OBJECT ORIENTED PROGRAMMING

Lab 6: OOP inheritance (1)

### Exercise 1:

What results does this program provide? Explain the results.

**class A**

**{**

**public A (int nn)**

**{**

**System.out.println ("Entree Constr A - n=" + n + " p=" + p) ;**

**n = nn ;**

**System.out.println ("Sortie Constr A - n=" + n + " p=" + p) ;**

**}**

**public int n ;**

**public int p=10 ;**

**}**

**class B extends A**

**{**

**public B (int n, int pp)**

**{**

**super (n) ;**

**System.out.println ("Entree Constr B - n=" + n + " p=" + p + "**

**q=" + q) ;**

**p = pp ;**

**q = 2\*n ;**

**System.out.println ("Sortie Constr B - n=" + n + " p=" + p + "q=" + q) ;**

**}**

**public int q=25 ;**

**}**

**public class TstInit**

**{**

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

**{**

**A a = new A(5) ;**

**B b = new B(5, 3) ;**

**}**

**}**

Answers

**It is necessary to take into account the order in which the initializations of the fields (explicit and implicit) and the calls to the constructors take place, namely:**

**• default initialization of the fields of the derived object (including those inherited),**

**• explicit initialization of the inherited fields,**

**• execution of the base class constructor,**

**• explicit initialization of fields specific to the derived object,**

**• execution of the constructor of the derived class.**

**This leads us to the following results:**

**Entree Constr A - n=0 p=10**

**Sortie Constr A - n=5 p=10**

**Entree Constr A - n=0 p=10**

**Sortie Constr A - n=5 p=10**

**Entree Constr B - n=5 p=10 q=25**

**Sortie Constr B - n=5 p=3 q=10**

### Exercise 2:

We have the following class:

**class Point**

**{**

**public void initialise (int x, int y)**

**{**

**this.x = x ; this.y = y ;**

**}**

**public void deplace (int dx, int dy)**

**{**

**x += dx ; y += dy ;**

**}**

**public int getX() { return x ; }**

**public int getY() { return y ; }**

**private int x, y ;**

**}**

Create a **PointA** class, derived from Point with a method **affiche** that display (in console window) the coordinates of a point. Write a small test program that uses the two classes **Point** and **PointA**.

What if the Point class did not have **getX** and **getY** methods?

Answers

**Just define a derived class using the extends keyword. The method affiche, like any method of a derived class, has access to all public members of the super class, so in particular getX and getY.**

**class PointA extends Point**

**{**

**void affiche()**

**{**

**System.out.println ("Coordonnees : " + getX() + " " + getY()) ;**

**}**

**}**

**We can then create objects of type PointA and apply to them the public methods of PointA and those of Point as in the following program:**

**public class TsPointA**

**{**

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

**{**

**Point p = new Point () ;**

**p.initialise (2, 5) ;**

**System.out.println ("Coordonnees : " + p.getX() + " " + p.getY()) ;**

**PointA pa = new PointA () ;**

**pa.initialise (1, 8) ; // methode form Point**

***Output illustration :***

Coordonnees : 2 5

Coordonnees : 2 5

**pa.affiche() ; // methode from PointA**

**}**

**}**

**Note that a call such as p.affiche() would lead to a compilation error since the class of p (Point) does not have a affiche method.**

**If the Point class did not have the getX and getY access methods, it would not have been possible to access its private x and y fields from the PointA class. The inheritance does not allow you to circumvent the principle of encapsulation.**

### Exercise 3:

We have the following class:

**class Point**

**{**

**public void setPoint (int x, int y)**

**{**

**this.x = x ; this.y = y ;**

**}**

**public void deplace (int dx, int dy)**

**{**

**x += dx ;**

**y += dy ;**

**}**

**public void affCoord ()**

**{**

**System.out.println ("Coordonnees : " + x + " " + y) ;**

**}**

**private int x, y ;**

**}**

Create a **PointNom** class, derived from **Point**, allowing to handle points defined by two coordinates (**int**) and a name (**character**). The following methods will be provided for:

* *setPointNom* to define the coordinates and name of a **PointNom** type object,
* *setName* to define only the name of such an object,
* *affCoordNom* to display the coordinates and name of a **PointNom** type object.

Write a small program using the **PointName** class.

