# 4CS001 Workshop 4

If you have not already done so, change directory to 4CS001 and make a new directory called Week04. Last week we gave you some examples of code for handling Strings.

1. The plus operator allows concatenation:

**String word1 = "Java";**

**String word2 = "I Like ";**

**String word3 = word2 + word1;**

1. There is also a concatenate method:

**String word1 = "Java";**

**String word2 = "I Like ";**

**String word3 = word2.concat(word1);**

1. String equality:

**String word1 = "Java";**

**String word2 = "Java";**

**boolean same;**

**System.out.println("same=" + word1.equals(word2));**

*equals* returns a *boolean* value

*true* if both strings contain the same sequence of characters

*false* if the character sequence is different.

In this example *same* = true.

*equalsIgnoreCase* can be especially useful

1. String length:

**String txt = "Hello World!";**

**System.out.print(txt + " contains ");**

**System.out.println( txt.length() + " characters");**

Returns the number of characters in the String

*txt.length()* in this example would return 12

1. Occurrence of a character in a String:

int indexOf(char)

Returns the index of the first occurrence of the char, txt.indexOf(“l”) returns 2.

int lastIndexOf(char)

Returns the index of the last occurrence of the char, txt.lastIndexOf(“l”) returns 9.

1. String replaceAll replaces all instances of old with new:

**String txt = "Hello World!";**

**String newTxt = txt.replaceAll("o","x");**

1. String toUpperCase converts all characters to upper case:

**String txt = "Hello World!";**

**String newTxt = txt.toUpperCase();**

1. String toLowerCase converts all characters to lower case:

**String txt = "Hello World!";**

**String newTxt = txt.toLowerCase();**

1. boolean startsWith(String):

txt.startsWith("He") returns *true*

txt.startsWith("he") returns *false*

1. boolean endsWithString:

txt.endsWith("orld!") returns *true*

txt.endsWith("world!") returns *false*

1. char charAt(int index) returns the character at a specified position (index):

char txt = txt.charAt(6) returns the character W

1. String substring(int beginIndex) returns the characters of the string following the index:

String result = txt.substring(6) returns the text World!

1. String substring(int beginIndex, int endIndex) returns the characters between the indices:

String result = txt.substring(6,11) returns the text World

## Processing Names – exercise 1.

The aim of this exercise is to gain experience working with programs that contain textual data and read textual data from the keyboard.

Write a program that inputs somebody's name from the keyboard (see lecture notes from week 2 on JOptionPane) outputs various details and formats of their name (see example below). Assume that the input always includes 3 parts: forename, middle name and last name and produces output like that below. Make sure that you test the program after completing each part.

Name is james tiberius kirk

Length is 19

Name in capitals is JAMES TIBERIUS KIRK

Index of first space is 5

Index of last space is 14

First name is james

Last name is kirk

Middle name is tiberius

Initials are JTK

Capitalised first name is James

Capitalised middle name is Tiberius

Capitalised last name is Kirk

Full capitalised name is James Tiberius Kirk

Email "from format" is Kirk, James T.

email address is J.T.Kirk@wlv.ac.uk

Try typing in a name with CAPS LOCK switched on and check that the output is still identical to this example.

## Processing Numbers – exercise 2.

Write a program to read in three strings, using JOptionPane, consisting of entirely numeric digits. Assume the user always types in strings that are in the format required for integers.

Convert the contents of the strings to integers (using Integer.parseInt) then add them together and print out the result.

## Processing Numbers – exercise 3.

Write a program similar to the previous one, but allow users to enter numbers that contain decimal places. You can use Double.parseDouble to convert strings to doubles.

## Revisiting your Calculations Programs – exercise 4.

Take what you have learnt from the previous two exercises and revisit one of the programs that you wrote last week or the week before, and make it interactive.

For example, adapt the coins change program you wrote in week 2 so that it asks the user to type in the cost of goods sold, and then use the code you wrote to output the required coins for the change.

## Selection Exercises

1. Copy the following code and paste it into your editor window.

**import javax.swing.JOptionPane;**

**public class Holidays1 {**

**public static void main(String[] args) {**

**String ageText = JOptionPane.showInputDialog(null, "Please enter your age:");**

**int age = Integer.parseInt(ageText);**

**if(age<18 || age>30) {**

**System.out.println("You are not eligible for the holiday");**

**} else {**

**System.out.println("You are eligible for the holiday");**

**}**

**}**

**}**

1. Compile the code and run the program. The program asks the user to enter their age and then outputs a message saying whether they are eligible to go on a Club 18-30 holiday. People between the ages of 18 and 30 inclusive should be eligible, whilst ages 17 or less and 31 or greater should not be eligible. Run the program several times using ages 17, 18, 19, 29, 30, 31 to test that it works properly.

This is TASK 5 of your WORKBOOK. Ensure it is marked by the end of week 6.

1. Create another class called Holidays2 and copy the program code from Holidays1 into it. Change the condition so that it makes use of && instead of ||. You will now have to make other changes to the code to ensure that it advises the user correctly. You should test your new program with the values used before to ensure it operates correctly.

This is TASK 5 of your WORKBOOK. Ensure it is marked by the end of week 6.