ITC502

Fundamentals of Programming and Problem Solving

**Assessment: Lab Exercises**

**Weighting: 20%**

**Learning Outcomes assessed in bold**

1. **Programming concepts and tools**
2. **Awareness of procedural and object oriented programming**
3. Principles of Implementation (user testing, deployment)
4. **Programmes must include core logic and mathematical concepts such as problem solving methods, critical thinking, abstract reasoning; and systems thinking**

**Instructions**

* Complete labs 1 to 40 marking getting your Tutor to mark your work as you complete the lab.
* Submit all lab via blackboard at the end of the semester

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Question Type** | **Possible Marks** | **Marks Achieved** | **%** |
| **Part A** | Labs 1 -40 | 40 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Total** |  | **40** |  |  |

# Getting started with BlueJ

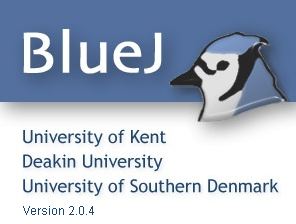
### You can download BlueJ from <http://www.bluej.org> to install on your personal PC, it’s usually easiest to go for the version that includes JDK.

### Before you start programming on either your personal PC or SIT PC. Copy the newclass.zip (available on blackboard) into the folder C:\BlueJ\lib\english\templates\newclass and unzip it overwriting old files that are in there (right click and chose the menu option extract to here).

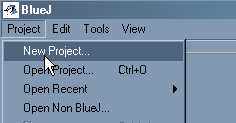
### Your first Program “Hello World”

Traditionally the first program that students of a new language write is a program which writes “Hello World” on the screen – this will take you through how to do that with BlueJ

* Start BlueJ



* From the Project menu choose New Project

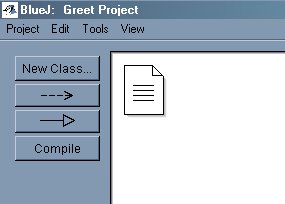


* Give the project a name – I suggest “Greet Project” and click Create



You will now have a project with one thing in it, this is a ReadME document in which you can write information about your project and yourself as the author –

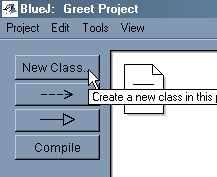
* double click on the ReadMe document to edit it
* fill in some details about yourself and the project
* then close the editor.



### You now want to create your Application program

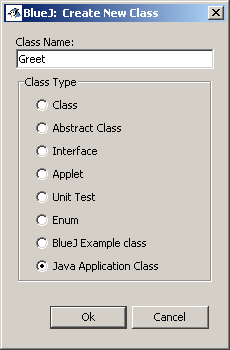
Each Java application program starts with a Java Class so we need to create a new class

* Click the **New Class** button



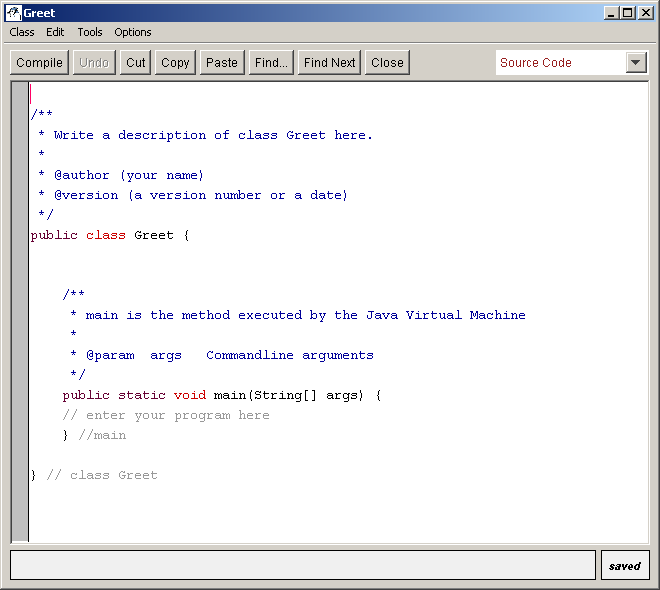
You will be asked What type of class you want – Choose the Java Application class

* Give the class a name (I suggest you call it Greet) then click OK



Now you have two things in your Project, your ReadMe document and your Greet class.

