Laboratory 4
60-212

Objectives:

- To understand how to use command line arguments.
- To understand arrays in Java.
- To write a static method that returns an array.

Download the partially defined class Lab4 from the course website. At the moment this contains only a main method.

Work to be done

a) Set the command line arguments as follows:
25 xyz +39 -43 abc "+   86" 34a6
This means your command line arguments are "25", "xyz", "+39", "-43", "abc",
"+ 86" and "34a6"
(**Note**: See instructions below on how to set command line arguments using Eclipse).

b) Write a method extractValidNumbers which has an array of strings as its parameter. Note that main calls extractValidNumbers using the array of command line arguments as the parameter. The method should return an array of int. The partial class definition I supplied does not include method extractValidNumbers which is your responsibility.

Hint: the method header can be as follows:
private static int[] extractValidNumbers(String commandLineArguments[])

(Questions you should be able to answer - why is the method private? Is that better than public? Why is the method static? What does it mean when the return type for the method is int[]?)

c) Develop the body of the method extractValidNumbers, so that the method

   i. checks, **using a regular expression**, each command line argument to find if the command line argument may be converted to an int. A command line argument may be converted to an int if the command line argument consists of an optional sign (i.e., + or -) followed by one of more digits. Some valid command line arguments that may be converted to an int are given below:
      25
      -432
      +5362
      Some invalid arguments are as follows:
      Abc
      43x
      "+   65"
      For the command line arguments shown below, notice that "25", "+39", "-43" may be converted to int.

      25 xyz +39 -43 abc "+   86" 34a6

      The remaining arguments may not be converted to int.

   ii. creates an array of int which will contain the int values corresponding to all command line arguments that may be converted to int. The method should return this array. This array of int should be **completely filled** with the int values that you computed from the command line arguments.
      Hint: Integer.parseInt(s) returns the int value corresponding to the string s.

      For instance, if the command line arguments are as follows:
      25 xyz +39 -43 abc "+   86" 34a6
      the array returned by the method should have 3 elements, containing 25, 39 and -43.

d) Make sure, using the main method that I supplied, that the output you get is as follows:

```
result[0] =25
result[1] =39
result[2] =-43
```

   Reminder: If you forget to specify the command line arguments, your program will not give any output. (Why?)

