Java Labs

Each weeks labs should be held in a separate folder: E.g. lab1.

Create this new folder each week in your user directory on the u drive - E.g. u:\software_development_1\lab1.

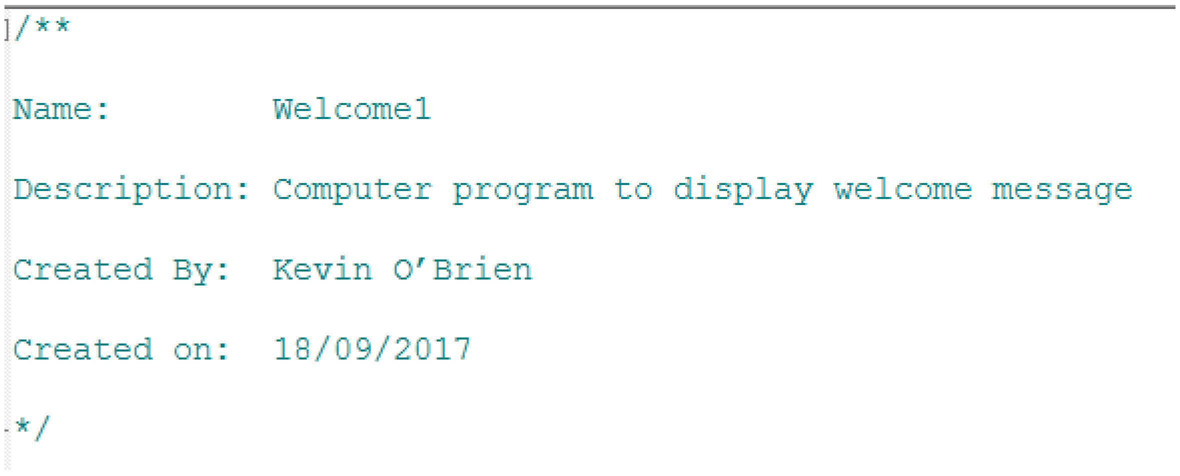
Save all files you work on in this lab to this folder.

All Exercises should be completed using TextPad.

To compile your code in TextPad, hit the ctrl key and 1 key. Hold down the ctrl key, while holding that key down press 1.

To run your compiled program, .class file, in TextPad, hit the ctrl key and 2 key. Hold down the ctrl key, while holding that key down press 2.

For all programs please insert a header similar to the following:



```
/**

Name:        Welcome1

Description: Computer program to display welcome message

Created By:  Kevin O'Brien

Created on:  18/09/2017

*/
```

For all exercises, follow the instructions to implement the solution.

Introduction

In this weeks lab we will be focussing on creating basic computer programs to implement:

- Methods
- Shadowing
- Overloading
- Arrays
- Try and Catch
- Enhanced For

Note that at the end of the end of the lab one or more of the exercises need to be submitted to your GitHub repository you created on week 19. Exercise(s) to submit to Git will be indicated to you by your lecturer.

Section A

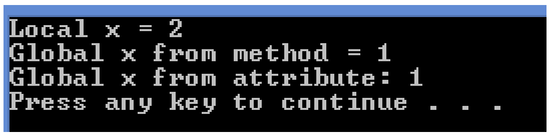
Exercises 1 to 9. Complete All.

Exercise 1 – Shadowing (Shadow.java)

Write a program, Shadow.java. The requirements for this program are as follows:

- Define a global variable, x, of type int and assigns the value 1 to it.
- Create a method named globalVar(), which returns the value assigned to this global variable x.
- Create a method named localVar(), which contains a local variable, also named x and of type int and assign it the value 2. The method should return this value.
- Create a main() method which calls both these methods, globalVar() and localVar() and displays the return values from both these program.
- Main() should specify an instance of Shadow, so it should be possible to use that instance to access the global value of x directly.

An example output of this program is as follows:



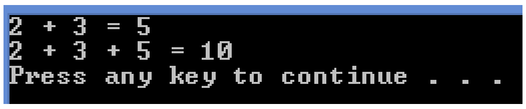
```
Local x = 2
Global x from method = 1
Global x from attribute: 1
Press any key to continue . . .
```

Exercise 2 – Overloading (Arithmetic.java)

Write a program, Arithmetic.java, which contains two definitions for a method named addNumbers(). One addNumbers() method definition accepts two arguments of type int, and returns the result of adding those two numbers together. The other addNumbers method definition accepts thress argument of type int, and returns the result of adding those 3 numbers together.

Implement a main() method which calls both versions of the addNumbers method and display the result returned. Just hardcode the values to pass to the methods.

