People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'hamed Bouguerra - Boumerdes
Institute of Electrical and Electronics Engineering
Electronics Department

Option: Computer engineering
LAB REPORT n°1
OCTOBER 21, 2022

Title

EE423: Advanced Programming/ Writing, Compiling and Executing a Java Program

Authored by :
Agli Wafa
Zidane Aymen

Instractor :
Dr.A Zitouni

Session : 2022/2023

Contents

1 Introduction

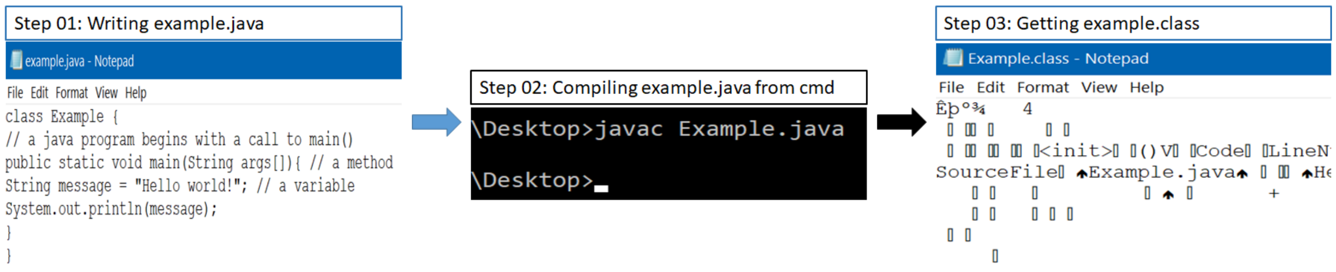
This lab is divided into two parts; in the first we learn how to compile and debug a code as well as types of errors that may occur during coding.
The second part introduces some of main concepts of OOP : object, class and encapsulation as well as other definitions will be discovered in the coming sections.

2 Part 01: Java environment

2.1 Setting up the environment

Installing the jdk; The Java Development Kit (JDK) is a software development environment used to develop Java applications. It contains JRE + development tools. The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc.
Installing eclipse IDE; An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development.

2.2 Writing, Compiling, Debugging the java code


Step 01: Writing example.java
example.java - Notepad
File Edit Format View Help
Step 02: Compiling example.java from cmd
Step 03: Getting example.class
Example.class - Notepad
File Edit Format View Help

```
class Example {
// a java program begins with a call to main()
public static void main(String args[]){ // a method
String message = "Hello world!"; // a variable
System.out.println(message);
}
}
```

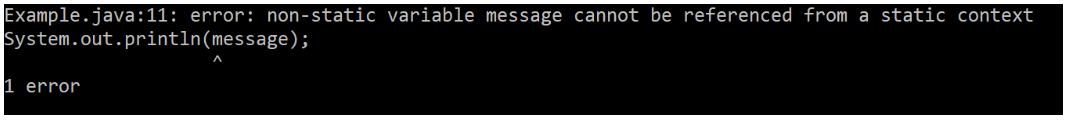
```
\Desktop>javac Example.java

\Desktop>_
```

```
Êþº¾    4
<init>  ()V  Code  LineN
SourceFile  Example.java  He
```

Figure 1: Process of creating and compiling a .java file

After executing the code we got the following error:

```
/* This is a simple Java program */
class Example {
// a java program begins with a call to main()
String message = "Hello world!"; // solution 1:
// declare it as a class
// variable
public static void main(String args[]){
//static String message = "Hello world!"; //solution 2:
// mark it static
System.out.println(message);
}
}
```



```
Example.java:11: error: non-static variable message cannot be referenced from a static context
System.out.println(message);
                   ^
1 error
```

We can't access a non-static variable from a static context. our variable(non-static) is outside of main method(static). We have two solutions: One is to define message variable as static outside the main:

```java
/* This is a simple Java program */
class Example {
// a java program begins with a call to main()
static String message = "Hello world!";
public static void main(String args[]){
System.out.println(message);
}
}
```

Two is to declare it inside the main:

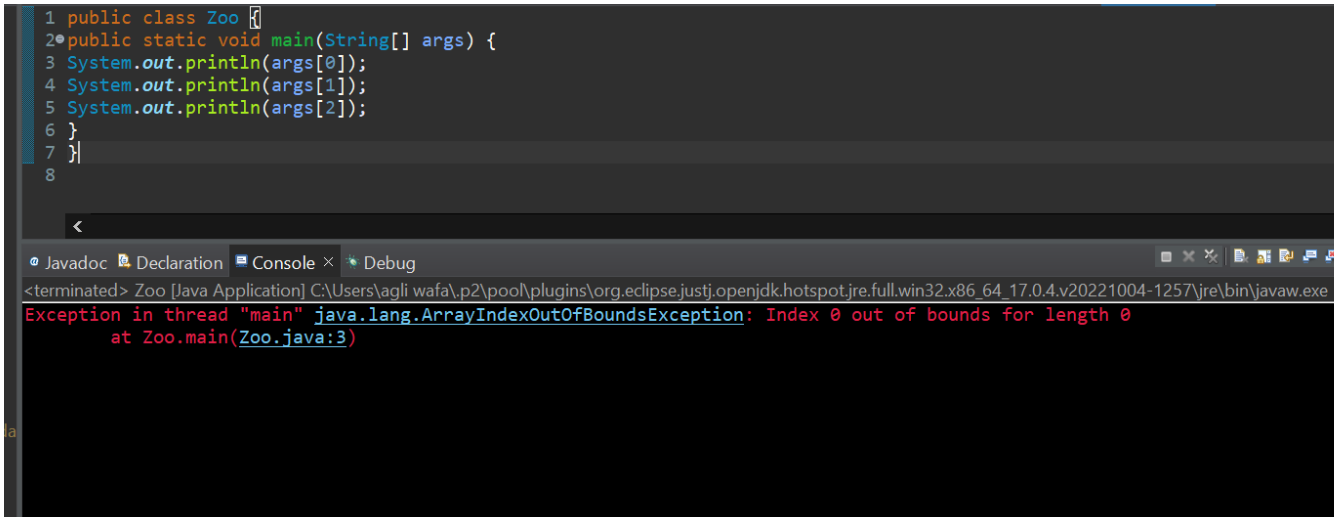
```java
/* This is a simple Java program */
class Example {
// a java program begins with a call to main()
public static void main(String args[]){
String message = "Hello world!"; // solution 2:
// declare it as a class
// variable
System.out.println(message);
}
}
```

2.3 Types of Errors in Java

Programming errors remain undetected until the program is compiled. Mainly errors in java are classified in two categories :

- **compilation errors**: Compiling Errors; refer sometimes to syntax errors such as a missing semi-colon at the end of a statement or a missing bracket, class not found, etc. these errors can be easily detected by the compiler.
- **Run-time errors** :

  After running the code it stops execution in line 3 and we get the following message: index 0 out of bounds for length 0 means that we are printing parameters which we did not pass.



```java
public class Zoo {
public static void main(String[] args) {
System.out.println(args[0]);
System.out.println(args[1]);
System.out.println(args[2]);
}
}
```

```
<terminated> Zoo [Java Application] C:\Users\agli wafa\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\jre\bin\javaw.exe
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 0 out of bounds for length 0
	at Zoo.main(Zoo.java:3)
```

In fact a run-time errors are detected during the execution of the program. these type of errors are discovered when **entering an invalid data** (ex; if the user inputs a data of integer when the computer is expecting another type, dividing by zero or Runtime Error caused by Assigning Value from an array using an index which is greater than the size of the array (ie. the above example)). During compilation. The JVM (Java Virtual Machine) detects these errors while the program is running.

3 Part 02: Objects, Classes, Encapsulation: First OOP concepts

3.1 Manipulating Point class

the diagram down-billow (Figure 2) represents a Class named **Point** with two attributes **X** and **Y** (that defines any point in 2D plan), these attributes set as **private** in order to restrict there use to the Point class only. Also we define for each attribute a get method with a return value equivalent to the attribute type so we can read it's value, as well as a set method capable to assign another value. we have created also two types of the constructor one without parameters and one by passing parameters. The print method displays the attributes in deserved format.