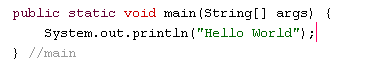
* Double click on the Greet class to edit it



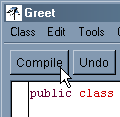
Fill in your name where it says (your name)

and version number 1 where it says (a version number or a date)

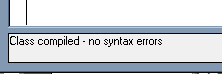
You need to add a line in the ***method*** called **main** to tell the program to print   
“Hello World”



* Enter this into your editor and click the **Compile** button



If you typed everything correctly you should see **Class compiled – no syntax errors** at the bottom of your editor



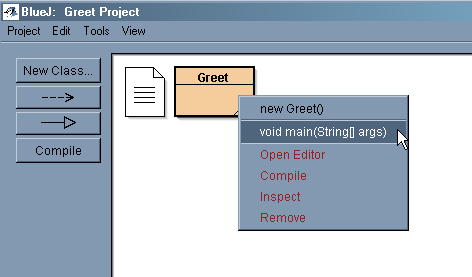
Now you can run your program,

* Click on the BlueJ project window from the taskbar

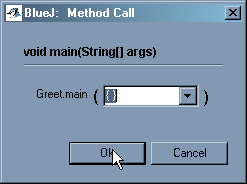


### Run your program

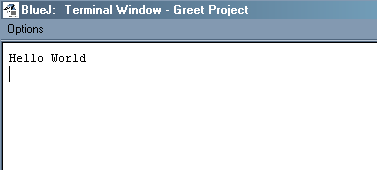
* Right click on the Greet class and select your main method from the list of things it gives you



* Just click OK on the next screen – later we will explain what it can be used for



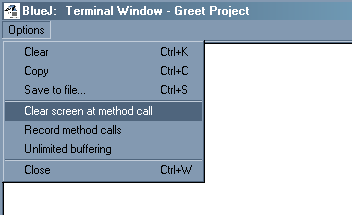
You should see a Terminal window come up and your program should run!



Before you close the terminal Window, tell it to clear the window each time a pogram runs – to do this

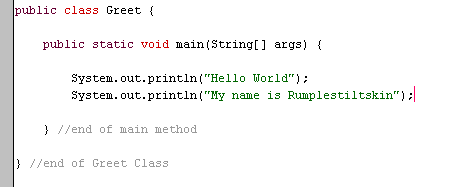
* choose **Clear screen at method call** from the Options menu.

## You only need to do this once



### Making some changes to your program.

* Go back to the Editor with the Greet class in it and add another System.out.println instruction – this time telling us something about you



* Click compile again and run your program

### What are the parts of the program?

The **public class Greet {** and the **} //end of Greet class** are the beginning and end of your program

The name Greet will change but those lines will be in all your programs

The **public static void main(String[] args){** and **} //end of main method** are the beginning and end of an action that you are teaching the computer to do. At the moment you are just writing things on the screen. The System.out.println instruction is the one that writes things on the screen. Anything you put inside System.out.println will be written onto the screen. Put quotes around anything you are writing out that isn’t number.

System.out.println(“Hello World”);

System.out.println(“Today is a nice day”);

System.out.println(42);

Write a program which write a story about you to the screen

*You can get the program to do maths too!*

* Try these

System.out.println( 4 + 5 );

System.out.println( 5 \* 4 );

System.out.println( 10 – 4);

#### Or you can mix things together - try this

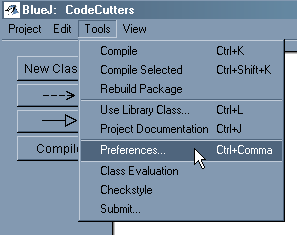
System.out.println( 5 + " times " + 4 + " is " + ( 5 \* 4) );

* Write a program that writes out the 15 times table doing its own calculations

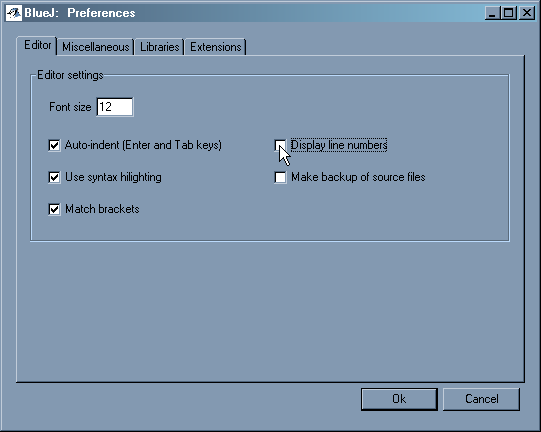
# Getting BlueJ to tell us what line we are on

Sometimes it is useful to know what line we are on

* From the **tools** menu – choose **preferences**



You will see this dialog box



* Click in the box which is labelled Display line numbers



* then click 

# Asking the user for information

We have so far learnt to write things to the screen, but to be really useful we need to be able to read things from the keyboard as well!

To do this we will need to use a special tool called a **Scanner**.

Scanner is a class just like your program is, and we can use **Scanner** objects and other pre-made objects to make our programs more powerful.