An example output is as follows:

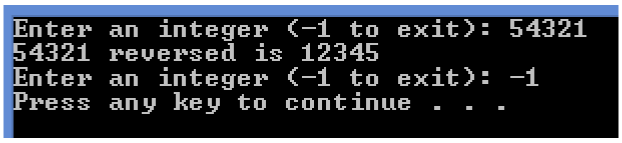


```
2 + 3 = 5
2 + 3 + 5 = 10
Press any key to continue . . .
```

Exercise 3 – Reversing Digits (Reverse.java)

Write a program, Reverse.java, which includes a method that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the method should return 1367. Incorporate the method into a program that reads the value from the user and displays the result. After each number is entered and the result is displayed, prompt the user again to enter a number. Use a sentinel-controlled loop for this purpose.

An example output from this program is as follows:



```
Enter an integer (-1 to exit): 54321
54321 reversed is 12345
Enter an integer (-1 to exit): -1
Press any key to continue . . .
```

Exercise 4 – Arrays (FirstArray.java)

Write a program, FirstArray.java, which implements the following array:

```
int[] x;
x = new int[4];
```

Using a for loop which uses the length property of an array in the condition, print the values of this array to the screen.

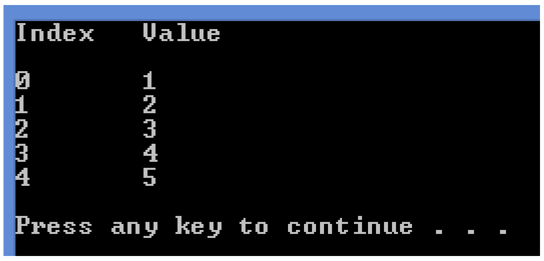
Exercise 5 – Arrays (SecondArray.java)

Write a program, SecondArray.java, which prints the values assigned to each element of the array. Use the following array definition for this purpose:

```
int[] x = {128, 132, 8, 156, 18};
```

Exercise 6 – Arrays (ThirdArray.java)

Write a program, ThirdArray.java, which creates an array that will hold 5 integers. Assign value of 1 to first element, value of 2 to second element, etc … Use a loop to iterate through the array in order to assign the values to each element of the array. Once the values are assigned to each element then iterate through the array to print out these values to the screen. An example output is as follows:



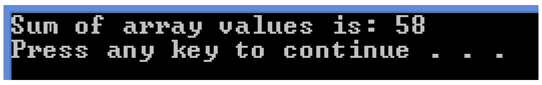
```
Index    Value

0        1
1        2
2        3
3        4
4        5

Press any key to continue . . .
```

Exercise 7 – Arrays (FourthArray.java)

Write a program, FourthArray.java, which implements the following array:

```
int[] x = {7, 6 , 11, 15, 19};
```

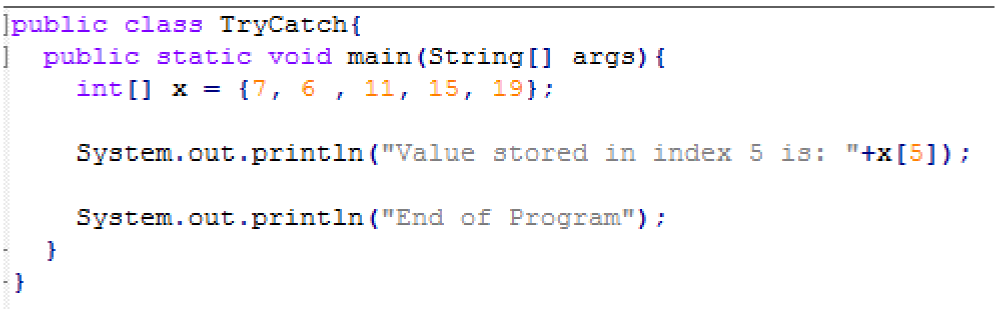
The program should iterate through the array and add all the values assigned to it to a total. The total should be displayed on the screen. An example output is as follows:



```
Sum of array values is: 58
Press any key to continue . . .
```

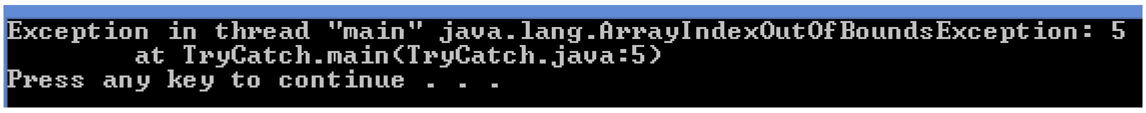
Exercise 8 – Try and Catch (TryCatch.java)

Implement the following program:



```
public class TryCatch{
  public static void main(String[] args){
    int[] x = {7, 6 , 11, 15, 19};

    System.out.println("Value stored in index 5 is: "+x[5]);

    System.out.println("End of Program");
  }
}
```

Compile and run this program. It will produce an error as follows:



```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at TryCatch.main(TryCatch.java:5)
Press any key to continue . . .
```

Amend this program to include a try and catch block to handle this exception.

Exercise 9 – Enhanced For Loop (EnhancedFor.java)

Take a copy of the solution for exercise 7, FourthArray.java, rename it EnhancedFor.java, and replace the for loop with an enhanced for loop to produce a similar output.