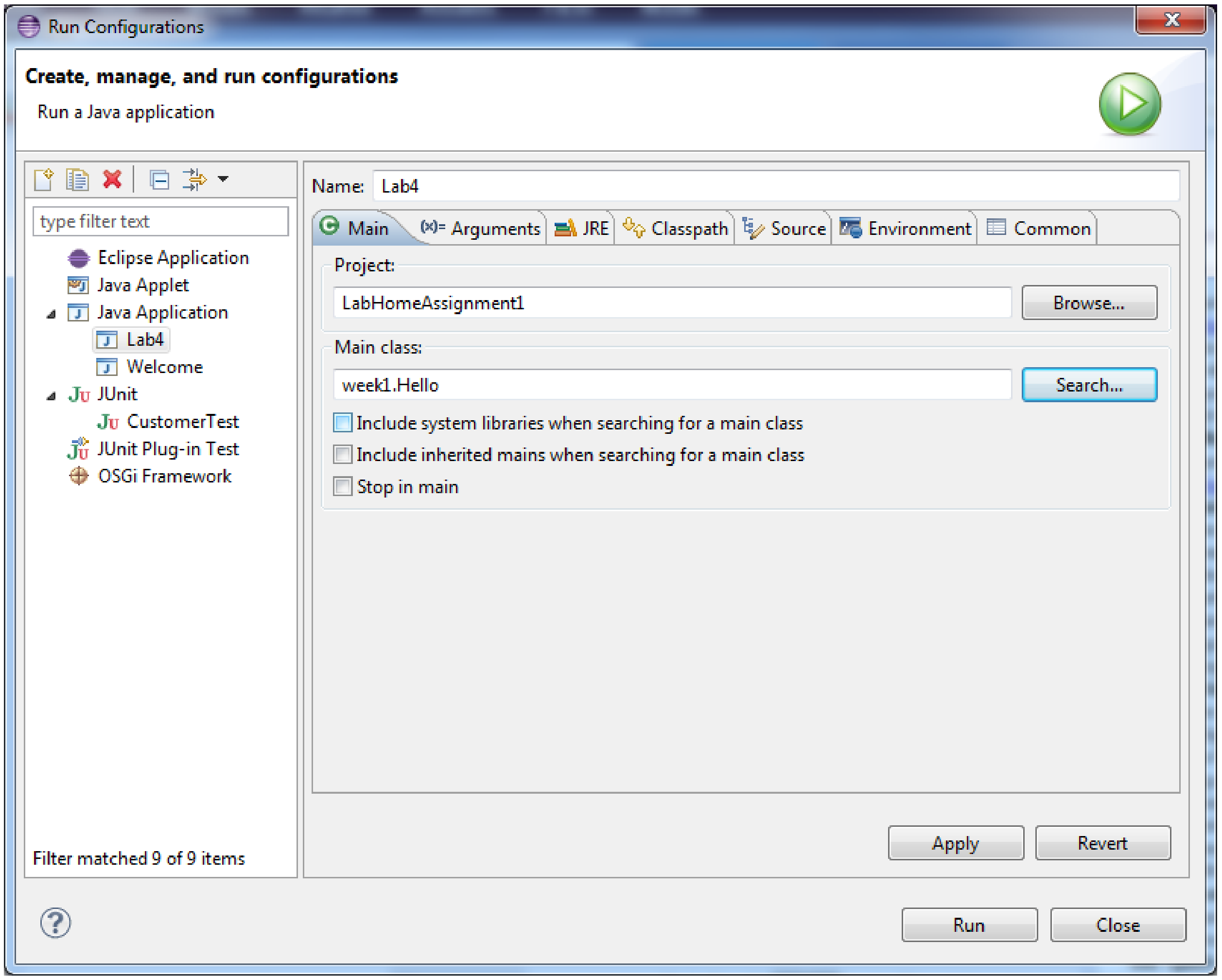
Quick tutorial on how to set command line arguments using Eclipse

Consider the following class Test which I defined under project LabAssign4 and package la4.:

```
public class Test {
        public static void main(String a[]){
                for (int i = 0; i < a.length; i++){
                        System.out.println("a[" + i + "] =" + a[i]);
                }
        }
}
```

Method main simply displays the values stored in the array a, the parameter of main. These values are called the command line arguments. You can specify, using Eclipse the values of the command line arguments in the following way.

1) Select Run, and then Run Configuration, giving the following window:


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: Lab4

Project: LabHomeAssignment1

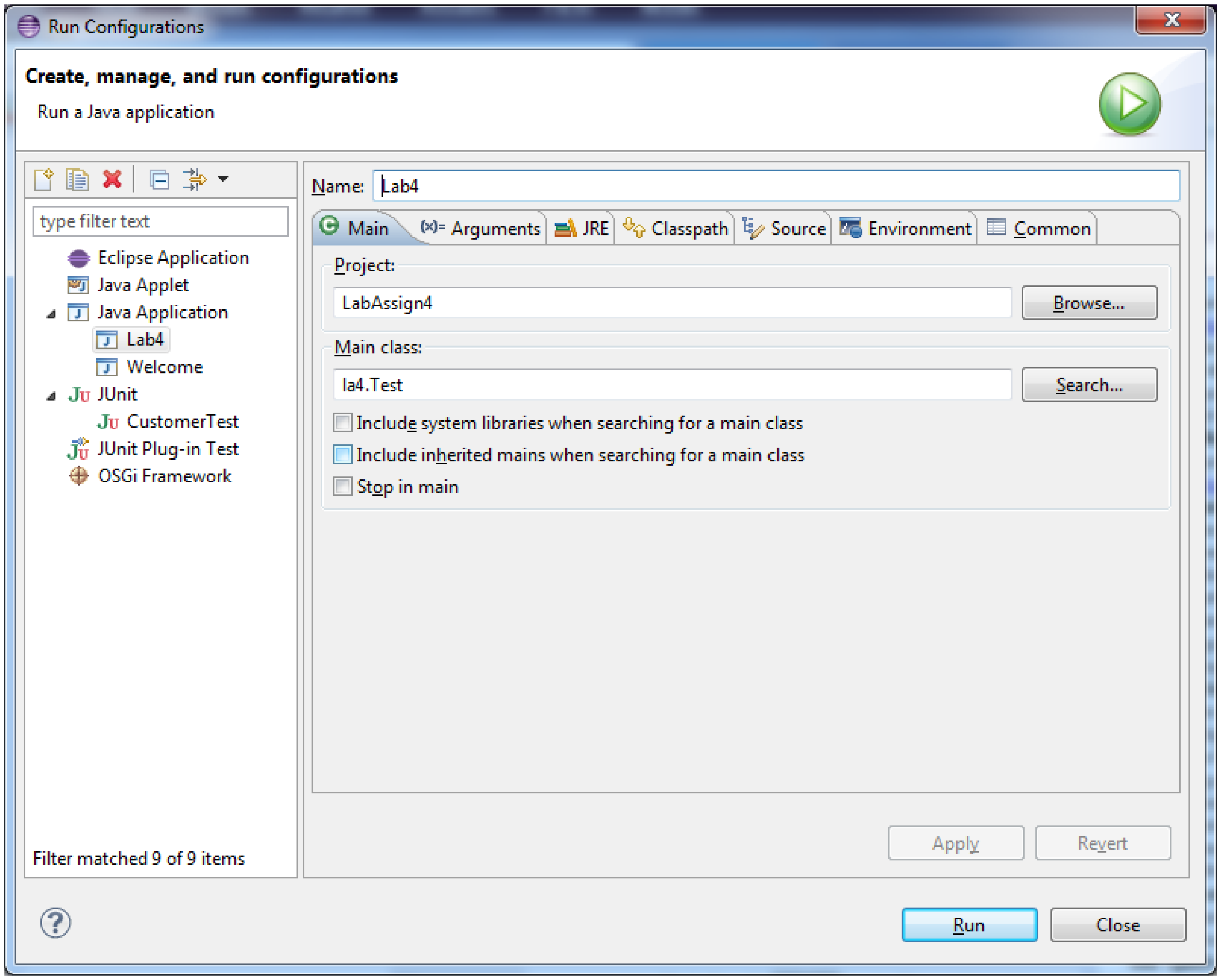
Main class: week1.Hello

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main

Filter matched 9 of 9 items

The name of the project and Main class will be different from what I have shown.

2) Press Browse and select the project your class is in. Then press Search and select the class where you defined your application. In my case the project was LabAssign4 and the class was Test. After I carried out the above steps, I got the following window:


Run Configurations

Create, manage, and run configurations

Run a Java application

type filter text

- Eclipse Application
- Java Applet
- Java Application
  - Lab4
  - Welcome
- JUnit
  - CustomerTest
- JUnit Plug-in Test
- OSGi Framework

