université de BORDEAUX

J1INAW12 – Approche Objet
Master MIAGE

Examen du 15 décembre 2015, durée 3h
Corrigé

Exercice - Itération

Écrire dans une classe `Iterations` une méthode de classe
`public static <T> Iterator<T> iterationAvecSauts(Iterator<T> it, int[] sauts)`.

```
public class Iterateurs {
    private static class IterateurAvecSauts<T> implements Iterator<T> {
        Iterator<T> it;
        int[] sauts;
        int indice = 0;

        private void ajuster() {
            if (indice >= sauts.length)
                return;
            for (int i = 0; i < sauts[indice] && it.hasNext(); i++)
                it.next();
            indice++;
        }

        IterateurAvecSauts(Iterator<T> it, int[] sauts) {
            this.it = it;
            this.sauts = sauts;
            ajuster();
        }

        public boolean hasNext() {
            return it.hasNext();
        }

        public T next() {
            T t = it.next();
            ajuster();
            return t;
        }

        public void remove() {
            throw new UnsupportedOperationException();
        }
    }

    public static <T> Iterator<T> iterateurAvecSauts(Iterator<T> it, int[] sauts) {
```

```
        return new IterateurAvecSauts<T>(it, sauts);
    }
}
```

Exercice - Graphe

On considère les interfaces `Graphe` et `Arete`.

Question 1 Modifier ces interfaces pour que le type des sommets soit générique dans l'interface `Graphe`.

```
public interface Arete<T> {
    public T sommet1();
    public T sommet2();
}

public interface Graphe<T> {
    /** Ajoute un sommet s au graphe */
    public void ajouterSommet(T s);

    /**
     * Ajoute une arete entre les sommets s1 et s2. Si au moins une des
     * extremites de l'arete e n'appartient pas au graphe, leve une
     * IllegalArgumentException. */
    public void ajouterArete(Arete<? extends T> e);

    /** Itere les sommets du graphe */
    public Iterator<T> sommets();

    /** Itere les aretes du graphe */
    public Iterator<Arete<? extends T>> aretes();

    /**
     * Vrai si s1 et s2 sont voisins. Si s1 ou s2 n'appartiennent pas au graphe,
     * leve une IllegalArgumentException. */
    public boolean sontVoisins(Object s1, Object s2);

    /**
     * Itere les voisins du sommet s. Si s n'appartient pas au graphe, leve une
     * IllegalArgumentException. */
    public Iterator<T> voisins(Object s);
}
```

Question 2 Modifier ces interfaces pour que le type des arêtes soit également générique dans l'interface `Graphe`.

`Arete` reste inchangée.

```
public interface Graphe<T, E extends Arete<? extends T>> {
    /** Ajoute un sommet s au graphe */
    public void ajouterSommet(T s);
```

```
    /**
     * Ajoute une arete entre les sommets s1 et s2. Si au moins une des
     * extremites de l'arete e n'appartient pas au graphe, leve une
     * IllegalArgumentException. */
    public void ajouterArete(E e);

    /** Itere les sommets du graphe */
    public Iterator<T> sommets();

    /** Itere les aretes du graphe */
    public Iterator<E> aretes();

    /**
     * Vrai si s1 et s2 sont voisins. Si s1 ou s2 n'appartiennent pas au graphe,
     * leve une IllegalArgumentException. */
    public boolean sontVoisins(Object s1, Object s2);

    /**
     * Itere les voisins du sommet s. Si s n'appartient pas au graphe, leve une
     * IllegalArgumentException. */
    public Iterator<T> voisins(Object s);
}
```

Problème - Éditeur de figures

On considère l'application composée des trois classes : `Editeur`, `Planche` et `Ellipse`.

Question 1 Décrire en quelques lignes (10 max) le comportement de cette application.

Au lancement de l'application apparaît une fenêtre blanche de 600 `x` *600 pixels.*

Lorsque l'on clique dans cette fenêtre, sont créées de petites croix noires symbolisant des points. Puis, lorsque l'on appuie sur la touche 'e', le comportement change : on doit cliquer sur les points existants et dès que l'on a cliqué sur deux points successivement,une ellipse est rajoutée définie par ces deux points.

L'application est quittée lorsque la fenêtre est fermée.