Scanner is part of a package of useful classes called **java.util**

### The Scanner object

To make a new **Scanner** object we need to give it a name then tell it where to get its information from.

A good name for the Scanner object we are using would be **keyboard** because we are going to use it to read from the keyboard.

*Here is how you make a Scanner object called keyboard.*

java.util.Scanner keyboard;

*and to set it up to get its information from the keyboard you write*

keyboard = new java.util.Scanner(System.in);

*if you like you can do it all in one step*

java.util.Scanner keyboard = new java.util.Scanner(System.in);

# Where do we put the information that you read in?

If we ask our Scanner to give us some information then we need to store it somewhere! We call this place a **variable**.

# A place to store a number

int num;

This creates a place to store an integer number such as 4, 10, -5

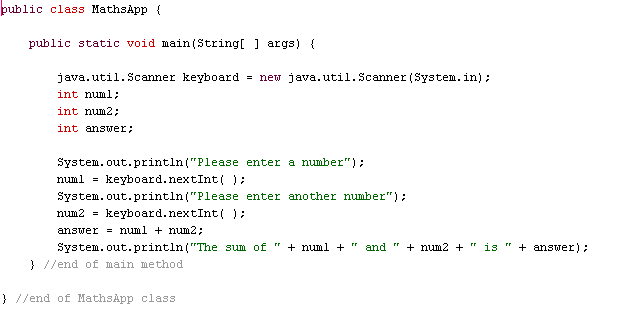
### An example

This program asks for two numbers

It then reads them in using a Scanner object and stores them in two variables, **num1** and **num2**

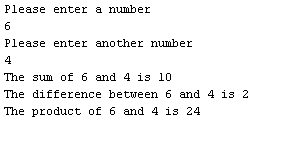
It then adds together the numbers which are stored in **num1** and **num2** and stores the sum in a variable called **answer**

It then prints out a line telling the user what the numbers were and what the sum was



* Try this program and then change it so it prints out the difference (num1 – num2) and the product (num1 \* num2 ) as well as the sum. Computers use \* for multiplication instead of x.

##### For example



# Some useful tips

**Blank lines**

This is how your can write a blank line

System.out.println( );

###### Staying on the same line

When we write **System.out.println("Please enter a number");** The next thing is written or read in from the line after "Enter a number".

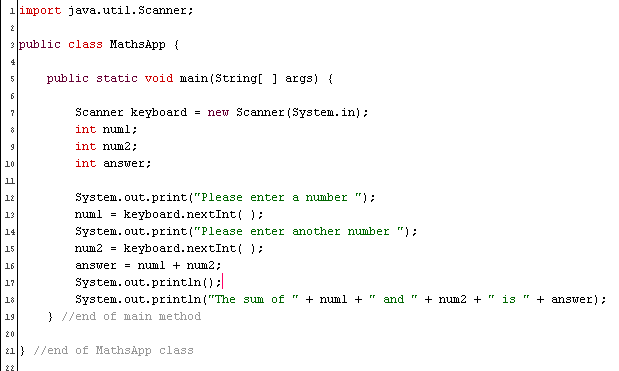
Sometimes we want to stay on the same line, so for example we can type in the answer on the same line as the request. We do this using **System.out.print** instead of **System.out.println**

Try **System.out.print("Please enter a number ");** in your program instead of **System.out.println("Please enter a number");** and look for the difference.

**import**

We can make life a little easier by telling Java to always look for **Scanner** in the **java.util** package instead of writing out the whole **java.util.Scanner** each time we want to use it. To do this we add a line **import java.util.Scanner;** at the start of our program.

###### So our program now becomes



# Reading in lines of words

When storing numbers we used a variable of type **int**.

To read in alphanumeric data such as names and addresses we need a different type of variable, a **String** variable

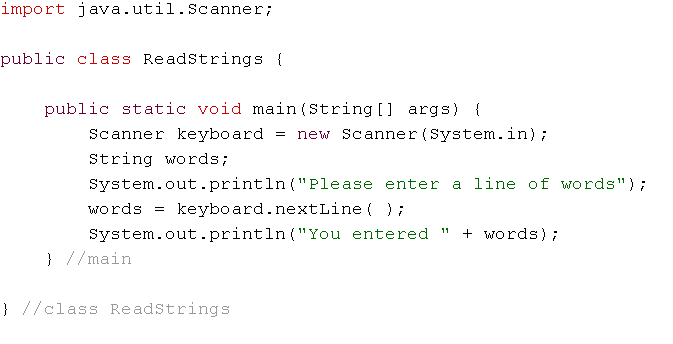
For example: String address;

