Exercice 1

Ecrivez une classe Point avec les attributs suivants :

- **x** : L'abscisse du point;
- **y** : L'ordonnée du point.

La classe Point doit disposer des constructeurs suivants :

- **Point**();
- **Point**(x, y).

La classe Point doit contenir les accesseurs (**get**) et mutateurs (**set**) et aussi une méthode **toString**() donnant une représentation du point.

Ecrivez une classe Rectangle héritant de Point avec les attributs suivants :

- **longueur** : La longueur du rectangle;
- **largeur** : La largeur du rectangle.

La classe Rectangle doit disposer des constructeurs suivants :

- **Rectangle**();
- **Rectangle**(x, y, longueur, largeur).

La classe Rectangle doit contenir des accesseurs (**get**) et mutateurs (**set**) et aussi les méthodes suivantes :

- **aire**() : Donne l'aire du rectangle (La surface = longueur * largeur);
- **toString**() : Donne une représentation du rectangle (surcharge).

Ecrivez une classe Parallelepipede héritant de Rectangle avec les attributs suivants :

- **hauteur** : La hauteur du parallélépipède.

La classe Parallelepipede doit disposer des constructeurs suivants :

- **Parallelepipede**();
- **Parallelepipede**(x, y, longueur, largeur, hauteur).

La classe Parallelepipede doit contenir des accesseurs (**get**) et mutateurs (**set**) et aussi les les méthodes suivantes :

- **aire**() : Donne l'aire du parallélépipède (surcharge);
- **volume**() : Donne le volume du parallélépipède;
- **toString**() : Donne une représentation du parallélépipède (surcharge).

Ecrivez aussi une classe TestParallelepipede afin de tester les classes.

Solution

```java

/*
 * Fichier: Point.java
 * Crée le: 29 janvier 2007.
 * Modifié: 7 juillet 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
  * the Free Software Foundation; either version 2 of the License, or
  * (at your option) any later version.
  *
  * This program is distributed in the hope that it will be useful,
  * but WITHOUT ANY WARRANTY; without even the implied warranty of
  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  * GNU General Public License for more details.
  *
  * You should have received a copy of the GNU General Public License
  * along with this program; if not, write to the Free Software
  * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
  */

package chapitre8.parallelepipede;



/**
 * <p>Classe représentant un point.</p>
 * @author Sébastien ESTIENNE.
 */
public class Point
{
   /** L'abscisse du point. */
   protected double x;

   /** L'ordonnée du point. */
   protected double y;


   /**
    * <p>Constructeur par défaut.</p>
    */
   public Point()
   {
      this(0.0, 0.0);
   }


   /**
    * <p>Constructeur de Point avec l'abscisse et l'ordonnée spécifiés.</p>
    * @param x L'abscisse du point.
    * @param y L'ordonnée du point.
    */
   public Point(double x, double y)
   {
      setX(x);
      setY(y);
   }
```

```
59.
60.
61.     /**
62.      * <p>Retourne l'abscisse du point.</p>
63.      * @return Renvoie l'abscisse du point.
64.      */
65.     public double getX()
66.     {
67.        return this.x;
68.     }
69.
70.
71.     /**
72.      * <p>Modifie l'abscisse du point.</p>
73.      * @param x L'abscisse du point.
74.      */
75.     public void setX(double x)
76.     {
77.        this.x = x;
78.     }
79.
80.
81.     /**
82.      * <p>Retourne l'ordonnée du point.</p>
83.      * @return Renvoie l'ordonnée du point.
84.      */
85.     public double getY()
86.     {
87.        return this.y;
88.     }
89.
90.
91.     /**
92.      * <p>Modifie l'ordonnée du point.</p>
93.      * @param y L'ordonnée du point.
94.      */
95.     public void setY(double y)
96.     {
97.        this.y = y;
98.     }
99.
100.
101.    /**
102.     * <p>Retourne une représentation du point.</p>
103.     * @return Renvoie une représentation du point.
104.     */
105.    @Override
106.    public String toString()
107.    {
108.       return new StringBuilder("Point [ ").append(getX()).append(";
    ").append(getY()).append(" ]")
109.             .toString();
110.    }
111. }
```

```
1.
2. /*
3.  * Fichier: Rectangle.java
4.  * Crée le: 29 janvier 2007.
5.  * Modifié: 7 juillet 2007.
```

```
6.  * Auteurs: Sébastien ESTIENNE.
7.  * SiteWeb: http://www.prog-info.org/
8.  *
9.  * This program is free software; you can redistribute it and/or modify
10.   * it under the terms of the GNU General Public License as published by
11.   * the Free Software Foundation; either version 2 of the License, or
12.   * (at your option) any later version.
13.   *
14.   * This program is distributed in the hope that it will be useful,
15.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
16.   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17.   * GNU General Public License for more details.
18.   *
19.   * You should have received a copy of the GNU General Public License
20.   * along with this program; if not, write to the Free Software
21.   * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
22.   */
23.
24.  package chapitre8.parallelepipede;
25.
26.
27.
28.  /**
29.   * <p>Classe représentant un rectangle.</p>
30.   * @author Sébastien ESTIENNE.
31.   */
32.  public class Rectangle extends Point
33.  {
34.     /** La longueur du rectangle. */
35.     protected double longueur;
36.
37.     /** La largeur du rectangle. */
38.     protected double largeur;
39.
40.
41.     /**
42.      * <p>Constructeur par défaut.</p>
43.      */
44.     public Rectangle()
45.     {
46.        this(0.0, 0.0, 0.0, 0.0);
47.     }
48.
49.
50.     /**
51.      * <p>Constructeur de rectangle avec l'abscisse, l'ordonnée, la longueur et la
    largeur
52.      * spécifiés.</p>
53.      * @param x L'abscisse de la position du rectangle.
54.      * @param y L'ordonnée de la position du rectangle.
55.      * @param longueur La longueur du rectangle.
56.      * @param largeur La largeur du rectangle.
57.      */
58.     public Rectangle(double x, double y, double longueur, double largeur)
59.     {
60.        super(x, y);
61.        setLongueur(longueur);
62.        setLargeur(largeur);
63.     }
64.
65.
```

```
66.     /**
67.      * <p>Retourne la largeur du rectangle.</p>
68.      * @return Renvoie la largeur du rectangle.
69.      */
70.     public double getLargeur()
71.     {
72.        return this.largeur;
73.     }
74.
75.
76.     /**
77.      * <p>Modifie la largeur du rectangle.</p>
78.      * @param largeur La largeur du rectangle.
79.      */
80.     public void setLargeur(double largeur)
81.     {
82.        if(largeur < 0.0)
83.        {
84.           this.largeur = 0.0;
85.        }
86.        else
87.        {
88.           this.largeur = largeur;
89.        }
90.     }
91.
92.
93.     /**
94.      * <p>Retourne la longueur du rectangle.</p>
95.      * @return Renvoie la longueur du rectangle.
96.      */
97.     public double getLongueur()
98.     {
99.        return this.longueur;
100.    }
101.
102.
103.    /**
104.     * <p>Modifie la longueur du rectangle.</p>
105.     * @param longueur La longueur du rectangle.
106.     */
107.    public void setLongueur(double longueur)
108.    {
109.       if(longueur < 0.0)
110.       {
111.          this.longueur = 0.0;
112.       }
113.       else
114.       {
115.          this.longueur = longueur;
116.       }
117.    }
118.
119.
120.    /**
121.     * <p>Calcule l'aire du rectangle.</p>
122.     * @return Retourne l'aire du rectangle.
123.     */
124.    public double aire()
125.    {
126.       return getLongueur() * getLargeur();
```

```
127.    }
128.
129.
130.    /**
131.     * <p>Retourne une représentation du rectangle.</p>
132.     * @return Renvoie une représentation du rectangle.
133.     */
134.    @Override
135.    public String toString()
136.    {
137.       return new StringBuilder("Rectangle [ position : (").append(getX()).append("; ").append(
138.             getY()).append("); longueur : ").append(getLongueur()).append("; largeur : ").append(
139.             getLargeur()).append(" ]").toString();
140.    }
141. }
142.
143. /*
144.  * Fichier: Parallelepipede.java
145.  * Crée le: 29 janvier 2007.
146.  * Modifié: 7 juillet 2007.
147.  * Auteurs: Sébastien ESTIENNE.
148.  * SiteWeb: http://www.prog-info.org/
149.  *
150.  * This program is free software; you can redistribute it and/or modify