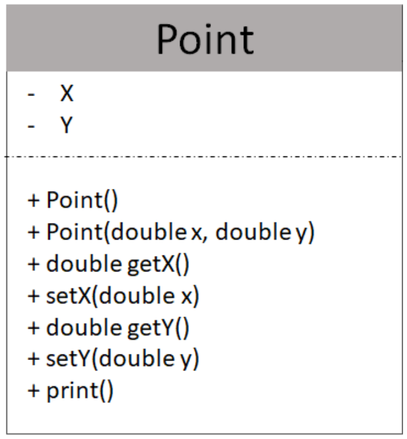


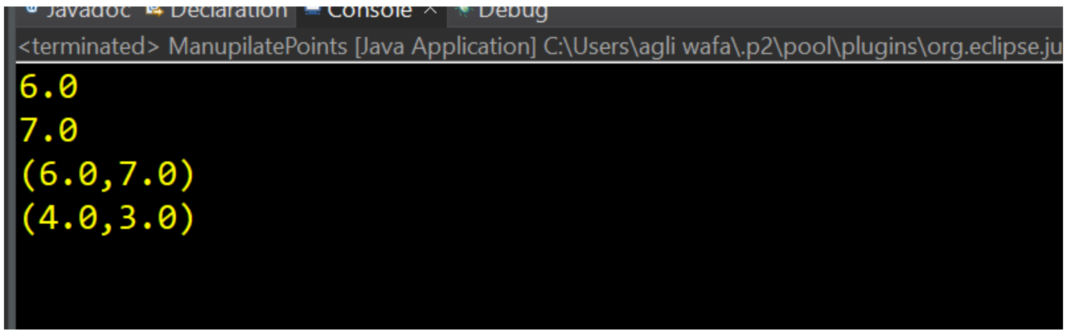
Figure 2: Diagram represents Point class

```java
package ExercisesOfLab1;
public class Point {
       private double x, y;
       public Point(double xv, double yv){x = xv; y = yv;} //constructor with params
       public Point(){} //constructor without params
       public double getX(){return x; } //return X's value
       public double getY(){return y; } //return Y's value
       public void setX(double newX){x = newX;} //asign new value to x
       public void setY(double newY){y = newY;}//asign new value to y
       /**
       display x and y in the format : (x,y)
       **/
       public void Print() {
      System.out.println("(" +x +"," +y +")");
      }
  }
```

In order to use point class we create *ManupilatePoints* class where we instantiate (create objects from Point).

```java
public class ManupilatePoints {
  public static void main(String[] args) {
    Point pt1= new Point(); // x, y initialized by 0
    pt1.setX(4); // x = 4
    pt1.setY(3.5); //y=3.5
    Point pt2 = new Point(6, 7); //intialization through constractor

    System.out.println(pt2.getX()); //print x value
    System.out.println(pt2.getY());

    pt2.Print(); //calling the print method

        pt1=null; //destruct the object
        pt2=null;
  }
}
```

After running the program we get the results shown in Figure2:



```
<terminated> ManupilatePoints [Java Application] C:\Users\agli wafa\.p2\pool\plugins\org.eclipse.ju
6.0
7.0
(6.0,7.0)
(4.0,3.0)
```

Figure 3: the results of first exercise

We notice that: we can use the constructor Point() even before it's creation in the class Point, and it initialize x and y to 0.0. When we want to initialize by different values we had to create our constructor by passing desired parameters.
The object passes by three stages: construction (born), use (example printing a point coordinates) and then destruction(dies) by making it illegible to garbage collector

3.2 Rectangle Class

```java
package ExercisesOfLab1;
public class Rectangle {
    private double width, height; //defining attributes
  public double getWidth() {return width;}
  public double getHeight() {return height;}
  public void setWidth(double width) {this.width= width;}
  public void setHeight(double height) {this.height= height;}
  //constructor
   public Rectangle(double width, double height)  {
         setWidth(width);
         setHeight(height);}
    //method to calculate the rectangle area
    public double getArea() {return width*height;}}
```

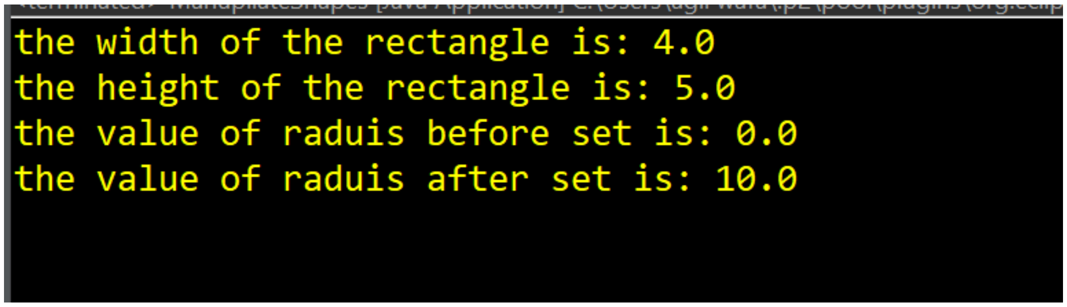
We notice that: in the constructor we called setters methods in order to initialize the instance variables width and height. However, we should ask what is the difference between a direct assignment and using setters inside constructor.

3.3 Circle Class

```java
package ExercisesOfLab1;
public class Circle {
  private double raduis; //the attribute of the circle
  public double getRaduis() {return raduis;} //the getter
  public void setRaduis(double newRaduis) {raduis= newRaduis;}
  //constructor capable to set the value of raduis to the specified value
  public Circle (double Raduis){
    setRaduis(Raduis)}
  //default constructor
  public Circle (){}

}
```

3.4 Main Class

```java
package ExercisesOfLab1;
public class ManupilateShapes {
  public static void main(String[] args) {

      Rectangle r= new Rectangle(4, 5);
      //to print the width and height we use get()
        System.out.println("the width of the rectangle is: " +r.getWidth());
        System.out.println("the height of the rectangle is: " +r.getHeight());

        obj=null; //GC

        /**
        if we use default constructor we use set to initialize by a value
        different than 0.0**/
        Circle C= new Circle();
        System.out.println("the value of raduis before set is: " +C.getRaduis());
        C.setRaduis(10);
        System.out.println("the value of raduis after set is: " +C.getRaduis());
  }
}
```



```
the width of the rectangle is: 4.0
the height of the rectangle is: 5.0
the value of raduis before set is: 0.0
the value of raduis after set is: 10.0
```

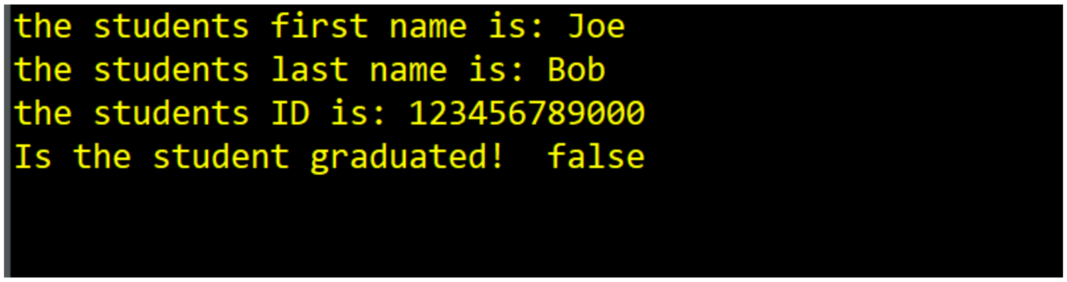
Figure 4: the results of fourth exercise

3.5 Student Class

```java
package ExercisesOfLab1;

import java.util.Date;

public class Student {
  private String firstname, lastname;
  private long ID; //takes 12 digits
  private Date dateofbirth; //usin Date type
  private boolean graduated;

    public Student(String firstname,String lastname, long ID, boolean graduated) {

       this.setFirstname(firstname);
       this.setLastname(lastname);
       this.setID(ID);
       this.setGraduated(graduated);
       }

    public boolean isGraduated() {return graduated;}
    public void setGraduated(boolean graduated) {this.graduated = graduated;}
    public String getFirstname() {return firstname;}
    public void setFirstname(String firstname) {
      this.firstname = firstname;
    }
    public long getID() {return ID;}
    public void setID(long iD) {ID = iD;}

    public Date getDateofbirth() {return dateofbirth;}
    public void setDateofbirth(Date dateofbirth) {this.dateofbirth = dateofbirth;}
    public String getLastname() {return lastname;}
    public void setLastname(String lastname) {this.lastname = lastname;}
    }
```

```java
    package ExercisesOfLab1;

    public class Teststudent {

      public static void main(String[] args) {
        Student S= new Student("Joe","Bob",123456789000L,false);

           System.out.println("the students first name is: " +S.getFirstname());
           System.out.println("the students last name is: " +S.getLastname());
           System.out.println("the students ID is: " +S.getID());
           System.out.println("Is the student graduated!  " +S.getGraduated());
          }

    }
```

after running the program we get:



```
the students first name is: Joe
the students last name is: Bob
the students ID is: 123456789000
Is the student graduated!  false
```

Figure 5: the results of fifth exercise

we notice that: we cannot use int type for attributes have digit numbers greater than 10, so we used long rather.
We removed dateofbirth in constructor since objects doesn't need it.

4 Conclusion

We conclude from this lab that Object-Oriented Programming is a paradigm to design a program using classes and objects. Where the class is an abstract representation of real-world entities (objects). OOPs provides data encapsulation; even global data cannot be accessed anywhere unlike procedural programming.