**Lab 1: Introduction to OOP using Java Programming**

**Introduction**

This laboratory exercise introduces Object Oriented Programming using JAVA. The topics covered in this exercise are Classes, Objects, Java Methods and Attributes.

**Objective**

At the end of this laboratory session the students should be able to.

1. Create a Java Class and how to define Class attributes and methods.
2. Use objects in JAVA programming.
3. Implement multi class program and how each Class can communicate to one another.

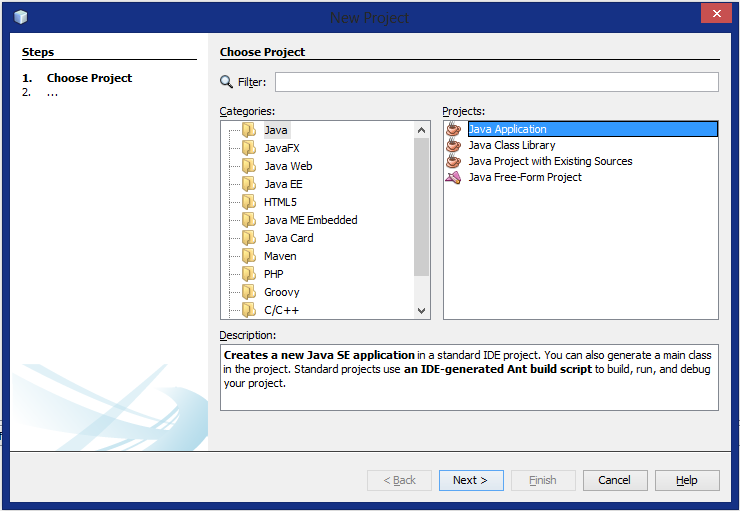
**Tools/Software Requirements**

The following software are needed to be installed in the computer system to test and run the given programming examples, and to solve the programming exercise.

1. NetBeans IDE (Version 7.2 or higher).
2. Java Development Kit (JDK Version 7 or higher).

**Description**

This laboratory exercise contains two examples. The first exercise is the Welcome.java that will show the basic Java construct, the program will just display “Welcome to Java!” in the output window. The second exercise is the Student.java and Student\_Records.java that will show how Class attributes and methods are used in Java programming, it will also show how these Classes can communicate with each other using objects and methods.

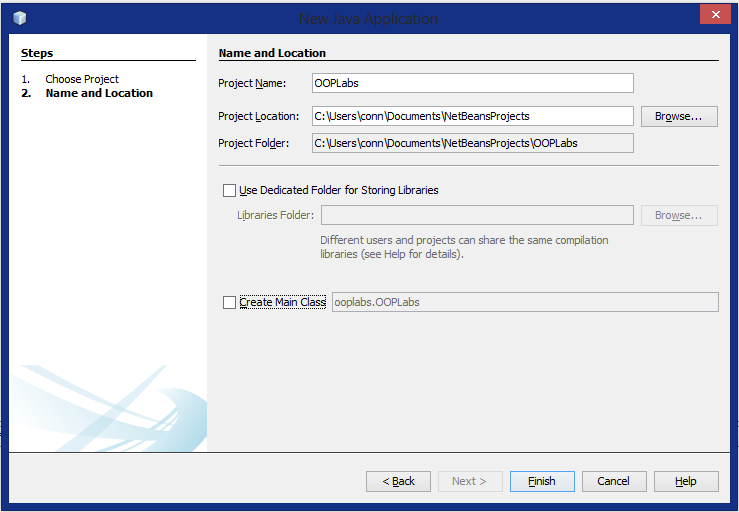


**Exercise 1: Welcome.java**

The welcome Java program uses the System.out.println(“”); method to display the text in the output window. Below are the steps to create this program.