151.  * it under the terms of the GNU General Public License as published by
152.  * the Free Software Foundation; either version 2 of the License, or
153.  * (at your option) any later version.
154.  *
155.  * This program is distributed in the hope that it will be useful,
156.  * but WITHOUT ANY WARRANTY; without even the implied warranty of
157.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
158.  * GNU General Public License for more details.
159.  *
160.  * You should have received a copy of the GNU General Public License
161.  * along with this program; if not, write to the Free Software
162.  * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
163.  */
164.
165. package chapitre8.parallelepipede;
166.
167.
168.
169. /**
170.  * <p>Classe représentant un parallélépipède rectangle.</p>
171.  * @author Sébastien ESTIENNE.
172.  */
173. public class Parallelepipede extends Rectangle
174. {
175.    /** La hauteur du parallélépipède rectangle. */
176.    protected double hauteur;
177.
178.
179.    /**
180.     * <p>Constructeur par défaut.</p>
181.     */
182.    public Parallelepipede()
183.    {
184.       this(0.0, 0.0, 0.0, 0.0, 0.0);
185.    }
```

```
186.
187.
188.    /**
189.     * <p>Constructeur de parallélépipède rectangle avec l'abscisse, l'ordonnée, la
   longueur, la
190.     * largeur et la hauteur spécifiés.</p>
191.     * @param x L'abscisse de la position du parallélépipède rectangle.
192.     * @param y L'ordonnée de la position du parallélépipède rectangle.
193.     * @param longueur La longueur du parallélépipède rectangle.
194.     * @param largeur La largeur du parallélépipède rectangle.
195.     * @param hauteur La hauteur du parallélépipède rectangle.
196.     */
197.    public Parallelepipede(double x, double y, double longueur, double largeur,
   double hauteur)
198.    {
199.       super(x, y, longueur, largeur);
200.       setHauteur(hauteur);
201.    }
202.
203.
204.    /**
205.     * <p>Retourne la hauteur du parallélépipède rectangle.</p>
206.     * @return Renvoie la hauteur du parallélépipède rectangle.
207.     */
208.    public double getHauteur()
209.    {
210.       return this.hauteur;
211.    }
212.
213.
214.    /**
215.     * <p>Modifie la hauteur du parallélépipède rectangle.</p>
216.     * @param hauteur La hauteur du parallélépipède rectangle.
217.     */
218.    public void setHauteur(double hauteur)
219.    {
220.       if(hauteur < 0.0)
221.       {
222.          this.hauteur = 0.0;
223.       }
224.       else
225.       {
226.          this.hauteur = hauteur;
227.       }
228.    }
229.
230.
231.    /**
232.     * <p>Calcule l'aire du parallélépipède rectangle.</p>
233.     * @return Retourne l'aire du parallélépipède rectangle.
234.     */
235.    @Override
236.    public double aire()
237.    {
238.       return 2 * getLongueur() * getLargeur() + 2 * getLongueur() * getHauteur() + 2
   * getLargeur()
239.             * getHauteur();
240.    }
241.
242.
243.    /**
```

```
244.     * <p>Calcule le volume du parallélépipède rectangle.</p>
245.     * @return Retourne le volume du parallélépipède rectangle.
246.     */
247.    public double volume()
248.    {
249.       return getLongueur() * getLargeur() * getHauteur();
250.    }
251.
252.
253.    /**
254.     * <p>Retourne une représentation du parallélépipède rectangle.</p>
255.     * @return Renvoie une représentation du parallélépipède rectangle.
256.     */
257.    @Override
258.    public String toString()
259.    {
260.       return new StringBuilder("Parallélépipède rectangle [ position :
   (").append(getX()).append(
261.             "; ").append(getY()).append("); longueur :
   ").append(getLongueur()).append(
262.             "; largeur : ").append(getLargeur()).append("; hauteur :
   ").append(getHauteur())
263.             .append(" ]").toString();
264.    }
265. }
266.
267. /*
268.  * Fichier: TestParallelepipede.java
269.  * Crée le: 29 janvier 2007.
270.  * Modifié: 7 juillet 2007.
271.  * Auteurs: Sébastien ESTIENNE.
272.  * SiteWeb: http://www.prog-info.org/
273.  *
274.  * This program is free software; you can redistribute it and/or modify
275.  * it under the terms of the GNU General Public License as published by
276.  * the Free Software Foundation; either version 2 of the License, or
277.  * (at your option) any later version.
278.  *
279.  * This program is distributed in the hope that it will be useful,
280.  * but WITHOUT ANY WARRANTY; without even the implied warranty of
281.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
282.  * GNU General Public License for more details.
283.  *
284.  * You should have received a copy of the GNU General Public License
285.  * along with this program; if not, write to the Free Software
286.  * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
287.  */
288.
289. package chapitre8.parallelepipede;
290.
291.
292.
293. import java.awt.BorderLayout;
294.
295. import javax.swing.JFrame;
296. import javax.swing.JScrollPane;
297. import javax.swing.JTextArea;
298.
299.
300.
301. /**
```

```
 * <p>Classe de test pour les classes Point, Rectangle et Parallelepipede.</p>
 * @author Sébastien ESTIENNE.
 */
public class TestParallelepipede extends JFrame
{
   /**
    * <p>Serial version UID.</p>
    */
   private static final long serialVersionUID = 1L;


   /**
    * <p>Construction de l'application.</p>
    */
   public TestParallelepipede()
   {
      // Appel du constructeur de la classe JFrame.
      super("Parallélépipède");

      // Ajoute les composants au conteneur.
      JTextArea zoneSortie = new JTextArea();
      getContentPane().add(new JScrollPane(zoneSortie), BorderLayout.CENTER);

      // Texte de sortie.
      StringBuilder sortie = new StringBuilder();

      // Point.
      Point p = new Point(1.3, 4.8);
      sortie.append(p).append('\n');

      // Rectangle.
      Rectangle r = new Rectangle(2.5, 6.2, 5.0, 3.0);
      sortie.append(r).append('\n');
      sortie.append("Aire : ").append(r.aire()).append('\n');

      // Parallélépipède.
      Parallelepipede m = new Parallelepipede(1.7, 3.2, 7.0, 4.0, 2.0);
      sortie.append(m).append('\n');
      sortie.append("Aire : ").append(m.aire()).append('\n');
      sortie.append("Volume : ").append(m.volume()).append('\n');

      // Met à jour la zone de sortie.
      zoneSortie.setText(sortie.toString());

      // Modifie les propriétés de la fenêtre.
      setSize(600, 200);
      setLocation(100, 100);
      setVisible(true);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }


   /**
    * <p>Débute l'exécution du test.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      new TestParallelepipede();
   }
}
```