IUT de Paris Descartes – Base de la Programmation Objet

Travaux Dirigés – Sujet n°7

Objectifs

— Révisions

Travail à faire

1. Soit la définition suivante

```
public class Coordonnée {
  private int x, y;

  public Coordonnée(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public String toString() {
    return "(" + x +",␣" + y + ")";
  }
}
```

Écrivez un court programme construisant une coordonnée de valeur $(1, 3)$ et l'affichant sur la sortie standard.

Solution:

```
public class Appli {
  public static void main(String[] args) {
    Coordonnée c = new Coordonnée(1, 3);
    System.out.println(c);
  }
}
```

2. En vous basant sur le constructeur existant, définissez un nouveau constructeur initialisant systématiquement les attributs `x` et `y` à la valeur 0.

Solution:

```
public class Coordonnée {
  ...
  public Coordonnée() {
    this(0, 0);
```

```
  }
  ...
}
```

3. On nous précise que les coordonnées doivent prendre nécessairement des valeurs dans les entiers naturels (i.e. supérieurs ou égaux à 0). Levez une exception à la construction lorsque cette contrainte n'est pas satisfaite. Vous ferez en sorte que le programme de la question 1 n'ait pas à être modifié.

Solution: Fournir des entiers négatifs est considéré comme étant une erreur de programmation. En conséquence, il faut lever une exception de type `RuntimeException` (qui n'implique pas un bloc `try/catch` pour l'appelant. Ici nous levons une exception de type `IllegalArgumentException` qui est de la même "famille" que `RuntimeException`.

```
public class Coordonnée {
  ...
  public Coordonnée(int x, int y) throws IllegalArgumentException {
    if (x < 0 || y < 0)
        throw new IllegalArgumentException("Entiers naturels attendus");
    this.x = x;
    this.y = y;
  }
  ...
}
```

4. Ajoutez à la classe `Coordonnée` une méthode retournant la distance entre deux coordonnées.
Soient deux coordonnées $A = (x_A, y_A)$ et $B = (x_B, y_B)$, nous rappelons que la distance entre A et B est égale à $\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$. La méthode de classe `Math.sqrt(double d)` retourne la racine carré de `d` et la méthode de classe `Math.pow(double v, double p)` retourne la valeur de `v` montée à la puissance `p`.

Solution:

```
public class Coordonnée {
  ...
  public double distance(Coordonnée c) {
    return Math.sqrt(Math.pow(c.x - x, 2) + Math.pow(c.y - y, 2));
  }
  ...
}
```

5. Complétez le programme de la question 1 de manière à afficher la distance séparant l'origine (0, 0) de la coordonnée déjà déclarée.

Solution:

```
public class Appli {
  public static void main(String[] args) {
    Coordonnée c = new Coordonnée(1, 3);
    System.out.println(c);
    System.out.println(c.distance(new Coordonnée()));
  }
}
```

6. Développez une classe représentant un segment de droite étant caractérisé par les coordonnées de ses deux extrémités. Cette classe doit nécessairement proposer un constructeur, une méthode indiquant la taille du segment, une méthode `toString` et une méthode permettant de décider si deux segments sont égaux. Vous ferez attention que si A et B sont deux coordonnées alors (A, B) ainsi que (B, A) désignent le même segment (i.e. ces deux segments sont égaux). Vous pouvez compléter la classe `Coordonnée` si cela s'avère nécessaire.

Solution: Pour limiter les ajouts de méthodes à la classe `Coordonnée`, on introduit uniquement une relation d'ordre entre les coordonnées. On peut noter que pour deux coordonnées `a` et `b`, si

`!a.estPlusPetiteQue(b) && !b.estPlusPetiteQue(a)`

alors `a` est égale à `b`.

```
public class Coordonnée {
  ...
  public boolean estPlusPetiteQue(Coordonnée c) {
    return x < c.x || (x == c.x && y < c.y);
  }
}
```

La méthode ci-dessus est suffisante pour développer la classe `Segment`. D'autres solutions sont évidemment possibles.

```
public class Segment {
  private Coordonnée début, fin;

  public Segment(Coordonnée a, Coordonnée b) {
    // On ordonne les deux extrémités pour faciliter les
    // comparaisons entre segments
    if (a.estPlusPetiteQue(b)) {
      début = a;
      fin = b;
    }
    else {
      début = b;
      fin = a;
    }
  }

  public double taille() {
    return début.distance(fin);
  }

  public boolean egal(Segment s) {
    return !début.estPlusPetiteQue(s.début) &&
        !s.début.estPlusPetiteQue(début) &&
        !fin.estPlusPetiteQue(s.fin) &&
        !s.fin.estPlusPetiteQue(fin);
  }

  public String toString() {
    return début + " -- " + fin;
  }
}
```

7. Dessinez le diagramme de classe regroupant la classe `Coordonnée`, celle développée à la question précédente et la classe contenant le programme principal. Vous devez répartir ces classes au sein de paquetages.

Solution: