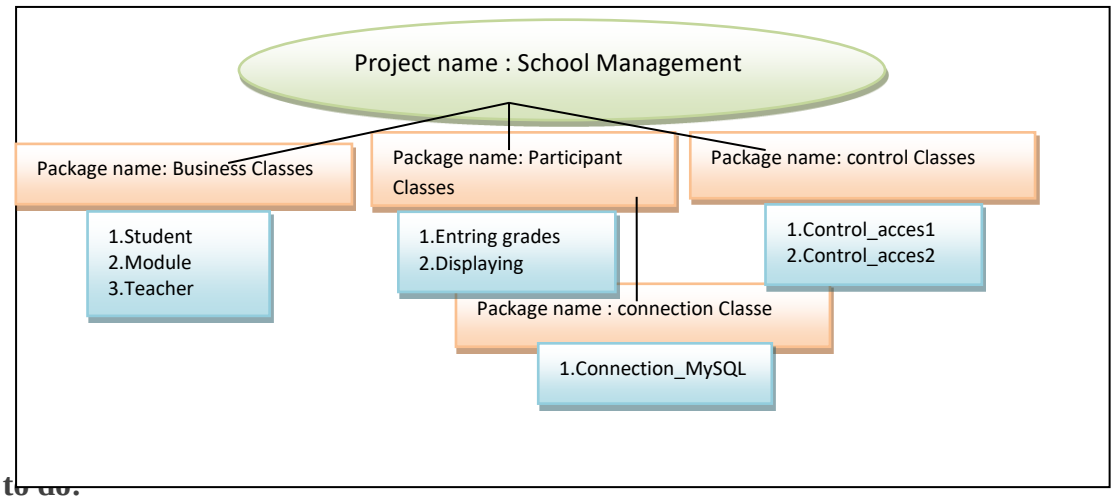
LAB1
(Creating Packages)

Statement: A school management application named " School_Management " includes 4 packages and 8 classes according to the following structure:



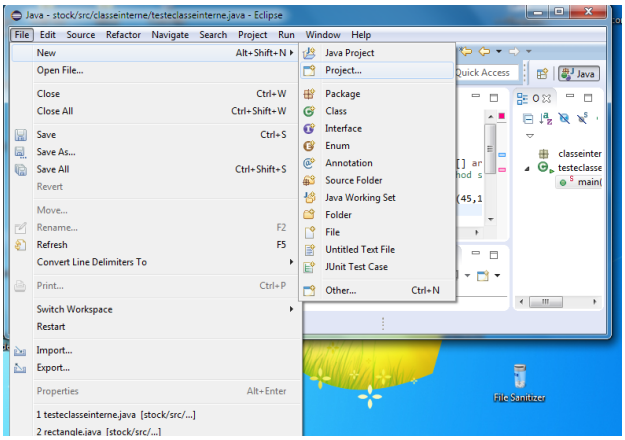
Task to do:

I. Launch Eclipse IDE

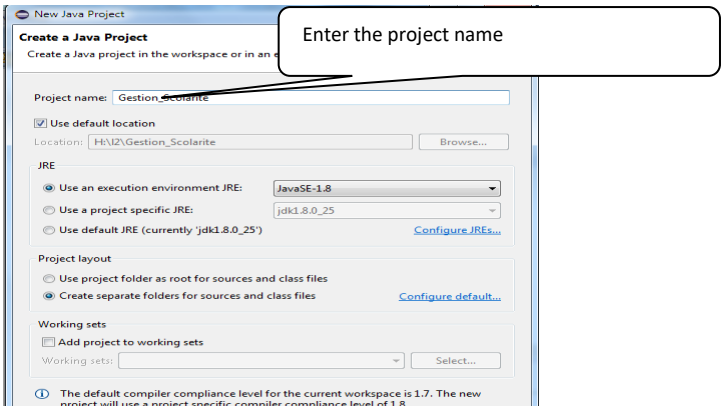
II. Create the project " School_Management " following the above schema.