Filter matched 9 of 9 items

Name: Lab4

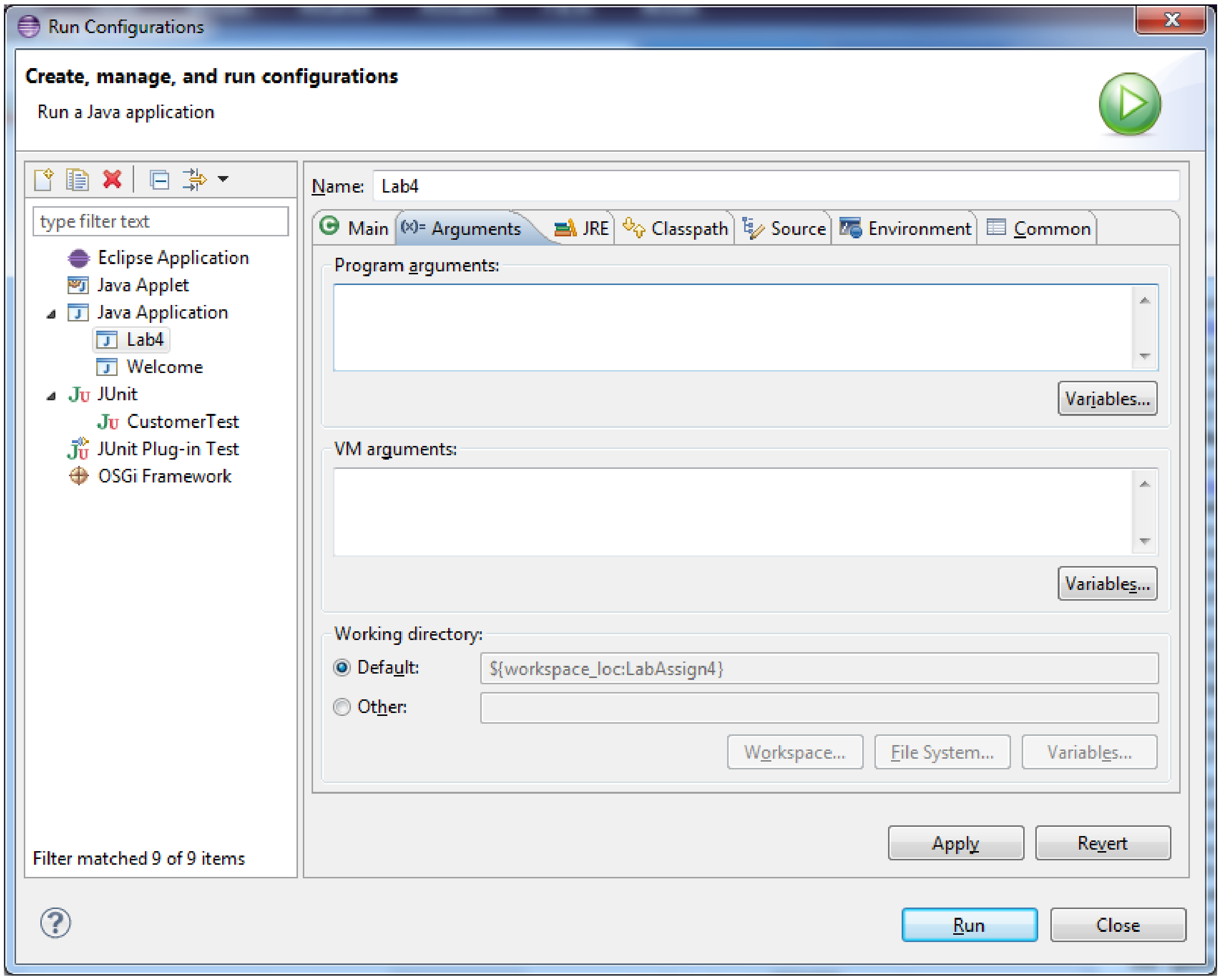
Main | Arguments | JRE | Classpath | Source | Environment | Common

Project:
LabAssign4 [Browse...]

Main class:
la4.Test [Search...]

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main

3) Now select Arguments (the second tab) giving the following window:


Run Configurations

Create, manage, and run configurations

Run a Java application

Name: Lab4

Program arguments:

VM arguments:

Working directory:

(●) Default: ${workspace_loc:LabAssign4}

( ) Other:

4) In the box under Program arguments give your command line arguments one by one, separated by spaces. Each commend line argument is a string. If a command line argument contains one or more spaces, put the argument inside a pair of double quotes.

For instance, it is valid to specify the following:
Hello how   "are you doing"   "25" "+   32 "

After specifying all command line arguments, press Apply and then press Close.

This means I have specified 5 command line arguments as follows:
"Hello"
"how"
"are you doing"
"25"
"+   32 "

Now if you run your application as usual (Run -> Run As 1 Java Application) you get the following output:

```
a[0] =Hello
a[1] =how
a[2] =are you doing
a[3] =25
a[4] =+    32
```

Note: It is convenient to use this feature to supply a small amount of input data to your application since you don’t have to specify the input every time you run your program.