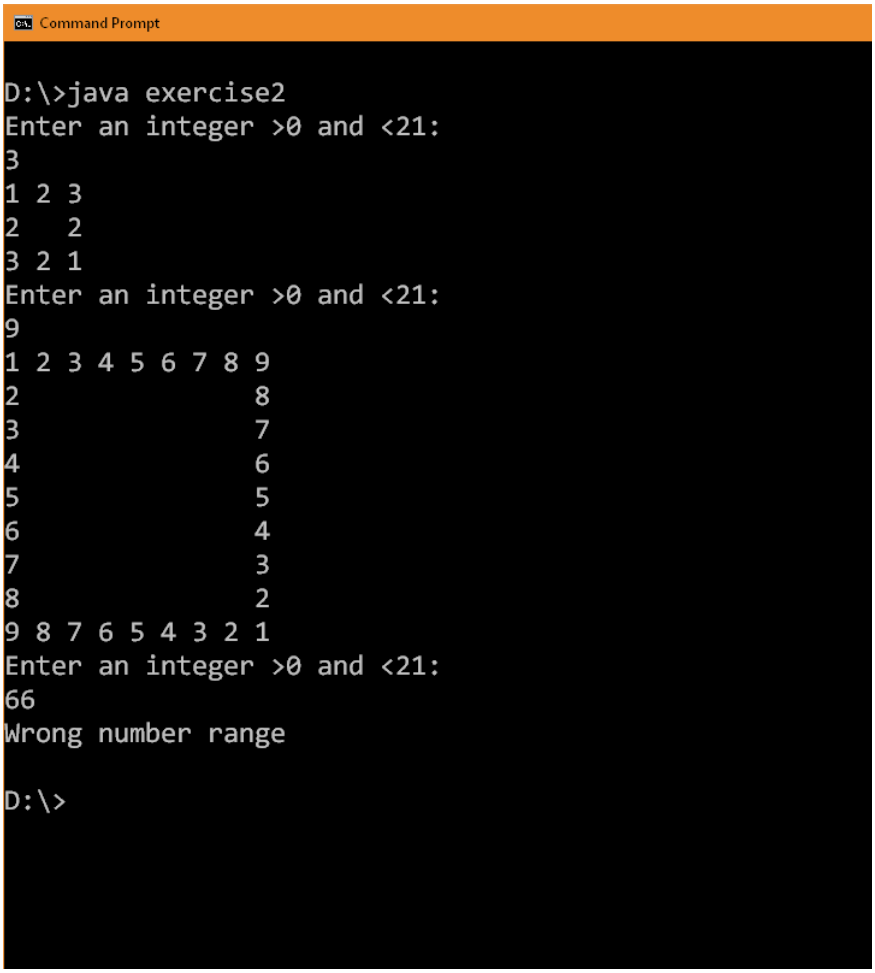
Write a Java program that takes in user input of a single integer value between the ranges of 1 to 20. Use the input to draw a square with the sides of the square being the value entered. For example, if your program reads in an input integer value of 5, it should display the following.



Your program should loop continuously and ask the user for the next input. If the user enters an out-of-range/illegal number, the program ends (Note your Java program may terminate on its own if you enter wrong data type and this is OK because we have not yet covered exception handling)

Exercise 2

Alter the control structure of the previous program such that when the user input is read, the item drawn on the screen is as follows. The range of acceptable input remains between 1 and 20.

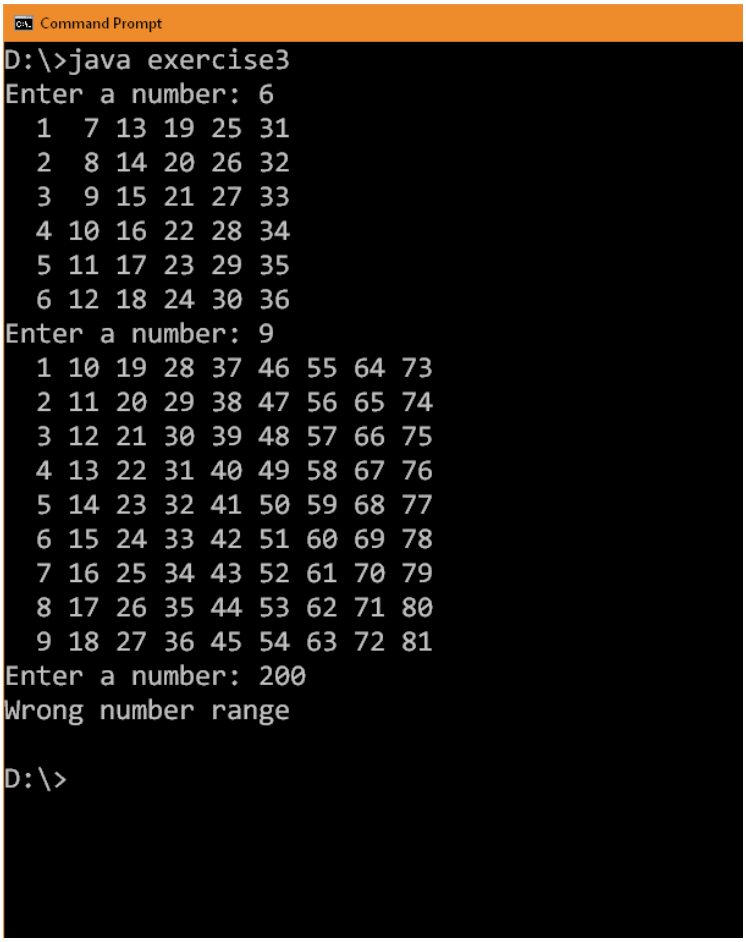


Your program should loop continuously and ask the user for the next input. If the user enters an out-of-range/illegal number, the program ends (Note your Java program may terminate on its own if you enter wrong data type and this is OK because you we have not yet covered exception handling)

Exercise 3

Alter the program from exercise 2 to display the following based on user input. Instead of a hollow square, display an incrementally increasing number starting from 1 onwards from top left to bottom right of the square. The range of acceptable input remains between 1 and 20.

NOTE: To get proper spacing, you can make use of the `format` display method instead of `print` or `println` but this is optional
(Ref: http://docs.oracle.com/javase/tutorial/essential/io/formatting.html)



Your program should loop continuously and ask the user for the next input. If the user enters an out-of-range/illegal number, the program ends (Note: your Java program may terminate on its own if you enter the wrong data type, and this is OK because we have not yet covered exception handling)

Exercise 4

For the final exercise, take in user input of a single word and create the four-sided figure as shown in the screenshot below (note the shift of the letters in each line).

You may need to reference online for additional methods to take character inputs and convert inputs to an array.