Answer

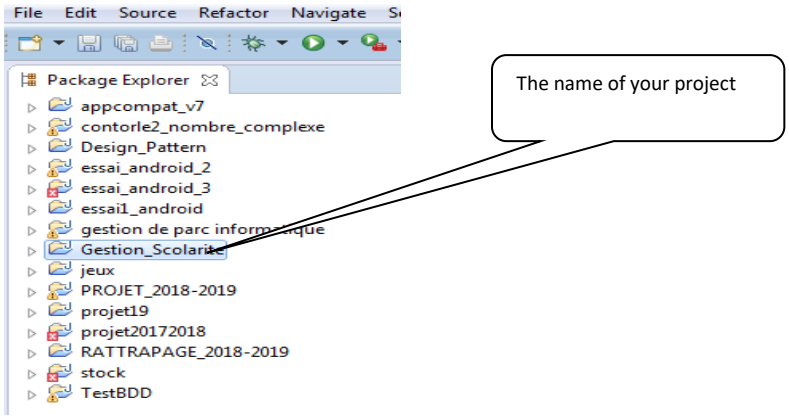
1. To create your project:



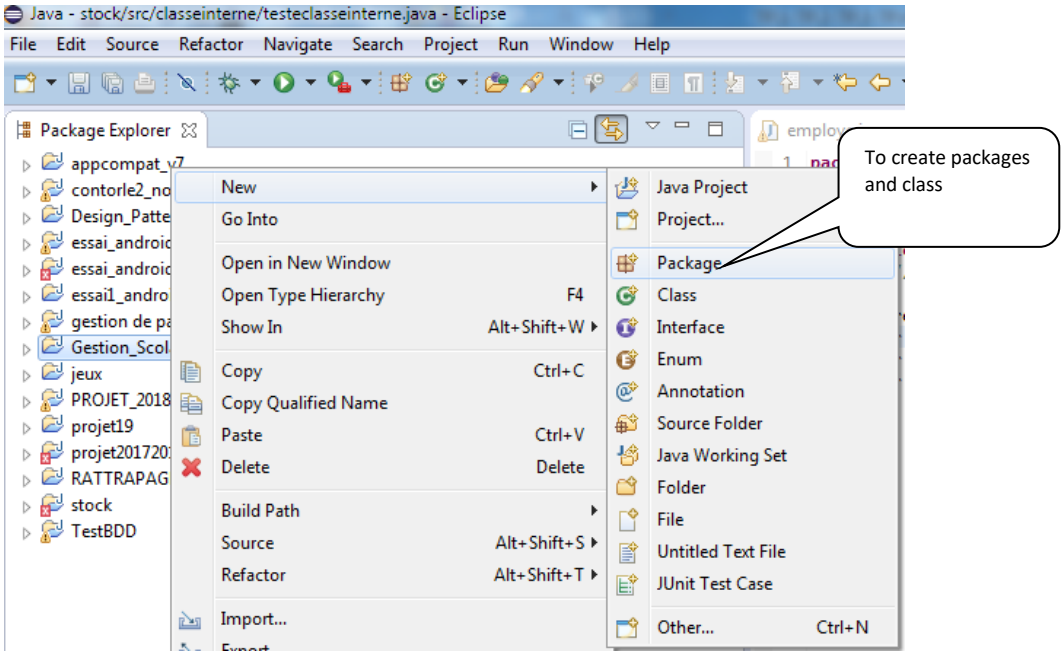
2. Enter the Project Name:



3. Your project is displayed in the Package Explorer pane:



4. By right-clicking on your project, you will have the commands to create the packages and associated classes:



You can continue the work .....

LAB2
(Visibility Modifiers)

Statement:

1. Create a new project named "LAB2_encapsulation".
2. Create the following class:

```
public class Rectangle {
   private int width;
   protected int length;
   public int lon;
   int lar;

   public Rectangle(int x, int y) {
      this.width = x;
      this.length = y;
   }

   public void display() {
      System.out.println("Width: " + width + " Length: " + length + " lon=====" + lon + "
lar===== " + lar);
   }
}
```

3. Create the following test class:

```
public class Test_Rect {
   public static void main(String[] args) {
      Rectangle r = new Rectangle(45, 15);
      r.display();
   }
}
```

4. Execute the test class and explain the displayed results.
5. Add the following code to the test class:

```
r.lon = 50;
r.lar = 60;
r.display();
r.length = 12;
r.width = 0;
```

6. Execute the test class and explain the displayed results.
7. Correct the errors.
8. Execute the test class.
9. Write a paragraph explaining the concept of encapsulation in your project.

