université de BORDEAUX

J1INAW12 – Approche Objet
Master MIAGE

Examen du 15 décembre 2015, durée 3h
Aucun document autorisé

Justifier clairement et succinctement vos choix, et décrivez brièvement vos algorithmes lorsque vous le jugez nécessaire.

Toute réponse difficile à comprendre sera considérée comme fausse.

Vous pouvez rajouter du code (méthode, donnée ou classe) non demandé à chaque fois que vous le jugerez nécessaire. Écrivez entièrement ce code, en particulier dans le cas d'exceptions, de classes abstraites ou d'interfaces.

Vous pouvez toujours considérer une classe ou une méthode demandée dans une question précédente comme disponible, même si vous n'avez pas traité cette question. Il vous est également conseillé de lire le sujet entièrement avant de commencer votre travail.

Si vous souhaitez utiliser une classe (ou une méthode particulière) de l'API Java dont vous ne vous souvenez pas du nom, donnez-lui un nom explicite et précisez-le clairement sur votre copie.

Vous pouvez à chaque fois que vous le jugez compréhensible utiliser des abréviations à la place des noms complets de classe et méthode ou encore '...' pour remplacer du code non modifié.

Les instructions `import` et `package` ne seront pas précisées.

Exercice - Itération

L'interface `java.util.Iterator` est rappelée ci-dessous :

```
public interface Iterator<T> {
    public boolean hasNext();
    public T next() throws NoSuchElementException;
    public void remove() throws UnsupportedOperationException;
}
```

Écrire dans une classe `Iterations` une méthode de classe
`public static <T> Iterator<T> iterationAvecSauts(Iterator<T> it, int[] sauts)`.
L'itérateur ainsi obtenu renverra les éléments retournés par l'itérateur `it` passé en paramètre en effectuant des sauts : entre le i^{eme} et le $i+1^{eme}$ appel à `next()`, `sauts[i]` éléments de `it` seront sautés. Si `sauts` compte moins de i éléments, aucun élément ne sera sauté entre le i et le $i+1^{eme}$ appel à `next()`.

Exemple : si `it` renvoie successivement 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, et `sauts` vaut `{1, 4}`, l'itérateur résultat de `iterationAvecSauts(it, sauts)` renverra successivement les valeurs 2, 7, 8, 9, 10.

La méthode `remove()` lèvera une exception `UnsupportedOperationException`.

Exercice - Graphe

On considère les interfaces `Graphe` et `Arete` ci-dessous :

```
public interface Graphe {
    /** Ajoute un sommet s au graphe */
    public void ajouterSommet(Object 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 e);

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

    /** Itere les aretes du graphe */
    public Iterator<Arete> 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<Object> voisins(Object s);
}

public interface Arete {
    public Object sommet1();
    public Object sommet2();
}
```

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

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

Problème - Éditeur de figures

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

```
public class Editeur extends JFrame {

    public Editeur(int hauteur, int largeur) {
        super("Editeur");
        JFrame.setDefaultLookAndFeelDecorated(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        final Planche planche = new Planche();
        add(planche, BorderLayout.CENTER);

        final MouseListener m1 = new MouseAdapter() {
                public void mouseClicked(MouseEvent arg0) {
                    planche.ajouterPoint(arg0.getPoint());
                    planche.repaint();
                }
            };
        planche.addMouseListener(m1);

        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.ajouterEllipse(new Ellipse(points));
                            planche.repaint();
                            points.clear();
                        }
                    }
                }
            };

        addKeyListener(new KeyAdapter() {
                public void keyTyped(KeyEvent arg0) {
                    if (arg0.getKeyChar() == 'e') {
                        planche.removeMouseListener(m1);
                        planche.addMouseListener(m2);
                    }
                }
            });

        setPreferredSize(new Dimension(hauteur, largeur));
        pack();
        setVisible(true);
    }

    public static void main(final String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
```

```
                public void run() {
                    new Editeur(600, 600);
                }
            });
    }
}

public class Planche extends JComponent {

    private static final int TAILLE_POINT = 2;
    private static final int EPSILON = 3;
    private List<Point> points = new ArrayList<Point>();
    private List<Ellipse> ellipses = new ArrayList<Ellipse>();

    public Planche() {
        setOpaque(true);
        setForeground(Color.black);
        setBackground(Color.white);
    }

    public void paintComponent(Graphics g) {
        if (isOpaque()) {
            g.setColor(getBackground());
            g.fillRect(0, 0, getWidth(), getHeight());
        }
        g.setColor(getForeground());
        for (Point p : points) {
            g.drawLine(p.x - TAILLE_POINT, p.y, p.x + TAILLE_POINT, p.y);
            g.drawLine(p.x, p.y - TAILLE_POINT, p.x, p.y + TAILLE_POINT);
        }
        for (Ellipse e : ellipses)
            e.dessine(g);
    }

    public void ajouterPoint(Point p) {
        points.add(p);
        repaint();
    }

    public Point point(int x, int y) {
        for (Point p : points)
            if (x >= p.x - EPSILON && x <= p.x + EPSILON && y >= p.y - EPSILON
                && y <= p.y + EPSILON)
                return p;
        return null;
    }

    public Point point(Point p) {
        return point(p.x, p.y);
    }

    public void ajouterEllipse(Ellipse e) {
        ellipses.add(e);
        repaint();
```

```
    }
}

public class Ellipse {
    private Point[] points = new Point[2];

    public Ellipse(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);
        g.drawOval(xmin, ymin, xmax - xmin, ymax - ymin);
    }
}
```

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

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 (`$xmin, ymin$`) (`$xmax, ymax$`)`

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.

Question 4 Donner le code de `PointExistantException`.

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.

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` suivante :

```
public interface Forme {
    public void dessine(Graphics g);
}
```

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.

Question 7 Écrire un décorateur `FormeColoree` qui permette de rajouter une couleur à une forme. La couleur souhaitée et la forme à colorer seront passées à l'instanciation de `FormeColoree`. Utiliser ce décorateur pour que les ellipses créées dans l'éditeur soient de couleur rouge (`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)`.

Question 9 Pour passer de la création d'un type de forme à un autre, on souhaite utiliser les touches du clavier, par exemple 'e' pour les ellipses, 't' pour les triangles, 'r' pour les rectangles ... Utiliser l'interface `FabriqueForme` ci-dessous pour modifier le code de `Editeur` afin de permettre les changements de types de formes.
`toucheClavier()` renverra la touche du clavier utilisée pour passer à ce type de formes (par exemple 'e' pour les ellipses), `nombrePoints()` le nombre de points nécessaires pour créer la forme (par exemple 2 pour une ellipse) et `creerForme(List<Point>)` retournera la nouvelle forme créée.
On donnera également les codes des classes implémentant `FabriqueForme`.

```
public interface FabriqueForme {
    public char toucheClavier();
    public int nombrePoints();
    public Forme creerForme(List<Point> points);
}
```