Practice Midterm

This exam is about the construction of a simple 2D graphics library. You are given the following class to start off things:

```
class BasicGLib {

    /** draw a circle of color c with center at current cursor position
     *  the radius of the circle is given by radius
     */
    public static void drawCircle(Color c, int radius) { /* ... */ }

    /** draw a rectangle of Color c
     *  with lower left corner at current cursor position
     *  the length of the rectangle along the x axis is given by xlength
     *  the length of the rectangle along the y axis is given by ylength
     */
    public static void drawRect(Color c, int xlength, int ylength) { /* ... */ }

    /** move the cursor by coordinate (xcoord,ycoord) { /* ... */ }
     */
    public static void moveCursor(int xcoord, int ycoord) { /* ... */ }

    /** clear the entire screen and set cursor position to (0,0)
     */
    public static void clear() { /* ... */ }
}
```

The initial cursor position is at (0,0).

For example:

```
BasicGLib.clear();                    // initialize
BasicGLib.drawCircle(Color.red, 3);   // draws a red circle, radius 3, center (0,0)
BasicGLib.drawRect(Color.blue, 3, 5); // draws blue rectangle
                                      // with coord (0,0),(3,0),(3,5),(0,5)

BasicGLib.moveCursor(2, 2);
BasicGLib.drawCircle(Color.green, 3); // draws green circle of radius 3
                                      // and center (2,2)

BasicGLib.drawRect(Color.pink, 3, 5); // draws pink rectangle
                                      // with coord (2,2),(5,2),(5,7),(2,7)

BasicGLib.moveCursor(-2, -2);         // cursor back to (0,0)
```

The draw methods above operate in the XOR mode --- i.e redrawing causes erasing. For example:

```
BasicGLib.clear();                    // initialize
```

```
BasicGLib.drawCircle(Color.red, 3);   //draws a red circle, radius 3, center (0,0)
BasicGLib.drawCircle(Color.red, 3);   //draws a red circle, radius 3, center (0,0)
```

The net effect of this program is that nothing happens since the second draw erases the effect of the first. In this exam, we will write code to build and manipulate complex graphics objects that are built out of circles and rectangles. The architecture of our design is as follows (the description of ShapeVisitor is postponed till later):

```
interface Shape extends Cloneable {
    void draw(Color c);
    Object accept(ShapeVisitor v);
    Object clone();
}
```

```
class Circle implements Shape {
    int r;
    public Circle(int radius) { r = radius; }
    public void draw(Color c) { BasicGLib.drawCircle(c, r); }
    public Object accept(ShapeVisitor v) { return v.visitCircle(this); }
    public Object clone() {
        try {
            return super.clone();
        } catch (CloneNotSupportedException e) {
            throw new InternalError();
        }
    }
}
```

```
class Rectangle implements Shape {
    int xl, yl;
    public Rectangle(int xlength, int ylength) { xl = xlength; yl = ylength; }
    public void draw(Color c) { BasicGLib.drawRectangle(c, xl, yl); }
    public Object accept(ShapeVisitor v) { return v.visitRectangle(this); }
    public Object clone() {
        try {
            return super.clone();
        } catch (CloneNotSupportedException e) {
            throw new InternalError();
        }
    }
}
```

Now, we look into stuff that we need to build complex shapes. As expected, we begin with a class ComplexShape. Answer the following questions

Problem 1

Finish the following code.

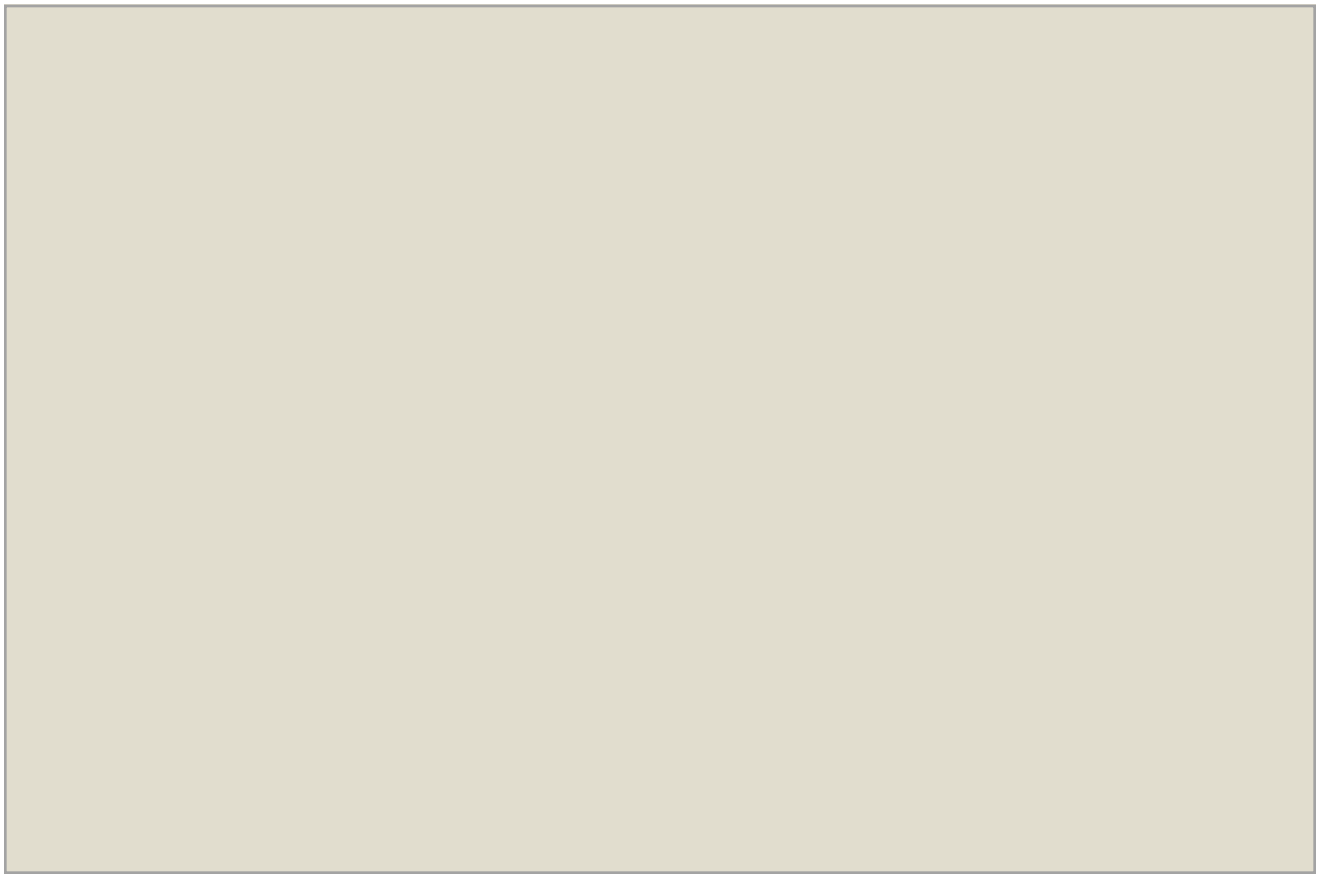
```
class ComplexShape implements Shape {
```

```
    ArrayList shapes = new ArrayList();   // a list of shapes

    /** add a shape to the composite */
    public void addShape(Shape s) {









    }

    /** draw all shapes in the composite */
    public void draw(Color c) {

















    }

    public Object clone() {

















    }

    public Object accept(ShapeVisitor v) { return v.visitComplexShape(this); }

}
```

