SEPTEMBER 2004 EXAMINATIONS

Bachelor of Arts : Year 2
Bachelor of Arts : Year 3
Bachelor of Engineering : Year 2
Bachelor of Science : Year 1
Bachelor of Science : Year 2

Advanced Object-Oriented Programming

TIME ALLOWED : Two hours and a half

INSTRUCTIONS TO CANDIDATES

Answer all questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).
In this exam, you are asked to develop some Java code that could be used for a simple tool that allows a user to draw a variety of shapes on a canvas. You are also asked to comment on some relevant design and programming issues. Some questions ask you to write Java code: for these questions, you should write code that is as detailed and as close to Java syntax as possible; however, minor divergences from Java syntax, and minor errors in the names of classes and methods in Java packages such as the AWT, will not be penalised. If you are uncertain about the syntactical constructs, some marks will be given for pseudocode, but you should make the pseudocode as detailed as possible.

1. If a shape is to be drawn on a canvas, it will have a particular location on the canvas; this location can be represented by x- and y-coordinates representing an ‘anchor point’ for the shape (for example, the anchor point for a rectangle might be its top left corner; for a circle, the anchor point might be the center of the circle). A shape will also have a particular colour. Thus, a shape might be represented in Java by a class which has fields to represent the x- and y-coordinates of its anchor point, and a field to represent a colour (say using the java.awt.Color class). These fields represent the kind of data that is common to all shapes; obviously, however, particular shapes differ in the way that they are drawn: circles look different to rectangles. The following is an abstract Java class that is intended to capture the common data representation of anchor points and colours; it will have concrete subclasses to represent the different kinds of shapes, so there will be one concrete subclass for circles, another for rectangles, and so on. Each concrete subclass will implement the draw() method in a way that is appropriate for drawing a particular kind of shape (e.g., a circle).

```java
import java.awt.Graphics;
import java.awt.Color;

public abstract class AbstractShape
{
    // The x-coordinate of the anchor point
    private int x;

    // The y-coordinate of the anchor point
    private int y;

    // The colour of the shape
    private Color color;

    public AbstractShape(int x, int y, Color c)
    {
        this.x = x;
        this.y = y;
        color = c;
    }
}
```
public Color getColor()
{
    return color;
}

// abstract method to draw the shape,
// given a Graphics context
public abstract void draw(Graphics g);

(a)  i. What are the differences between the scope modifiers private, protected
     and public?  [9 marks]
   ii. The x and y fields in AbstractShape are declared private. Does this raise
       any problem in implementing concrete subclasses of AbstractShape, and if
       so, what changes could be made to the AbstractShape class to deal with the
       problem?  [6 marks]

(b) Give Java code for a concrete subclass Rectangle of AbstractShape, which
      will represent a rectangle that can be drawn on a canvas. (A useful method for
      this class is the

          java.awt.Graphics.fillRect(int, int, int, int),

      method, which draws a rectangle with top left corner at the x- and y-coordinates
      specified by the first two parameters, and whose width and height are specified by
      the third and fourth parameters.) Here, and in all remaining questions, you should
      assume that the AbstractShape class contains any modifications given in your
      answer to Question 1(a)(ii) above.  [10 marks]

2. A graphical user-interface (GUI) for a tool to allow users to draw shapes on a canvas
might be built up using the following code, which would be called from the constructor
of a top-level class, say called DrawTool, that extends the javax.swing.JFrame class (the
ShapeCanvas class will be discussed in Question 3):

    Container pane = getContentPane();

    // a TextField to display simple instructions to the user
    final TextField instructions = new TextField(40);
    instructions.setEditable(false);
    instructions.setText("Select a shape to draw");

    pane.add(instructions, BorderLayout.NORTH);

    // A Canvas to display the drawn shapes
    final ShapeCanvas canvas = new ShapeCanvas();

    pane.add(canvas, BorderLayout.CENTER);
// controls to allow the user to select a shape
// to be drawn
Panel shapesPanel = new Panel();
// add controls here ...

// controls to allow the user to select a colour
// for the shape to be drawn
Panel colorsPanel = new Panel();
// add controls here ...