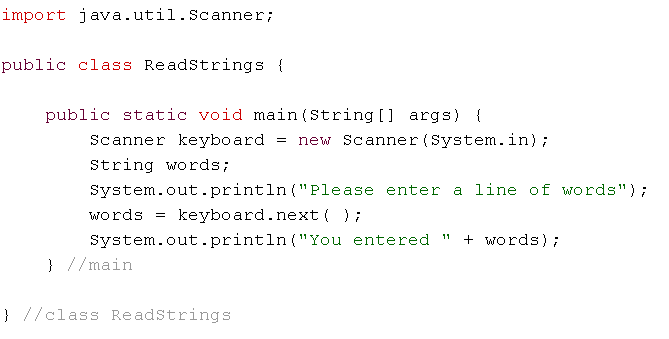
As you have seen a Scanner has a built-in method for reading **int** data and that method is **nextInt( )**

For example: num = keyboard.nextInt( );

A Scanner also has another methods for reading in String data, next( ) and   
nextLine( ).   
**next( )** reads data until it encounters a space, **nextLine( )** reads the entire line

Try these programs and compare how next( ) and nextLine( ) work





# Laboratory Exercise 1 – Greet

Read, from standard input, a person’s family name, their first name and their home location – each on a separate line

Write a greeting to that person in the form

Hello *first-name* *family-name* of *location*

(use single spaces for separators)

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Smith | Hello Anne Smith of Bluff |
| Anne |  |
| Bluff |  |

# Laboratory Exercise 2 – Basic Maths

Read, from standard input two integer numbers and write out their sum, difference and product as shown in the example (use single spaces for separators)

**(Hint Use printf)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 12 6 | 12 + 6 = 18 |
|  | 12 – 6 = 6 |
|  | 12 x 6 = 72 |

# Laboratory Exercise 3 – Shopping

Read, from standard input, the name of an item, its price (a double) and the quantity of that item (an integer) each on a single line.

Write out the details on a single line with the total price as shown in the example below.

All currency amounts should be displayed to 2 decimal place precision and with a $ in front. The quantity should have no decimal places.

(use single spaces for separators)

**(Hint Use printf)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Bread | 3 x Bread @ $1.15 Total: $3.45 |
| 1.15 |  |
| 3 |  |

# Laboratory Exercise 4 – Formal Names

Read, from standard input, a name consisting of a surname followed by a first name and separated by a single comma and a single space.

The output of should be the first name followed by the surname and separated by a ***single*** space.

**(Hint Use substring)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Smith, Anne | Anne Smith |

# Laboratory Exercise 5 – Discount

Read, from standard input, the name of an item, its price (a double) and the quantity of that item (an integer) each on a single line.

Write out the details on a single line with the total price as shown in the example below.

If the quantity is 10 items or more then give the customer a 10% discount

All currency amounts should be displayed to 2 decimal place precision and with a $ in front. The quantity should have no decimal places.

(use single spaces for separators)  
**(Hint Use printf,** in printf "%%" is used to print out a single % symbol)

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Bread | 3 x Bread @ $1.15 |
| 1.15 | Total: $3.45 |
| 3 |  |

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Bread | 12 x Bread @ $1.15 |
| 1.15 | Subtotal: $13.80 |
| 12 | -10% Discount: $1.38 |
|  | Total: $12.42 |

# Laboratory Exercise 6 – Greet many names

Read, from standard input, a series of names each on a separate line and ending with a sentinel value of # on a line of its own.

For each name output a line in the form Hello name as shown below

Do not process the sentinel value.

**(Hint Use while – sentinel controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Sam | Hello Sam |
| Sue | Hello Sue |
| # |  |

# Laboratory Exercise 7 – PASS FAIL

Read, from standard input a series of integers, one per line, ending in a sentinel value of -1;

Each integer represents a test score (in the range 0 to 100).

For each score output a line with FAIL for a score less than 50 or PASS for a score greater or equal to 50

Do not process the sentinel value.

**(Hint Use while – sentinel controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 12 | FAIL |
| 60 | PASS |
| -1 |  |

# Laboratory Exercise 8 – PASS FAIL MERIT

Read, from standard input a series of integers, one per line, ending in a sentinel value of -1;

Each integer represents a test score (in the range 0 to 100).

For each score output a line with FAIL for a score less than 50; PASS for a score greater or equal to 50 but less than 80; or MERIT if the score is greater than or equal to 80.

Do not process the sentinel value.

**(Hint Use while – sentinel controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 12 | FAIL |
| 90 | MERIT |
| 55 | PASS |
| -1 |  |

# Laboratory Exercise 9 – Sum and Average

Read, from standard input a series of integers, one per line, ending in a sentinel value of 999. The numbers will be in the range 0 to 100.

Output on a line **Sum:** followed by the sum of the series of integers as shown below.