LAB3
(Visibility Levels)

Hint:

- Errors will be signaled by the **compiler** (a line containing an error will be indicated on the left by a small red rectangle with a cross inside).
- We will introduce the concept of inheritance using the reserved word **extends** (see next chapter) in this LAB to demonstrate visibility levels.
- During this LAB, you are asked at each step to identify and explain the errors.

Statement:

1. Create a project named "LAB3_encapsulation".
2. Create two packages, P1 and P2, with the associated programs according to the following schema:

```
package p1 ;
class C1 {
public  int a=10;
protected int b=20;
int c=30;
private int d=50;
static int m=5;}
class C2 extends C1 {}
class C3 {}
```

```
package p2 ;
class C4 extends C1 {}
class C5 {}
```

3. Add the following code in class C2:

```
int e=a+15;
int f=b+12;
int j=c+47;
int u=d+2;
```

4. Identify and explain the errors.
5. Add the following code in class C3:

```
C1 t =new C1();
int o=t.a;
int p=t.b;
int l=t.c;
int i=t.m;
int j=C1.m;
```

5. Are there any errors? Explain the situation.
6. Add the following code in class C4:

```
C1 c11=new C1();
int e=c11.a+1;
int f=b+1;
int h=c+12;
```

7. Explain the errors.
8. Add the following code in class C5:

```
C1 t =new C1();
int o=t.a+45;
int k=t.b+5;
int p=t.c+1;
```

9. Identify and explain the errors.
10. Fill the following table with **Y** when the variable is accessible and **N** otherwise:

	Variable (a)	Variable (b)	Variable (c)	Variable (d)	Variable (m)
Accessibility of C2					
Accessibility of C3					
Accessibility of C4					
Accessibility of C5					

11. Write a test class containing the main() method to display the content of all instance variables (a, b, c, d, ...) and the class variable (**static int m = 5**).

Answer: Follow the steps, and you will find the same results explained in Chapter 2: Encapsulation.

LAB4
(Accessors: Setters and Getters)

Hint:

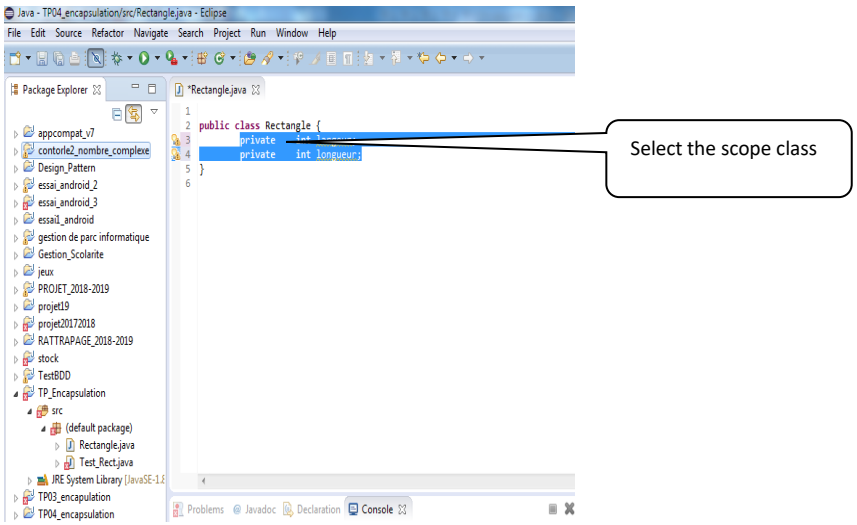
- The development of constructors, setters, and getters can be automatically generated by Eclipse, or they can be manually developed.