Question 2 Modifier le code de `Ellipse` pour que, si e est une ellipse inscrite dans le rectangle d'extrémités $(xmin, ymin), (xmax, ymax)$ l'instruction

```
System.out.println(e);
```

affiche sur la sortie standard
`Ellipse (`$xmin, ymin$`) (`$xmax, ymax$`)`

Pour éviter la duplication de code, il est souhaitable de factoriser le calcul des valeurs minimum et maximum. Si les points peuvent bouger, ce calcul doit être refait à chaque fois que l'on utilise ces coordonnées. Sinon, il peut être fait dans le constructeur.

```
public class Ellipse {
```

```
    private Point[] points = new Point[2];
    private int xmin, xmax, ymin, ymax;
    ...
    private void calculMinMax() {
        xmin = Math.min(points[0].x, points[1].x);
        ymin = Math.min(points[0].y, points[1].y);
        xmax = Math.max(points[0].x, points[1].x);
        ymax = Math.max(points[0].y, points[1].y);
    }

    public void dessine(Graphics g) {
        calculMinMax();
        g.drawOval(xmin, ymin, xmax - xmin, ymax - ymin);
    }

    public String toString() {
        calculMinMax();
        return "Ellipse (" + xmin + "," + ymin + ") (" + xmax + "," + ymax
            + ")";
    }
}
```

Question 3 Modifier la classe `Planche` pour que, lorsque l'on ajoute un point, si un autre point est déjà présent à la même position (à epsilon près), une exception `PointExistantException` soit levée.

```
public void ajouterPoint(Point p) throws PointExistantException {
    Point p2 = point(p);
    if (p2 != null) {
        throw new PointExistantException(p2);
    }
    points.add(p);
    repaint();
}
```

Question 4 Donner le code de `PointExistantException`.

```
public class PointExistantException extends Exception {
    Point point;

    public PointExistantException(Point p) {
        this.point = p;
    }

    public Point point() {
        return point;
    }

    public String getMessage() {
        return "Impossible de creer un point pres de ("+
            point.x + "," + point.y+ ")";
    }
}
```

Question 5 En tenant compte du code des questions précédentes, modifier le code de `Editeur` pour que, si l'on essaye de créer un point à une position où un autre point existe déjà, un message d'avertissement s'affiche sur la sortie standard avec comme contenu "Impossible de créer un point près de (x, y)" où x et y sont les coordonnées du point déjà existant. L'application continuera ensuite de fonctionner normalement.

```
public class Editeur extends JFrame {
    ...
    public Editeur(int hauteur, int largeur) {
        ...
        final MouseListener m1 = new MouseAdapter() {
            public void mouseClicked(MouseEvent arg0) {
                try {
                    planche.ajouterPoint(arg0.getPoint());
                    planche.repaint();
                } catch (PointExistantException e) {
                    System.out.println(e.getMessage());
                }
            }
        };
        ...
    }
    ...
}
```

On souhaite maintenant pouvoir ajouter des formes autres que des ellipses, par exemple des triangles, des rectangles, etc. On souhaite donc utiliser l'interface `Forme`.

Question 6 Remplacer la méthode `Planche.ajouterEllipse(Ellipse e)` par `Planche.ajouterForme(Forme f)` et modifier les codes des différentes classes en conséquence. Ne spécifier sur la copie que les modifications, sans récrire tout le code existant.

```
public class Planche extends JComponent {
    ...
    private List<Forme> formes = new ArrayList<Forme>();
    ...
    public void paintComponent(Graphics g) {
        ...
        for (Forme e : formes)
            e.dessine(g);
    }
    ...
    public void ajouterForme(Forme e) {
        formes.add(e);
        repaint();
    }
}

public class Ellipse implements Forme {
    ...
}
```

```
public class Editeur extends JFrame {

    public Editeur(int hauteur, int largeur) {
        ...
        final MouseListener m2 = new MouseAdapter() {
                List<Point> points = new ArrayList<Point>();

                public void mouseClicked(MouseEvent arg0) {
                    Point p = planche.point(arg0.getPoint());
                    if (p != null) {
                        points.add(p);
                        if (points.size() == 2) {
                            planche.ajouterForme(new Ellipse(points));
                            planche.repaint();
                            points.clear();
                        }
                    }
                }
            };
        ...
    }

    ...
    }
}
```

Question 7 Écrire un décorateur `FormeColoree` qui permette de rajouter une couleur à une forme. Utiliser ce décorateur pour que les ellipses créées dans l'éditeur soient de couleur rouge (`Color.RED`).

```
public class FormeColoree implements Forme {
    Forme delegue;
    Color color;

    public FormeColoree(Forme f, Color c) {
        this.delegue = f;
        this.color = c;
    }

    public void dessine(Graphics g) {
        Color c = g.getColor();
        g.setColor(color);
        delegue.dessine(g);
        g.setColor(c);
    }
}
```

Il suffit ensuite de remplacer dans `Editeur` l'instruction

```
planche.ajouterForme(new Ellipse(points));
```

par

```
 planche.ajouterForme(new FormeColoree(new Ellipse(points), Color.RED));
```

Question 8 Écrire deux nouvelles implémentations de `Forme` : `Triangle` et `Rectangle`. Pour dessiner un rectangle, on utilisera la méthode de `Graphics drawRect(int x, int y, int width, int height)`. Pour dessiner un triangle il faudra utiliser trois fois la méthode de `Graphics drawLine(int x0, int y0, int x1, int y1)`.