**Step 1: Creating Project and Java File**

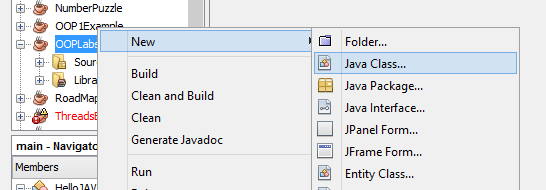
* Go to the Menu bar and press File > New Project, and the window below will be displayed on your screen. Or use the shortcut, press Ctrl+Shift then N.
* In this window you need to select Java in the Categories and Java Application in the Project as shown in the screenshot then press Next button below. After this move to the next screenshot.

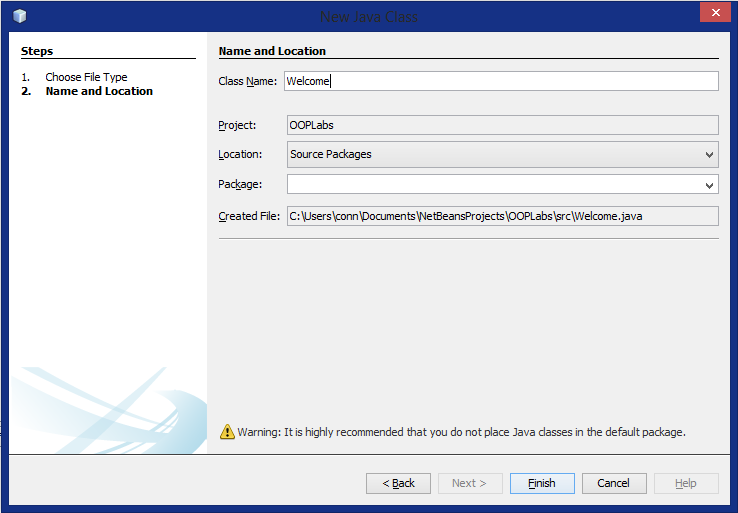


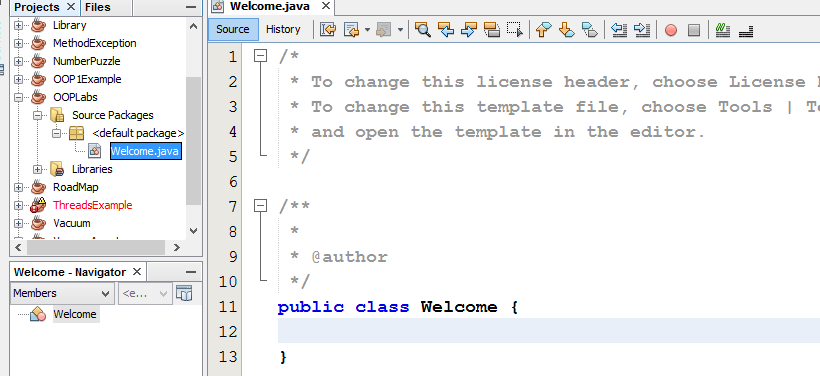
* In this window you need to set the Project Name to OOPLabs as shown in the screenshot.

If this part has a checked, remove it by clicking the box.

After this click the Finish Button below, then proceed to the next screenshot.

* Once the project is already created, you can now see it in the project explorer window.
* To create a new Java file, right click on the Project and press New > Java Class and the next window will be shown on the screen.



* In this window you need type the Class Name, use Welcome as the class name as shown in the screenshot then press Finish button to create the file.
* Once the Java file is created you will have this window on your screen. As you can see the Class Welcome is already created therefore you can now start by writing the Java statement to print the text Welcome to Java!
* **Reminders**: The class name must always be the same as the File name that you have entered in the previous window. Don’t try to change the Class name in the editor window because the program will not work. To change the name you will have to change first the name of the file then change the name of the Class in the code with this you can assure that the File name has the same name as the Class name.

**Step 2: Writing, Testing and Running Java program**

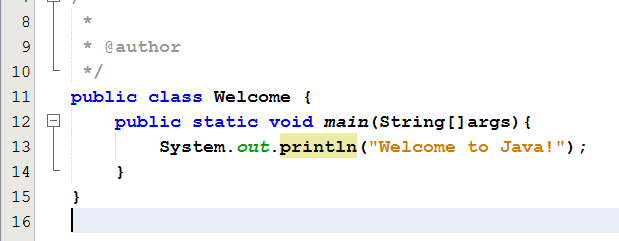
Type the following Java Code inside the Class Welcome.

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

**System.out.println("Welcome to Java!");**

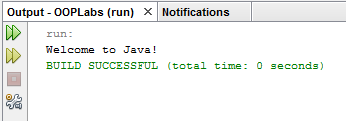
**}**

See the screenshot below for the actual program.

* Line 12 in the code: public static void main(String[]args) is the main method. This is the starting point of the Java program. A class that has this method in the project will be executed first. If a class doesn’t have this method it cannot be executed on its own.
* Line 13 in the code: System.out.println(“Welcome to Java!”); this is a method that is used to display text in the output window.
* To test the program for errors press Alt+F6 (Shortcut for Test File). If there is an error check the output window for information.
* To run the program press Shift+F6 (Shortcut for Run File).

The next window shows the output of the program.

**Program Output**



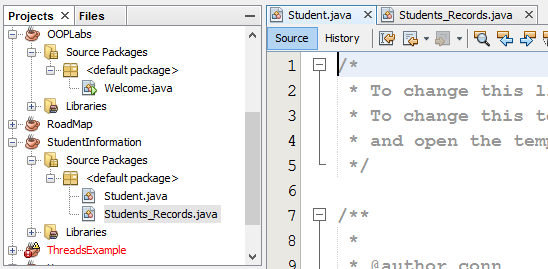
**Exercise 2: Student.java and Students\_Records.java**

This program shows how to declare Classes and define their attributes and methods, it will also show how Classes can communicate to each other using Objects and their methods.

*Note: The step in creating project and Java file is no longer included in this example because the process is the same. Only instructions are provided for you to follow.*

**Step 1: Create Project and Java file**

* Create Project named StudentInformation. (Note: uncheck the Create Main Class).
* The same procedure at Example 1, create a file named Student in the StudentInformation Project.
* Again to the same process and create another Java file named Students\_Records.
* When you are done, your project and editor tab should look like the screenshot in the next page.

****

There should be 2 tabs at top of the editor window showing the Files that you have created.

In your project explorer Check if you have the same arrangement of files as shown in this screenshot

* Ones you are done, you can now write the code for each of the Class.

**Step 2: Coding the Students.java**

1. **CLASS ATTIBUTES**

* First Click on the Student.java file in the editor tab to access the file and start writing the Java code.
* The next thing you will have to do is declare Class Attributes and Methods.
* **Class Attributes** are information that belongs to an object.
  + To declare Class Attributes you must first identify the data type of each attribute. The table below shows the Student attributes and its type.

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data type** | **Description** |
| student\_id | int | Student Id number |
| student\_name | String | Student Name |
| Student\_year | int | Student Year Level |
| student\_department | String | Student Department |

* To add attributes to the Class the following syntax needs to be followed.
  + Data\_type Attribute name;
    - Example: int student\_id;
* Include the following attributes inside the Student Class.
  + **int student\_id;**
  + **String student\_name;**
  + **int student\_year;**
  + **String student\_department**
* Your code now should look like this.

**public class Student {**

**int student\_id;**

**String student\_name;**

**int student\_year;**

**String student\_department;**

**}**

* These attributes will serve as container of data for every object that you will declare using this Class, if multiple objects is created each object will have its own set of attributes.