// add the controls at the right-hand side
Panel choices = new Panel();
choices.setLayout(new GridLayout(2,1));
choices.add(shapesPanel);
choices.add(colorsPanel);
pane.add(choices, BorderLayout.EAST);

The controls at the right-hand side allow the user to select a shape (e.g., a circle, or a rectangle) to draw, and a colour for the shape.

(a) What GUI components, from either the java.awt or javax.swing packages would be appropriate for allowing the user to select a kind of shape to draw on the canvas? [5 marks]

(b) Assuming that the user can only choose between drawing rectangles or circles, give Java code that adds the appropriate GUI components you identified in part (a) above. [10 marks]

(c) When a user uses these controls to select 'Rectangle', it would be useful to use the instructions TextField to prompt the user with the message ‘Drag the mouse to set the diagonal’. Give Java code that creates and registers an appropriate event listener to the GUI component for selecting rectangles. The event listener should set the text of the instructions TextField to ‘Drag the mouse to set the diagonal’ (using the

\[
\text{java.awt.TextField.setText(String)}
\]

method). [10 marks]
3. The `ShapeCanvas` class referred to in the previous question will do the following:

- store and display all the shapes that have been drawn on the canvas, and
- allow the user to add a new shape of the chosen kind and colour by dragging the mouse on the canvas.

(a) The shapes that have been drawn on the canvas will be stored in an array, `shapes`, of length 200 and of type `AbstractShape[200]`.

An integer variable `index` will hold a count of the number of shapes that have been added to the canvas; this variable will be incremented each time a new shape is added to the canvas.

Give Java code that declares and initialises the variables `shapes` and `index` as fields of the `ShapeCanvas` class. [5 marks]

(b) The `ShapeCanvas` class should also store information that says what kind of shape and what colour have been selected by the user from the controls given in Question 2. The information should be stored in fields of the `ShapeCanvas` class, and there should be methods that allow the values in these fields to be set when the user interacts with the GUI components.

i. Assuming there are just two kinds of shape that the user can choose, rectangles and circles, give Java code that declares a field, `shapeKind`, to store information saying what shape has been selected, and a method to update the field. (Note that you will have to choose some data representation for this information, and this data is used in two separate classes: in the `ShapeCanvas` class, where it is stored, and in the `DrawTool` class, where the updating method will be called. Your data representation may use an existing Java data type, or even declare a new class, so your answer may involve more than just a simple field declaration.) [10 marks]

ii. Modify your answer to Question 2(c) so that when the user selects a rectangle to be drawn, the `shapeKind` field is updated appropriately. [5 marks]

(c) The user adds a new shape to the canvas by selecting the kind of shape and its colour and then dragging the mouse on the canvas. Assume the available kinds of shape are rectangles and circles, implemented by concrete subclasses of `AbstractShape` called `Rectangle` and `Circle`, respectively. To draw a rectangle, the user clicks and holds the mouse where the top left corner of the rectangle is to be placed, and drags the mouse and releases it where the bottom right corner is to be placed. To draw a circle, the user clicks and holds the mouse where the centre of the circle is to be placed and drags the mouse and releases it at any point on the edge of the circle. To capture these mouse events, the `ShapeCanvas` class implements the `java.awt.event.MouseListener` interface. The relevant methods in this interface are
public void mousePressed(MouseEvent e);

public void mouseReleased(MouseEvent e);

When the mouse is pressed, the x- and y-coordinates of the mouse’s location are stored in fields anchorPointX and anchorPointY:

class ShapeCanvas implements MouseListener
{
    private int anchorPointX;
    private int anchorPointY;

    public void mousePressed(MouseEvent e)
    {
        anchorPointX = e.getX();
        anchorPointY = e.getY();
    }

    ...
}

Give Java code for the method mouseReleased that creates an instance of the appropriate class (Rectangle or Circle, depending on the value currently stored in the shapeKind field), and with the appropriate colour, which you may assume is stored in a field shapeColor of type Color, and adds it to the array shapes. You may assume that the Circle class has a constructor

Circle(int ax, int ay, int ex, int ey, Color c)

whose first two arguments are the x- and y-coordinates of the centre of the circle, and whose third and fourth arguments are the x- and y-coordinates of any point on the edge of the circle.  

(d) Describe the ‘Model-View-Controller’ architecture, and say to what extent the classes DrawTool, ShapeCanvas, and AbstractShape, together with its concrete subclasses for rectangles, etc., implement this architecture.  

[15 marks]