La solution proposée ici factorise le code entre `Ellipse` et `Rectangle` à l'aide d'une classe abstraite `Rectangulaire`.

```
public abstract class Rectangulaire {

    protected Point[] points = new Point[2];

    public Rectangulaire(List<Point> points) {
        if (points.size() != 2)
            throw new IllegalArgumentException();
        points.toArray(this.points);
    }

    public void dessine(Graphics g) {
        int xmin = Math.min(points[0].x, points[1].x);
        int ymin = Math.min(points[0].y, points[1].y);
        int xmax = Math.max(points[0].x, points[1].x);
        int ymax = Math.max(points[0].y, points[1].y);
        dessineForme(g, xmin, ymin, xmax - xmin, ymax - ymin);
    }

    protected abstract void dessineForme(Graphics g, int x, int y, int largeur,
                                         int hauteur);
}

public class Rectangle extends Rectangulaire implements Forme {
    public Rectangle(List<Point> points) {
        super(points);
    }

    protected void dessineForme(Graphics g, int x, int y, int largeur,
                                int hauteur) {
        g.drawRect(x, y, largeur, hauteur);
    }
}

public class Ellipse extends Rectangulaire implements Forme {
    public Ellipse(List<Point> points) {
        super(points);
    }

    protected void dessineForme(Graphics g, int x, int y, int largeur,
                                int hauteur) {
        g.drawOval(x, y, largeur, hauteur);
    }

}

public class Triangle implements Forme {
```

```
    Point[] points = new Point[3];

    public Triangle(List<Point> points) {
        if (points.size() != 3)
            throw new IllegalArgumentException();
        points.toArray(this.points);
    }

    public void dessine(Graphics g) {
        g.drawLine(points[0].x, points[0].y,points[1].x, points[1].y);
        g.drawLine(points[1].x, points[1].y,points[2].x, points[2].y);
        g.drawLine(points[2].x, points[2].y,points[0].x, points[0].y);
    }
}
```

Question 9 Pour passer de la création d'un type de forme à un autre, on souhaite utiliser les touches du clavier. Utiliser l'interface `FabriqueForme` pour modifier le code de `Editeur` afin de permettre les changements de types de formes.

On donnera également les codes des classes implémentant `FabriqueForme`.

```
public class Editeur extends JFrame {

    MouseListener mouseListener = null;

    public Editeur(int hauteur, int largeur) {
        ...
        mouseListener = m1;

        FabriqueForme[] fabriques = { new FabriqueEllipse(),
                                      new FabriqueTriangle() ,
                                      new FabriqueRectangle() };

        for (final FabriqueForme ff : fabriques) {
            final MouseListener m2 = new MouseAdapter() {
                    List<Point> points = new ArrayList<Point>();

                    public void mouseClicked(MouseEvent arg0) {
                        Point p = planche.point(arg0.getPoint());
                        if (p != null) {
                            points.add(p);
                            if (points.size() == ff.nombrePoints()) {
                                planche.ajouterForme(ff.creerForme(points));
                                planche.repaint();
                                points.clear();
                            }
                        }
                    }
                };

            addKeyListener(new KeyAdapter() {
                    public void keyTyped(KeyEvent arg0) {
                        if (arg0.getKeyChar() == ff.toucheClavier()) {
                            planche.removeMouseListener(mouseListener);
```

```
                            mouseListener = m2;
                            planche.addMouseListener(m2);
                        }
                    }
                });
        }
        ...
    }
    ...
    }
}

public class FabriqueEllipse implements FabriqueForme {

    public Forme creerForme(List<Point> points) {
        return new Ellipse(points);
    }
    public int nombrePoints() {
        return 2;
    }
    public char toucheClavier() {
        return 'e';
    }
}

public class FabriqueRectangle implements FabriqueForme {

    public Forme creerForme(List<Point> points) {
        return new Rectangle(points);
    }
    public int nombrePoints() {
        return 2;
    }
    public char toucheClavier() {
        return 'r';
    }
}

public class FabriqueTriangle implements FabriqueForme {

    public Forme creerForme(List<Point> points) {
        return new Triangle(points);
    }
    public int nombrePoints() {
        return 3;
    }
    public char toucheClavier() {
        return 't';
    }
}
```