Statement:

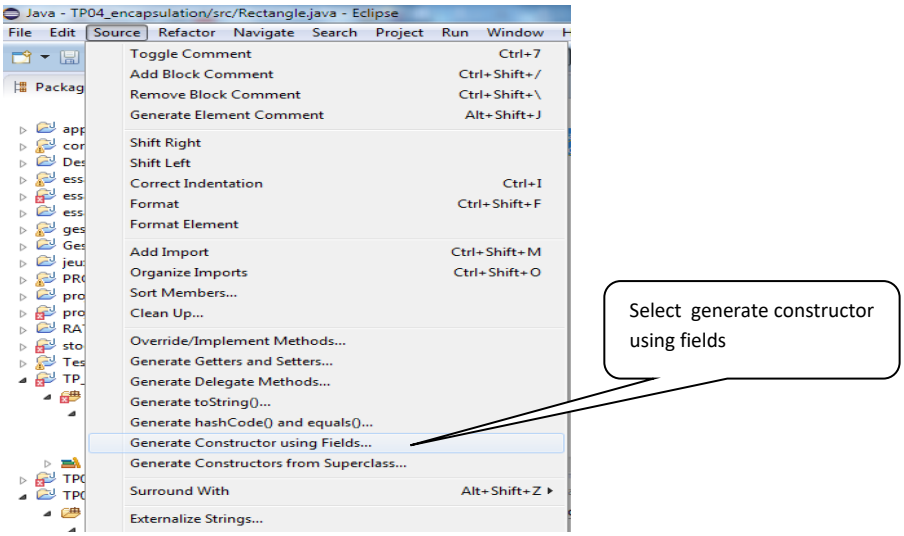
1. Create a project named "LAB4_encapsulation" with a package named "Set_Get".
2. Write the following Rectangle class:

```
public class Rectangle {
   private int width;
   private int length;
}
```

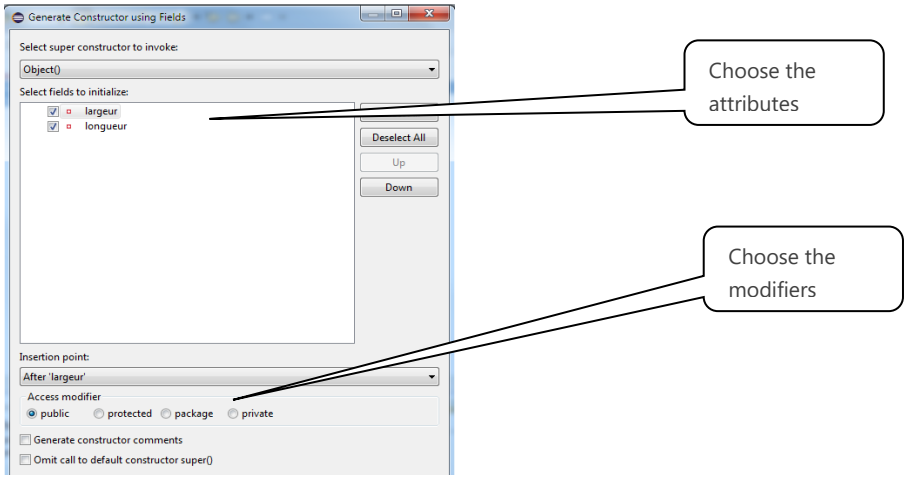
a. Automatic generation of constructors, setters, and getters.
a.1. Select the scope of the class as indicated in the following figure:



a.2. Go to the Source menu and validate the command "Generate Constructor using Fields", as indicated in the following figure:



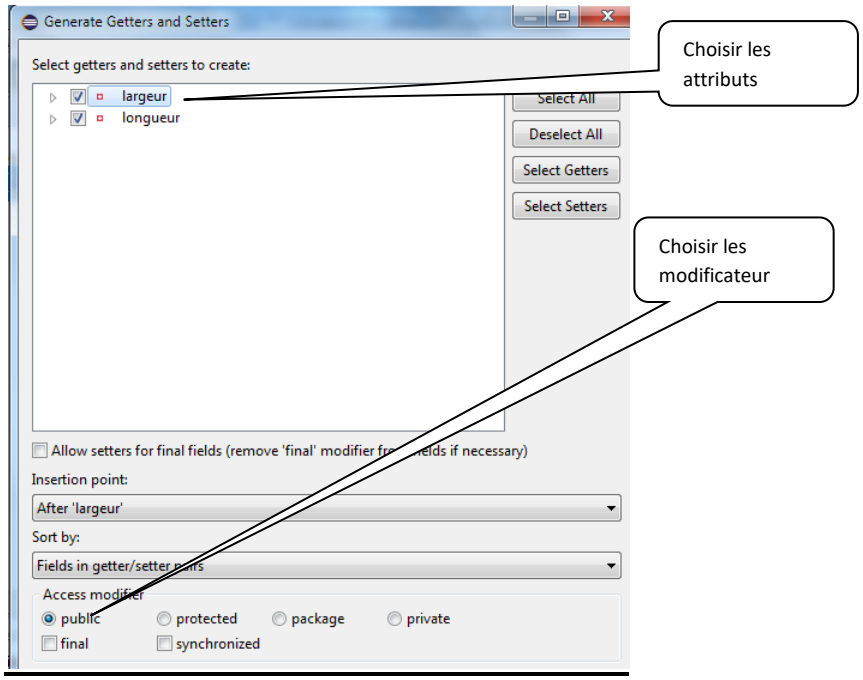
a.3. Choose the attributes and their access modifiers for your constructor, as indicated in the flowing figure



a.4. Validate the OK button (a parameterized constructor with the two attributes will be automatically generated).

Generating setters and getters]

a a.1. From step a.2 of generating the constructor, validate the command "Generate Getters and Setters", and you will have the following situation:



a .4. Validate the OK button: four methods will be generated (two setters and two getters for length and width).

Note:

- The automatic method remains limited as it only generates the skeleton of the methods. It is up to the programmer to complete the code if necessary.
- The Manual method: It is done by the programmer, who always has full control over the code.

Task to do: Create a test class to execute the Rectangle class.

LAB5
("this" Object)

Statement: A Circle is defined by a radius, and we can calculate its surface and perimeter. Using the **this** object, create a project named LAB5_encapsulation" that defines the Circle class with:

a. Two constructors, one parameterized by the radius and the other without parameters but calling the first constructor to create an object with a radius equal to 0.
b. Setters and getters.
c. Two methods to calculate the area and perimeter of a circle.
d. A display method to show the characteristics of a Circle object.
II.2 Create a test class named "Test_LAB5" containing the main method.
II.3 Instantiate the Circle class by creating several Circle objects.
II.4 Execute the project for several cases of radius.

Answer

Constructors

```
public class Circle {
   int radius;

   Circle(int radius) {
      this.radius = radius; // "this" is used as an object
   }
```

```
    Circle() {
        this(0); // "this" is used as a method    }}
```

Setters and Getters

```
public int getRadius() {    return radius;}

public void setRadius(int radius) {    this.radius = radius;}
```

The two methods: surface and perimeter

```
public int surface () {     // Complete the code...}

public void perimeter() {     // Complete the code...}
```

The display method

public void display() {   System.out.println("......Complete the code....");}

..... Complete and display the results on the console.

LAB 6
(Class Variables and Methods (static))

Statement:

In a store, an item is characterized by a code, price, and name. We want to count the number of items in the store (provide a static variable to increment it during each instantiation of the item class) and a static method to display the number of items stored in the store.

Task to do:

1. Create the Item class according to the LAB statement.
2. Test the Item class.

Answer

```
public class Item {
    private int code;
    private float price;
    private String name;
    private static int count;

    public Item (int code, String name, float price) {
        this.code = code;
        this.name = name;
        this.price = price;
        count++;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }
```

```
    public float getPrice() {
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void display() {
        System.out.println("Code: " + code + " Name: " + name + " Price: " + price);
    }

    public static void itemCount() {
        System.out.println(count);
    }
}
```

Test the class:

```
public class Test_LAB6_encapsulation {
    public static void main(String[] args) {
        Item r = new Item(45, "computer", 15);
        r.display();
        Item.articleCount(); // Call to the static method
        r.itemCount(); // Call to the static method
    }
}
```