Output on a new line **Average:** followed by the mean average of the series of numbers formatted to a precision of 2 decimal places as shown below.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 12 | Sum: 30 |
| 7 | Average: 10.00 |
| 11 |  |
| 999 |  |

# Laboratory Exercise 10 – Times table

Read, from standard input a single integer

Write out the times table of that integer from **1 times** the number to **10 times** the number and formatted as shown below.

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 | 1 times 5 is 5 |
|  | 2 times 5 is 10 |
|  | 3 times 5 is 15 |
|  | 4 times 5 is 20 |
|  | 5 times 5 is 25 |
|  | 6 times 5 is 30 |
|  | 7 times 5 is 35 |
|  | 8 times 5 is 40 |
|  | 9 times 5 is 45 |
|  | 10 times 5 is 50 |

# Laboratory Exercise 11 – Star Count

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Output the number of times an asterix occurs on each line of input

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4\*ert\*qwe | 2 |
| Qwweerrt | 0 |
| 45t\*qwe\*rtesdf\*\*dgdr | 4 |
| \*\*\*\*\* | 5 |
| # |  |

# Laboratory Exercise 12 – Word Count

Write a program that reads in a series of lines terminated by a line containing only a single '#'

By counting the spaces, output the number of words on each line

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Hello world | 2 |
| Today is sunny | 3 |
|  | 0 |
| Abracadabra | 1 |
| # |  |

# Laboratory Exercise 13 – Reverse a String

Write a program that reads in a series of lines terminated by a line containing only a single '#'

For each line output the line backwards

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Hello | olleh |
| today is sunny | ynnus si yadot |
| # |  |

# Laboratory Exercise 14 – How many vowels?

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Output the number of vowels in each line of input

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4\*ert\*quwe | 3 |
| Qwweerrt | 2 |
| 45at\*qwe\*rtesdf\*\*dgdr | 3 |
| \*\*\*\*\* | 0 |
| # |  |

# Laboratory Exercise 15 - Vowel Count

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Output the number of times each vowel occurs in each line of input.

Output the results in the format lowercaseVowel:count with a comma separating each vowel:count pair eg: a:3,e:0,i:2,o:4,u:1

Ignore the case of the vowels read (ie aAAaEe counts as a:4,e:2,i:0,o:0,u:0)

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4\*ert\*quwe | a:0,e:2,i:0,o:0,u:1 |
| QwwEerrt | a:0,e:2,i:0,o:0,u:0 |
| 45At\*qwe\*rtEsdf\*\*dgdr | a:1,e:2,i:0,o:0,u:0 |
| \*\*\*\*\* | a:0,e:0,i:0,o:0,u:0 |
| # |  |

# Question 16 – Count the stars II

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Output the number of times an asterix occurs on each line of input

If an ! appears on the line do not count asterixes beyond the !

Do not output a result for the line containing the #

**(Hint Use break)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4\*ert\*qwe | 2 |
| Qwweerrt | 0 |
| 45t\*qwe\*rtes!df\*\*dgdr | 2 |
| \*\*\*\*\* | 5 |
| # |  |

# Question 17 – Test Scores

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains names and four test scores in the format   
Surname,Firstname, score1, score2, score3, score4

You are to output results in the format Firstname Surname TotalScore

Do not output a result for the line containing the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Smith,Anne,10,20,15,50 | Anne Smith 95 |
| Hornblower,Horatio,1,50,20,9 | Horatio Hornblower 80 |
| # |  |

# Laboratory Exercise 18 – Largest Score

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains 2 single word names and the game score for each name

Write out the names and scores in descending (high to low) order of score

Do not output a result for the line containing the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Sam 50 Tom 90 | Tom 90 Sam 50 |
| Sue 70 Harry 10 | Sue 70 Harry 10 |
| Horatio 50 Mary 50 | Horatio 50 Mary 50 |
| # |  |

# Laboratory Exercise 19 – Sums of lines of numbers

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a series of integers

Output the sum of the integers on each line of input

Do not output a result for the line containing the #

**(Hint Use hasNextInt)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 1 3 5 | 9 |
| 6 | 6 |
|  | 0 |
| 2 4 6 8 | 20 |
| # |  |

# Laboratory Exercise 20 – Sums of lines of valid numbers

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a series of integers and words

Ignore the words and only output the sum of the integers on each line of input

Do not output a result for the line containing the #

**(Hint Use hasNextInt)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 1 3 5 | 9 |
| Score 6 | 6 |
|  | 0 |
| 2 4 Cat 6 8 | 20 |
| # |  |

# Laboratory Exercise 21 – Test Scores

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains names and a series of test scores in the format   
Surname,Firstname, score1, score2, score3, score4 etc

You are to output results in the format Firstname Surname TotalScore

There could be any number of scores.

If there are no scores the total is of course 0

Do not output a result for the line containing the #

**(Hint Use hasNextInt)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Smith,Anne,10,20 | Anne Smith 30 |
| Bloggs,Joe | Joe Bloggs 0 |
| Hornblower,Horatio,1,50,20,9 | Horatio Hornblower 80 |
| # |  |

# Laboratory Exercise 22 – Chess Games

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line represents games of Chess between two players

The line consists of a series of letters A signifying a win by player A and B representing a win by player B.

For each line of input you are to output the total number of games won by player A and player B as shown in the sample output

There could be any number of games.

If there are no games the total is 0

Do not output a result for the line containing the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| ABAAABBA | A 5 B 3 |
|  | A 0 B 0 |
| AAAA | A 4 B 0 |
| # |  |

# Laboratory Exercise 23 – Min and Max numbers

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a series of integers

Write out the smallest and largest number in each line as shown in the example

* You can assume there will always be at least one number on the line
* Do not output a result for the line containing the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 8 2 5 9 6 | Min: 2 Max: 9 |
| 18 3 5 8 9 3 11 | Min: 3 Max 18 |
| 5 | Min: 5 Max: 5 |
| # |  |

# Laboratory Exercise 24 – Median

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a set of integers

The first number specifies how many numbers will be in the rest of the line.

For each line output the median of the remaining numbers on the line to a precision of 2 decimal places.

Do not output a result for the line containing the #

**(Hint Use an array)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 12 16 5 9 2 | 9.00 |
| 4 99 6 54 55 | 54.50 |
| # |  |

# Laboratory Exercise 25 – Sort Names I

Write a program that reads in a series of names and writes them out in alphabetical order.

The first line of input specifies how many names will follow.

**(Hint Use an array)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4 | Anatoly |
| Horatio | Anna |
| Anatoly | Evelyn |
| Evelyn | Horatio |
| Anna |  |

# Laboratory Exercise 26 – Grading Scores

Write a program which reads in a series of marks entered as floating point numbers (type double) and terminated by a -1.

Calculate the average mark

Output each mark on a line by itself and to a precision of 2 decimal places, and beside each mark output and appropriate grade: BELOW AVERAGE, AVERAGE or ABOVE AVERAGE

Marks range from 0 to 100 and there will be no more than 200 marks entered.

Do not output a result for the line containing the -1

**(Hint Use an ArrayList)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 34.5 | 34.50 BELOW AVERAGE |
| 17.0 | 17.00 BELOW AVERAGE |
| 80.2 | 80.20 ABOVE AVERAGE |
| 19.6 | 19.60 BELOW AVERAGE |
| 45.8 | 45.80 ABOVE AVERAGE |
| -1 |  |

# Laboratory Exercise 27 – Sort numbers

Write a program that reads in a series of positive integers terminated by a -1

Output the numbers in ascending numerical order – one number per line

Do not include the line with the-1 in the sorted output.

**(Hint Use an ArrayList)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 | 2 |
| 2 | 4 |
| 4 | 5 |
| -1 |  |

# Laboratory Exercise 28 – Sort Names II

Write a program that reads in a series of names terminated by a line containing a single # and writes the names out in alphabetical order.

There will be no more than 200 names.

Do not include the line with the # in the sorted output.

**(Hint Use an ArrayList)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Horatio | Anatoly |
| Anatoly | Anna |
| Evelyn | Evelyn |
| Anna | Horatio |
| # |  |

# Laboratory Exercise 29 – Test Scores II

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a series of integers

Write out each line of integers in ascending order

The integers will range from 0 to 99.

There will be at least 1 number per line and there will be no more than 20 integers per line.

Do not output a result for the line containing the #

**(Hint Use an ArrayList for each line)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 4 1 7 4 3 | 1 3 4 4 7 |
| 5 | 5 |
| 6 3 4 5 5 1 | 1 3 4 5 5 6 |
| 1 2 3 4 | 1 2 3 4 |
| # |  |

# Laboratory Exercise 30 Standard Deviation

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a set of real numbers (double)

For each line output the standard deviation of the numbers on the line, format the result to 3 decimal places

Do not output a result for the line containing the #

**(Hint Use the formula given, no need to use an Arrays or an ArrayList)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 12.3 1 179.123 31.0 0.1 | 76.167 |
| 25.5 11.6 9.2 | 8.800 |
| # |  |

Hint:

Standard Deviation can be calculated by the following formula

(note:  is the mean of the series)



# Laboratory Exercise 31 – Horizontal Name Sort

Write a program that reads in a series of lines terminated by a line containing only a single '#'

Each line contains a series of names separated by a comma

Write out each line of names in ascending order

There will be no more than 40 names per line.

Do not output a result for the line containing the #

**(Hint Use an ArrayList)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Harry,Sue,Mary,Bruce | Bruce,Harry,Mary,Sue |
| Tom | Tom |
|  |  |
| Jack, Jill | Jack, Jill |
| # |  |

# Laboratory Exercise 32 – File Processing

Write a program that reads in a series of lines from file /data/raw.txt processes them and writes them out to standard output

Each line contains a name and a series of scores for a game - all data on the line is separated by spaces

Output the name and the top three scores in descending order

There will always be at least 3 scores for each name in the raw data file

|  |  |
| --- | --- |
| Sample File | Sample Output |
| Mary 45 87 23 76 | Mary 87 76 45 |
| Joe 34 76 12 78 34 87 | Joe 87 78 76 |
| Anne 90 5 99 | Anne 99 90 5 |

# Laboratory Exercise 33 – Counting names

Write a program that reads in a series of lines from file /data/names.txt and stores them in an ArrayList of Strings.

The program should then continue to read a series of names from standard input until a line with only a # is reached

For each name it should output the name, a colon then the number of times that name is located within the ArrayList (as shown below).

Do not process the line with the #

|  |
| --- |
| Sample file “names.txt” |
| Harry |
| Sue |
| John |
| Mary |
| James |
| Mary |
| Mary |
| Anne |

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Mary | Mary: 3 |
| Joe | Joe: 0 |
| Anne | Anne: 1 |
| # |  |

# Laboratory Exercise 34 – Location of a name

Write a program that reads in a series of lines from file names.txt and stores them in an ArrayList of Strings.

The program should then continue to read a series of names from standard input until a line with only a # is reached

For each name it should output the name, a colon then the position of the first occurrence of the name in the array (0 based)

Output “**Not Found**” if that name is not located within the ArrayList

(see example).

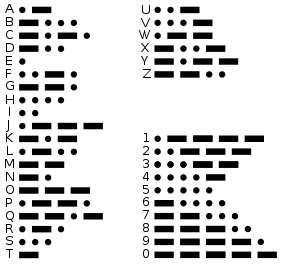
Do not process the line with the #

|  |
| --- |
| Sample file “names.txt” |
| Harry |
| Sue |
| John |
| Mary |
| James |
| Mary |
| Mary |
| Anne |

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Harry | Harry: 0 |
| Joe | Joe: Not Found |
| James | James: 4 |
| Anne | Anne: 7 |
| # |  |

# Laboratory Exercise 35 – Morse Code Generator

Create a text file containing letter code pairs based on the Morse Code table below. Use a full stop for a dot and a hyphen for a dash.



***Morse Code table from http://en.wikipedia.org/wiki/Morse\_code***

Write a program that

(a) reads your text file and creates a HashMap with the letter as the key and the code as the data.

(b) reads in a series of lines from standard input until a line with only a # is reached

For each line it should output the morse code for each word on the line with one space between each morse character and three spaces between each word.

Ignore case in the line from standard input.

Do not process the line with the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Hello World | …. . .-.. .-.. --- .-- --- .-. .-.. -.. |
| Today is sunny | - --- -.. .- -.-- .. ... ... ..- -. -. -.-- |
| # |  |

# Note: remember to submit your text file tooLaboratory Exercise 36 – Shopping List I

The file /data/pricelist.txt contains a list of goods in a diary and their prices.

Each line in the file contains a name (without spaces) and a price (a double)

Read the list and create a HashMap using the item name as the key and the price as the data

|  |
| --- |
| Sample file /data/pricelist |
| Bread 2.20 |
| Lemonade 1.95 |
| Butter 5.80 |
| Mince\_Pie 0.95 |
| Chocolate 3.50 |

From standard input read a shopping list

Each line contains an item name and a quantity required (as an integer)

For each line output the name of the item, the quantity, and @ symbol, its price (which is to be looked up in the map) to two decimal places and preceded by a $ symbol and the total price to 2 decimal places and preceded by a $ symbol.

The input data will end in a single line containing a #

Do not process the line with the # in it

If the shopping list item does not exist in the inventory then output   
the item name with the text “Not available” beside it

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Butter 2 | Butter 2@$5.80 $11.60 |
| Lemonade 2 | Lemonade 2@$1.95 $3.90 |
| Avacado 2 | Avacado Not available |
| Mince\_Pie 3 | Mince\_Pie 3@$0.95 $2.85 |
| # |  |

# Laboratory Exercise 37 – Phone Bill

The file /data/phonelist.txt contains a list of phone numbers and the person or company associated with them..