Problem 2

Use the classes Circle, Rectangle and ComplexShape to write Java code to construct a complex shape that consists of:

- A circle of radius 3 with center at (0,0).
- A circle of radius 5 with center at (0,0)
- A square of side 8 with lower left corner at (0,0)



Problem 3

Now, we focus on the ability to construct complex shapes. Consider the object consisting of :

- A circle of radius 3 at center (0,0)
- A circle of radius 5 at center (2,3)

The class Translate below is intended to help in describing such objects. For example, the above object would be constructed as:

```
Shape o = new ComplexShape();
o.addShape(new Circle(3));
o.addShape(new Translate(2, 3, new Circle(5)));
```

Finish the following code.

```
class Translate implements Shape{

    Shape sh;

    int xtrans, ytrans;
```

```
    public Translate(int x, int y, Shape sh){
        xtrans = x; ytrans = y;
        this.sh = sh;
    }

    public void draw(Color c) {
        // Hint: Move cursor by amount xtrans, ytrans; draw;
        // and then restore the original position of cursor









    }

    public Object clone() {









    }

    public Object accept(ShapeVisitor v) { return v.visitTranslate(this); }

}
```

Problem 4

Use the classes Circle, Rectangle, Translate and ComplexShape to write Java code to construct a complex shape that consists of:

- A circle of radius 3 with center at (0,0).
- A circle of radius 5 with center at (3,4).
- A complex shape that consists of A circle of radius 3 at the (8,9), and a circle of radius 5 with center (10,12)

Problem 5

The interface ShapeVisitor is as follows:

```
interface ShapeVisitor {
    Object visitCircle(Circle c);
    Object visitRectangle( Rectangle r);
    Object visitComplexShape(ComplexShape c);
    Object visitTranslate(Translate t);
}
```

Write the code for a PrintVisitor that will print out a Shape with nice indentation. For example, the code:

```
Shape o = new ComplexShape();
o.addShape(new Circle(3));
o.addShape(new Translate(2, 3, new Circle(5)));
System.out.println(o.accept(new PrintVisitor()));
```

should output:

```
ComplexShape(
    Circle(3),
    Translate(2, 3,
        Circle(3)
    )
)
```

```
class PrintVisitor implements ShapeVisitor {
```

```
}
```

Problem 6

Write the code for a BBoxVisitor that computes the bounding box of a Shape. The bounding box of a shape is the smallest rectangle with sides parallel to the axis that contains the entire shape. For example:

- A circle of radius 3 with center at (0,0) has bounding box (xleft = -3, xright = +3; ydown = -3, yup = +3).
- A circle of radius 5 with center at (3,4) has bounding box: (xleft = -2, xright = 8; ydown = -1, yup = 9).
- A complex shape that consists of: a circle of radius 3 with center (8,9), and a circle of radius 5 with center (10,12) has bounding box (xleft = 5, xright = 15; ydown = 6, yup = 17)

```
class BBox {
    int xleft, xright;
    int ydown, yup;
    public BBox(int xl, int xr, int yd, int yu) { /* ... */ }
}

class BBoxVisitor implements ShapeVisitor {
```

```
}
```

Problem 7

Write a method `renderShape` that draws the shape sh with color fore. The color fore is used to color the part of the bounding box of the shape that is occupied by the shape --- i.e. the background part of the bounding box is colored with color back.

```
void renderShape(Shape sh, Color back, Color fore) {









}
```