Exercice 1:

Soit la classe Java Suivante :

public class Pixel {

int x ;

int y ;

}

1/ Proposez un constructeur paramétré à la classe précédente

2/ Encapsulez les attributs x et y de cette classe

3/ A partir de la classe précédente, on veut construire la classe « ColoriedPixel » des pixels coloriés. Proposez une implémentation de cette classe avec son constructeur.

N.B : La couleur est un tableau de 3 valeurs entières (dans le plan RGB)

4/ Proposez une encapsulation adéquate à l'attribut couleur.

Exercice 2 :

- Donnez le code Java d'une classe nommé « Segment » contenant deux objets de la classe ColoriedPixel comme attributs (encapsulés).
- On veut construire maintenant la classe des rectangles de deux manières différentes :

o La première par héritage de la classe Segment

o La deuxième par composition (la classe comporte deux attributs objets de la classe Segment).

- Donnez le code des deux variantes.
- Quelle est la meilleure implémentation à votre avis.

Solution :

```
public class Pixel {
   private int x;
   private int y;

   // Parametrized Constructor
   public Pixel(int x, int y) {
      this.x = x;
      this.y = y;
   }
```

```
    // Getters and Setters for x and y
    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }
}

class ColoredPixel extends Pixel {
    private int[] color; // RGB color array

    // Constructor for ColoredPixel
    public ColoredPixel(int x, int y, int[] color) {
        super(x, y);
        this.color = color;
    }

    // Getter and Setter for color
    public int[] getColor() {
        return color;
```

```
    }

    public void setColor(int[] color) {
        this.color = color;
    }
}


public class Segment {
    private ColoredPixel pixel1;
    private ColoredPixel pixel2;

    // Constructor for Segment
    public Segment(ColoredPixel pixel1, ColoredPixel pixel2) {
        this.pixel1 = pixel1;
        this.pixel2 = pixel2;
    }

    // Getters and Setters for pixel1 and pixel2
    public ColoredPixel getPixel1() {
        return pixel1;
    }

    public void setPixel1(ColoredPixel pixel1) {
        this.pixel1 = pixel1;
    }

    public ColoredPixel getPixel2() {
        return pixel2;
    }
```

```
    public void setPixel2( coloredPixel pixel2) {
        this.pixel2 = pixel2;
    }
}
```

1ère Variante (Héritage)

```
 class Rectangle extends Segment {
/ / pixel1 e t pixel2 sont hérités de la classe segments
    private ColoredPixel pixel3;
    private ColoredPixel pixel4;
    public Rectangle(ColoredPixel pixel1, ColoredPixel pixel2, ColoredPixel pixel3, ColoredPixel pixel4) {
        super(pixel1, pixel2);
this.pixel3= pixel3 ;
this.pixel4= pixel4 ;

    }
/ / à compléter par les accesseurs « getters » de pixel3 e t pixel4 de l a même maniéré que dans Segment pour pixel1  et pixel2
}
```

2eme variante

```
class Rectangle1 {
    private Segment segment1;
    private Segment segment2;

    // Constructor using composition
    public Rectangle1(Segment segment1, Segment segment2) {
        this.segment1 = segment1;
        this.segment2 = segment2;
    }

    // Getters and Setters for segment1 and segment2
```

```
    public Segment getSegment1() {
        return segment1;
    }

    public void setSegment1(Segment segment1) {
        this.segment1 = segment1;
    }

    public Segment getSegment2() {
        return segment2;
    }

    public void setSegment2(Segment segment2) {
        this.segment2 = segment2;
    }
}
```