Each line in the file contains a number (without spaces) and a description of the person or company(which can have spaces)

Read the list and create a HashMap using the phone number as the key and the description as the data

|  |
| --- |
| Sample file /data/phonelist |
| 032149970 The Warehouse |
| 021211234567 Anna Jones |
| 0508446987 Hoyts |
| +6339876543 Australian Immigration |

From standard input read a list of phone numbers

Each line contains a phone number from a telephone account

For each line output the phone number followed by its description.

The input data will end in a single line containing a #

Do not process the line with the # in it

If the phone number does not exist in the inventory then output   
the phone number with the text “unknown” beside it

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 021211234567 | 021211234567 Anna Jones |
| 032149970 | 032149970 The Warehouse |
| +6339876543 | +6339876543 Australian Immigration |
| 021211234567 | 021211234567 Anna Jones |
| # |  |

# Laboratory Exercise 38 – Holiday

The file /data/travel.txt contains a list of travel distances between holiday locations.

Each line is in the form **From:To,Distance**

Where **From** and **To** are South Island locations, and **Distance** is an integer measured in km.

|  |
| --- |
| Sample file /data/travel.txt |
| Gore:Winton,65 |
| Invercargill:Gore,65 |
| Invercargill:Wanaka,246 |
| Winton:Invercargill,31 |
| Wanaka:Queenstown,70 |
| Queenstown:Invercargill,190 |

* You are to read a series of planned holidays from standard input
* The first line is the maximum distance you want to travel (because your time and finance is limited for the holiday)
* Following that are a series of lines each representing the travelling done in a holiday journey.
* Each holiday journey is in the form  
  **From:To,From:To,From:To,From:To**
* There may be up to 50 From:To journey pairs per line.

*The input data will end in a single line containing a #*

*Do not process the line with the # in it*

As you have limited time for your holiday your task is to only output journeys which total the specified distance or less and their total distance.

**(Hint Use a HashMap)**

|  |
| --- |
| Sample Input |
| 200 |
| Invercargill:Gore,Gore:Winton,Winton:Invercargill |
| Invercargill:Wanaka,Wanaka:Queenstown, Queenstown:Invercargill |
| # |

|  |
| --- |
| Sample Output |
| Invercargill:Gore,Gore:Winton,Winton:Invercargill 161km |

**Laboratory Exercise 39 – Shopping List II**

The file /data/pricelist.txt contains a list of goods in a diary and their prices.

Each line in the file contains a name (without spaces) and a price (a double)

Read the list and create a HashMap using the item name as the key and the price as the data

|  |
| --- |
| Sample file /data/pricelist |
| Bread 2.20 |
| Lemonade 1.95 |
| Butter 5.80 |
| Mince\_Pie 0.95 |
| Chocolate 3.50 |

From standard input read a shopping list

Each line contains a series of item names and the quantity of each required (as an integer)

For each line output the total price of the grocery order formatted to a precision of 2 decimal places and preceded by a $ symbol.

The input data will end in a single line containing a #

Do not process the line with the # in it

Assume all items exist in the inventory.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| Butter 1 Lemonade 2 | $9.70 |
| Chocolate 2 Mince\_Pie 3 Bread 1 | $12.05 |
| # |  |

# Laboratory Exercise 40 – CSV Files

A CSV file (Comma Separated Values) is a text file format where fields are separated by commas – often used for exporting and importing data.

Strings are often surrounded by quote marks.

Eg: "Jane","Doe",21,"Stormwind"

We can easily identify each field in Java by breaking up a line from the file based on commas.

The problem arises when a comma is part of a field

Eg: "Jane","Doe",21,"Trade District, Stormwind"

Write a program that reads in a series of lines from standard input terminated by a line containing only a single '#'

Each line contains fields separated by commas and string fields surrounded by double quotes.

You output a line to standard output where all commas in each line are replaced with a semicolon ';' character except those commas which are inside a quoted string

Do not output a result for the line containing the #

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| "Sam",10,20,"Bluff" | "Sam";10;20;"Bluff" |
| "Smith, Anne",20,"Gore" | "Smith, Anne";20;"Gore" |
| # |  |

# Laboratory Exercise Extra – Decode Decode!

Write a program that reads in a series of lines terminated by a line containing only a single '#'

All lines will contain only lowercase letters and spaces.

The line contains a coded message – the coding method is quite simple,

At the start of each line is a number (between 1 and 25 inclusive) each letter is shifted up through the alphabet by that number, wrapping around to the start if it goes beyond 'z'. For example with a code number of 5 'b' becomes 'g', and 'y' becomes 'd'. Spaces are not changed.

For each line output the decoded line without the code number.

Do not output a result for the line containing the #

**(Hint Use for – counter controlled loop)**

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| 5 Hello | Mjqqt |
| # |  |