1. **CONSTRUTOR**

* Since you already have attributes the next thing that you have to do is create a method to assign values to these attributes.
* There are different ways to do this. First you can create a method to assign a value to an attribute of the class, we call this mutator methods. The second one is to create a **constructor** with arguments to acquire a value and pass it to the Class attributes.
* For this example, constructor will be used to acquire data and pass it to the Class attributes. Below is constructor to be used in the program.
  + **Student(int id, String name, int year, String department){**

**this.student\_id = id;**

**this.student\_name = name;**

**this.student\_year = year;**

**this.student\_department = department;**

**}**

* Write this code inside the Student Class after the attributes declaration.
* *Needs to remember when using a constructor*.
  + Constructor must have the same name as the Class.
  + It doesn’t have a return type like any other methods.
  + Constructor may or may not have arguments.
  + Constructors can only be executed whenever an object of a class is created. Later in this exercise we are going to use objects in the program.
* Statements inside the constructor Expalnation:
  + this.student\_id = id; > this line of code assigns the value of id to the student\_id. The “this” keyword before student\_id is used to refer to the current class attribute or method. Although the this keyword is not necessary in the program. You can actually re-write the code to
    - student\_id = id; with this code the effect will be the same.
  + this.student\_name = name; > assigns the name value to the student\_name.
  + this.student\_year = year; > assigns the year value to the student\_year.
  + this.student\_department = department > assigns department value to the student\_department.
* After you are done with this, you code should look like this.

**public class Student {**

**int student\_id;**

**String student\_name;**

**int student\_year;**

**String student\_department;**

**Student(int id, String name, int age, String department)**

**{**

**this.student\_id = id;**

**this.student\_name = name;**

**this.student\_year = age;**

**this.student\_department = department;**

**}**

**}**

1. **METHODS**

* The next part in coding is declaring the methods.
  + Method are program modules that performs a specific task.
    - Syntax:

modifier return\_type method\_name(optional arguments){

program statements;

}

* + - Example:

public int getID(){

return student\_id;

}

* The methods that are used in this example is called accessor methods. These methods are used to accessed the content of the attributes.
* Below are the methods used in this example and their description.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public int getID() | Returns the ID of the student. |
| public String getName() | Returns the Name of the student. |
| public int getYear() | Returns the Year Level of the student |
| public String getDepartment() | Returns the Department of the student. |

* Accessor methods uses the “get” prefix when declaring method names, this will signify that this method is an accessor.
* With this write the following code inside the Student Class after the constructor. Your code should look like this.

public class Student {

int student\_id;

String student\_name;

int student\_year;

String student\_department;

Student(int id, String name, int age, String department)

{

this.student\_id = id;

this.student\_name = name;

this.student\_year = age;

this.student\_department = department;

}

public int getId(){

return student\_id;

}

public String getName(){

return student\_name;

}

public int getYear(){

return student\_year;

}

public String getDepartment(){

return student\_department;

}

}

* Notice the statement inside each accessor method, the “return” keyword is used to return a value from the method to the caller (the method who calls for the declared methods in this class). Methods can be executed only if it is called by another method.
* After you finish coding the Student.java, save the File. Do not run this file it will not give you any output since it does not have the main method but you can compile this file to check if there is an error(Note: The NetBeans IDE automatically checks for syntax error in the code. If there is an error it will be marked red).
* Now the Student. java Class is completed its time to move on to the Students\_Records.java.

**Step 3: Coding the Students\_Records.java**

* This class will serves as the main class wherein the main method should be declared in this Class.

1. **Arraylist**

* The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed. Comparing to an array which has a finite set of data. ArrayList can grow according to the number of object saved in the list.
* Below is the Java code on how to use ArrayList in the program.
  + ArrayList<Student> list = new ArrayList<Student>();
    - Where Student is the type of the List which refers to the Student Class in the Project.
  + After including this code into your Students\_Records Class, your code should look like this.

**import java.util.\*;**

**public class Students\_Records {**

**ArrayList<Student> list = new ArrayList<Student>();**

**}**

*Note: You need to import java.util.\* to use the Class ArrayList and later on the Scanner Class that will be used to accept an input.*

1. **Main method**

* After the ArrayList is already declared the next thing to do is to create the main method. Below is the code for the main method

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

**new Students\_Records();**

**}**

* The program statement inside the main call for its own constructor, usually when we wanted to execute constructor we instantiate an object. Below is another way to execute the content of the constructor by declaring an object
  + Students\_Records studRec = new Students\_Records();
* Adding this part of the program after the declaration of the ArrayList, your code should look like this.

