université de BORDEAUX

J1INAW12 – Approche Objet
Master Informatique
Pôle Universitaire Français

Final Exam, May 27th 2016
No document except a dictionary, length 3h

Any answer difficult to understand will be considered as false.

You may add any code (method, data, class) not requested each time you think it is necessary.

A class or a method requested may be considered available in the follow-up of the problem, even if you have not treated the question. It is recommended to read all the subject before starting your work.

If you want to use a class (or a given method) from the Java API, and you don't remember its name, give to it an explicit name and explain clearly the goal of the class on your copy.

You may use, each time you think it is comprehensible, an abbreviation instead the complete name of a class or a method, and also use '...' to replace an unmodified part of the code.

The instructions `import` *and* `package` *will be not given.*

Exercise - Iterator

We recall the interface `Iterator`

```
public interface Iterator<E> {
    /** returns true if the iteration has more elements. */
    public boolean hasNext();
    /* returns the next element of the iteration. Throws
       NoSuchElementException if the iteration has no more element. */
    public E next();
    /* Removes from the underlying collection the last element
       returned by the iterator. Throws UnsupportedOperationException
       if the remove  operation is not supported by this Iterator. */
    public void remove();
}
```

In a class `Iterators`, write a static method `public static <E> Iterator<E> subIterator(Iterator<? extends E> it, int fromIndex, int toIndex)` which returns an iterator that iterates elements from `it` with a rank between `fromIndex` and `toIndex`. Note that the rank of the first element is 0.

For example, the following instructions

```
for (Iterator<Integer> it =
        subIterator(Arrays.asList(0, 1, 2, 3, 4, 5, 6).iterator(), 2, 5);
     it.hasNext();)
    System.out.print(it.next() + " ");
```

will display

```
2 3 4 5
```

Problem - Shape editor

We consider the software composed by the three following classes : `Editor`, `DrawingBoard` and `Oval`.

```
public class Editor extends JFrame {

        public Editor(int height, int width) {
                super("Editor");
                JFrame.setDefaultLookAndFeelDecorated(true);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                final DrawingBoard board = new DrawingBoard();
                add(board, BorderLayout.CENTER);

                final MouseListener m1 = new MouseAdapter() {
                        public void mouseClicked(MouseEvent arg0) {
                                board.addPoint(arg0.getPoint());
                                board.repaint();
                        }
                };

                final MouseListener m2 = new MouseAdapter() {
                        List<Point> points = new ArrayList<Point>();

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

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

                board.addMouseListener(m1);

                setPreferredSize(new Dimension(height, width));
                pack();
                setVisible(true);
```

```
        }

        public static void main(final String[] args) {
                javax.swing.SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                                new Editor(600, 600);
                        }
                });
        }
}

public class DrawingBoard extends JComponent {

        private static final int POINT_SIZE = 2;
        private static final int EPSILON = 3;
        private List<Point> points = new ArrayList<Point>();
        private List<Oval> ovals = new ArrayList<Oval>();

        public DrawingBoard() {
                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 - POINT_SIZE, p.y, p.x + POINT_SIZE, p.y);
                        g.drawLine(p.x, p.y - POINT_SIZE, p.x, p.y + POINT_SIZE);
                }
                for (Oval e : ovals)
                        e.draw(g);
        }

        public void addPoint(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 addOval(Oval e) {
                ovals.add(e);
                repaint();
        }
}

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

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

        public void draw(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 Describe in few lines (10 max) how runs this software. More precisely, explain what's happen before pushing the key 'o', and after pushing the key 'o'.

Question 2 Modify the code of `Oval` in such a way that, if o is an instance of `Oval` inside a rectangle with coordinates $(xmin, ymin), (xmax, ymax)$, the instruction

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

will write on the standard output
`Oval (`$xmin, ymin$`) (`$xmax, ymax$`)`

Question 3 Modify the method `addPoint(Point p)` in the class `DrawingBoard` in such a way that, when we want to add a point `p`, if another point is yet present at the same position (with a precision of `EPSILON`), an exception `ExistingPointException` is thrown. *Indication : to detect if a point is yet existing near the position of* `p`*, use the method* `Point point(Point p)`.

Question 4 Give the code of the class `ExistingPointException`.

Question 5 Using the code of questions 3 and 4, modify the code of the class `Editor` in such a way that, if we try to create a point too closed to a yet existing point, a warning message will appear on the standard output with content : "Impossible to create a point near (x, y)" where x and y are the coordinates of the existing point. After writing the warning, the software will continue to run as usual.

Now, we want to be able to use other kind of shapes than oval, like triangles, rectangles ... For that, we will use the interface `Shape` :

```
public interface Shape {
        public void draw(Graphics g);
}
```

Question 6 Replace the method `DrawingBoard.addOval(Oval o)` by `DrawingBoard.addShape(Shape s)` and modify accordingly the code of the different classes to take this modification in account. *Indication : just specify the modifications on your copy, without rewriting all the code..*

Question 7 Write a decorator `ColoredShape` which allows to add a color to a shape. The color and the shape to color will be given as parameter to the constructor of the class `ColoredShape`. Use this decorator in such a way that the ovals created in the editor will be colored in red (`Color.RED`).

Question 8 Write two new implementations of `Shape` : `Triangle` and `Rectangle`. To draw a rectangle, use the method of `Graphics` `drawRect(int x, int y, int width, int height)`. To draw a triangle, use three times the method of `Graphics` `drawLine(int x0, int y0, int x1, int y1)`.

Question 9 To change the kind of shape to create, we want to use the keys of the keyboard, for example 'o' for the ovals, 't' for the triangles, 'r' for the rectangles ... Use the interface `ShapeFactory` to modify the code of `Editor` in such a way that it allows the change of the kind of shapes.
`key()` will return the key to push to select this kind of shape (for example 'o' for the ovals, 't' for triangles, 'r' for rectangles), `nPoints()` the number of points necessary to create this kind of shape (for example 2 for an oval or a rectangle, 3 for a triangle) and `createShape(List<Point>)` will return the new instance of shape created by the factory.
Give also the code of the classes which implement `ShapeFactory` for ovals, triangles and rectangles.

```
public interface ShapeFactory {
        public char key();

        public int nPoints();

        public Shape createShape(List<Point> points);
}
```