Answers

**We define a derived class using the extends keyword:**

**class PointName extends Point**

**In this PointName class, we introduce a field (preferably private) intended to contain the name of the point:**

**private char name;**

**The setName method is trivial. Because of Point's data encapsulation, our other two methods absolutely must use Point's public methods.**

**Ultimately, here is the definition of our PoitnName class:**

**class PointName extends Point**

**{**

**public void setPointName (int x, int y, char name)**

**{**

**setPoint (x, y);**

**this.name = name;**

**}**

**public void setName (char name)**

**{**

**this.name = name;**

**}**

**public void affCoordName ()**

**{**

**System.out.print ("Name point" + name + "");**

**affCoord ();**

**}**

**private char name;**

**}**

**//Here is a program for using PointNom:**

**public class TsPointN**

**{**

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

**{**

**Point p = new Point ();**

**p.setPoint (2, 5);**

**p.affCoord ();**

**PointName pn1 = new PointName ();**

**pn1.setPointName (1, 7, 'A'); // method of PointName**

**pn1.affCoordName (); // method of PointName**

**pn1.deplace (9, 3); // Point method**

**pn1.affCoordName (); // method of PointName**

***Output illustration :***

Coordonnees : 2 5

Point de nom A Coordonnees : 1 7

Point de nom A Coordonnees : 10 10

Point de nom B Coordonnees : 4 3

Coordonnees : 4 3

**PointName pn2 = new PointName ();**

**pn2.setPoint (4, 3); // Point method**

**pn2.setName ('B'); // method of PointName**

**pn2.affCoordName (); // method of PointName**

**pn2.affCoord (); // Point method**

**}**

**}**

**Note:**

**1. Here again, as our classes do not have a constructor, it is possible to create objects without initializing them. In this case, their fields will simply have a "null" value, that is, here the integer value 0 for the coordinates and the null code character for the name.**

**2. While the Point class does not have coordinate access methods, we see that the affCoordNom method has no other possibility than to resort to the affCoord method of Point, which imposes constraints on the presentation of the coordinates. results. In particular, it would be impossible to display on the same line the name of the point before the coordinates.**

### Exercise 4:

We have the following class (this time with a manufacturer):

**class Point**

**{**

**public Point (int x, int y) {**

**this.x = x ; this.y = y ;**

**}**

**public void affCoord(){**

**System.out.println ("Coordonnees : " + x + " " + y) ;**

**}**

**private int x, y ;**

**}**

Create a **PointName** class, derived from **Point**, allowing you to handle points defined by their coordinates (**integers**) and a name (**character**). The following methods will be provided for:

* *constructor* to define the coordinates and the name of a **PointNom** type object,
* *affCoordNom* to display the coordinates and name of a **PointNom** type object.

Write a small program using the **PointName** class..

Answers

**This exercise is similar to Exercise 61, but this time both classes have a constructor. That of the derived class PointName must support the construction of the entire corresponding object, even if it means relying on the constructor of the base class (which is essential here since the class**

**Point does not have access methods). Remember that the call to the constructor of the base class (made using the super keyword) must constitute the first instruction of the constructor of the derived class.**

**Ultimately, here is what the definition of our PointName class could be:**

**class PointName extends Point**

**{**

**public PointName (int x, int y, char name)**

**{**

**super (x, y);**

**this.name = name;**

**}**

**public void affCoordName ()**

**{**

**System.out.print ("Name point" + name + "");**

**affCoord ();**

**}**

**private char name;**

**}**

**//Here is a little program for using PointNom:**

**public class TsPointC**

**{**

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

**{**

**PointName pn1 = new PointName (1, 7, 'A');**

**pn1.affCoordName (); // method of PointName**

**PointName pn2 = new PointName (4, 3, 'B');**

**pn2.affCoordName (); // method of PointName**

**pn2.affCoord (); // Point method**

**}**

**}**

***Output illustration :***

Point de nom A Coordonnees : 1 7

Point de nom B Coordonnees : 4 3

Coordonnees : 4 3

### Exercise 5:

We have the following class:

**class Point**

**{**

**public Point (int x, int y) {**

**this.x = x ; this.y = y ;**

**}**

**public void affiche(){**

**System.out.println ("Coordonnees : " + x + " " + y) ;**

**}**

**private int x, y ;**

**}**

Create a **PointNom** class, derived from **Point**, allowing you to handle points defined by their coordinates and a name (**character**). The following methods will be provided for:

* *constructor* to define the coordinates and the name of a **PointNom** type object,
* *affiche* to display the coordinates and the name of a **PointNom** type object.

Answers

**This exercise is similar to the previous exercise. The constructor's writing remains the same. But, this time, we have to redefine the affiche method in the derived class. Displaying the name is no problem:**

**System.out.print ("Name point" + name + "");**

**However, we need to call the affiche method of the base class. To do this, we use the keyword super:**

**super.affiche ();**

**Ultimately, here is the definition of our class, along with a small usage program:**

**class PointName extends Point**

**{**

**public PointName(int x, int y, char Name)**

**{**

**super (x, y);**

**this.Name = Name;**

**}**

**public void affiche()**

**{**

**System.out.print ("Name point" + Name + "");**

**super.display ();**

**}**

**private char Name;**

**}**

**public class TsPointR**

**{**

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

**{**

**Point p = new Point (3, 7);**

**p.affiche(); // Point method**

**PointName pn = new PointName (1, 7, 'A');**

**pn.affiche(); // method of PointName**

**}**

**}**