**import java.util.\*;**

**public class Students\_Records {**

**ArrayList<Student> list = new ArrayList<Student>();**

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

**{**

**new Students\_Records();**

**}**

**}**

1. **Constructor**

* In this program the constructor is used again but not to acquire data, but rather to execute Java statements.
* Since the constructor will not accept any values, the arguments must be left blank. Below is the code for the Students\_Records constructor and explanation.

**public Students\_Records()**

**{ *//Declare a Scanner object to use methods to accept data***

***//From the user***

**Scanner input = new Scanner(System.in);**

**for (int i = 0; i < 3; i++) { //*Loop statement to repeat code 3 times***

***//Display message in the output window***

**System.out.print("Enter Student\_Id: ");**

**int id = input.nextInt();*//Accepts data from the user and store to id***

**//*Display another message in the output window***

**System.out.print("Enter Student\_Name: ");**

***//Accepts data from the user and store to name***

**String name = input.next();**

**//*Display another message in the output window***

**System.out.print("Enter Student\_Year: ");**

***//Accepts data from the user and store to year***

**int year = input.nextInt();**

**//*Display another message in the output window***

**System.out.print("EnterStudent\_Department: ");**

***//Accepts data from the user and store to department***

**String department = input.next();**

**//*Add id,name,year and department to ArrayList***

**list.add(new Student(id, name, year, department));**

**}**

***//Display the content of the ArrayList***

**for(int ctr=0;ctr<3;ctr++){**

**//*Classes communicate to one another by using objects to access methods from the Class***

**System.out.print(" "+list.get(ctr).getId());**

**System.out.print(" "+list.get(ctr).getName());**

**System.out.print(" "+list.get(ctr).getYear());**

**System.out.print(" "+list.get(ctr).getDepartment());**

**System.out.println("");**

**}**

* **The red lines in the code are called comments, this will not affect the output of your program since it will not be included when the program executes.**
* **Below is the complete program for the Students\_Records.java**

**import java.util.\*;**

**public class Students\_Records {**

**ArrayList<Student> list = new ArrayList<Student>();**

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

**{**

**Students\_Records studRec = new Students\_Records();**

**}**

**public Students\_Records()**

**{**

**Scanner input = new Scanner(System.in);**

**for (int i = 0; i < 3; i++) {**

**System.out.print("Enter Student\_Id: ");**

**int id = input.nextInt();**

**System.out.print("Enter Student\_Name: ");**

**String name = input.next();**

**System.out.print("Enter Student\_Year: ");**

**int year = input.nextInt();**

**System.out.print("EnterStudent\_Department: ");**

**String department = input.next();**

**list.add(new Student(id, name, year, department));**

**}**

**for(int ctr=0;ctr<3;ctr++){**

**System.out.print(" "+list.get(ctr).getId());**

**System.out.print(" "+list.get(ctr).getName());**

**System.out.print(" "+list.get(ctr).getYear());**

**System.out.print(" "+list.get(ctr).getDepartment());**

**System.out.println("");**

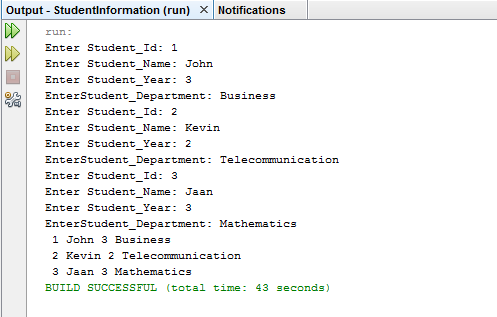
**}**

**}**

**}**

* Test the Program to see if there is an error, if no errors where found Run the Program to see the output.
* Below is an example screenshot of the program.

**Program output**

****

**Deliverables**

* After the student have verified the output of the program is correct, the program should:
  + Show their work to their instructor for verification.
  + Upload the Project folder in Edmodo.com in zip or rar format.
    - The project folder of NetBeans can be found at: Documents\NetBeansProjects\.
    - Locate you project name, compressed it using zip or rar before uploading.