



NATIONAL DIPLOMA IN COMPUTING
(Information Technology)

Object Orientation with Design Patterns
CM302

Semester I

Internal Examiner(s): Ms. Orla McMahon

External Examiner(s): Mr John Dunnion
Prof. Gerard Parr

January 2004
Time of examination here

Instructions to candidates:

- 1) Section A: Attempt any five parts.**
- 2) Section B: Answer any 3 Questions.**
- 3) All questions carry equal marks.**

DO NOT TURN OVER THIS PAGE UNTIL YOU ARE TOLD TO DO SO

Section A

Attempt any 5 parts of this question

(5 marks each)

a)	<p>What are the overall goals of the Design Pattern approach to software development?</p> <p>In particular how does it differ from more traditional methods?</p> <p>[5 Marks]</p>
b)	<p>Any object in Java can be queried to see what methods and parameters make up its interface.</p> <p>Write a code sample that demonstrates how you would query a Java object for it's methods only.</p> <p>[5 Marks]</p>
c)	<p>Discuss the following code sample with regard to FlyWeight classes:</p> <pre>String s1 = new String("Hello"); String s2 = new String("Hello"); if (s1 == s2) System.out.println("Same");</pre> <p>[5 Marks]</p>
d)	<p>Name five Behavioral Design Patterns</p> <p>[5 Marks]</p>
e)	<p>Briefly describe the difference between the Class Adapters and Object Adapters.</p> <p>[5 Marks]</p>
f)	<p>Briefly describe how the Proxy Design Pattern works and give three situations where it might be used.</p> <p>[5 Marks]</p>
g)	<p>What is reflection.</p> <p>Using a Java code example illustrate how can it be used to create dynamic adapters.</p> <p>[5 Marks]</p>

(25 marks)

Section B

Candidates should attempt any 3 of the following questions.

Question 2

2a)	<p>In general a pattern has four essential elements.</p> <p>Identify and describe the four essential elements.</p> <p>[4 Marks]</p>
2b)	<p>What is meant by the term creational patterns.</p> <p>Name four creational patterns.</p> <p>[6 Marks]</p>
2b)	<p>The easiest way to create a class that can have only one instance is to embed a static variable inside of the class that is set on the first instance and then check for it each time that you enter the constructor.</p> <p>i) Write a Java program that implements the statements above. [7 Marks]</p> <p>ii) Write a Java program that tests this implementation. [4 Marks]</p> <p>iii) What design pattern have you implemented? [1 Marks]</p> <p>iv) Using two real world examples, describe why you might use this pattern. [3 Marks]</p>

(25 marks)

Question 3

3a)	<p>The program given in code listing 1 (below) uses a simple Decorator Pattern to create a cool button. A cool button is a button for which borders appear only when the user moves the mouse over the button.</p> <p>Describe in detail the role each class plays in order to implement the Decorator Pattern, along with an explanation of each of the class methods.</p> <p style="text-align: right;">[13 Marks]</p>
3b)	<p>The Adapter Pattern allows unrelated classes to work together in a program. Name the two mechanisms that can be used to do this and briefly describe both approaches.</p> <p>Using code samples show one case where the Adapter Pattern is used in Java.</p> <p style="text-align: right;">[6 Marks]</p>
3c)	<p>Creating objects by using a class name directly can lead to problems. What are these problems and how can they be overcome?</p> <p>Give a simple example.</p> <p style="text-align: right;">[6 Marks]</p>

(25 marks)

Code Listing 1

Decorater.Java

```
public class Decorator extends JComponent {  
    public Decorator(JComponent c) {  
        setLayout(new BorderLayout());  
        add("Center", c);  
    }  
}
```

CoolDecorator.java

```
public class CoolDecorator extends Decorator {
    boolean mouse_over;    //true when mose over button
    JComponent thisComp;

    public CoolDecorator(JComponent c) {
        super(c);
        mouse_over = false;
        thisComp = this;    //save this component
        //catch mouse movements in inner class
        c.addMouseListener(new MouseAdapter() {
            //set flag when mouse over
            public void mouseEntered(MouseEvent e) {
                mouse_over = true;
                thisComp.repaint();
            }
            //clear flag when mouse not over
            public void mouseExited(MouseEvent e) {
                mouse_over = false;
                thisComp.repaint();
            }
        });
    }

    //paint the button
    public void paint(Graphics g) {
        super.paint(g);    //first draw the parent button
        //if the mouse is not over the button
        //erase the borders
        if (! mouse_over) {
            Dimension d = super.getSize();
            g.setColor(Color.lightGray);
            g.drawRect(0, 0, d.width-1, d.height-1);
            g.drawLine(d.width-2, 0, d.width-2, d.height-1);
            g.drawLine(0, d.height-2, d.width-2, d.height-2);
        }
    }
}
```

SlashDecorator.java

```
public class SlashDecorator extends Decorator {
    int x1, y1, w1, h1;

    public SlashDecorator(JComponent c) {
        super(c);
    }
    public void setBounds(int x, int y, int w, int h) {
        x1 = x; y1= y;
        w1 = w; h1 = h;
        super.setBounds(x, y, w, h);
    }
    public void paint(Graphics g) {
        super.paint(g);
        g.setColor(Color.red);
        g.drawLine(0, 0, w1, h1);
    }
}
```

slashWindow.java

```
public class slashWindow extends JFrame
    implements ActionListener
{
    JButton CButton, DButton, Quit;
    public slashWindow()
    {
        super ("Deco Button");
        JPanel jp = new JPanel();

        getContentPane().add(jp);

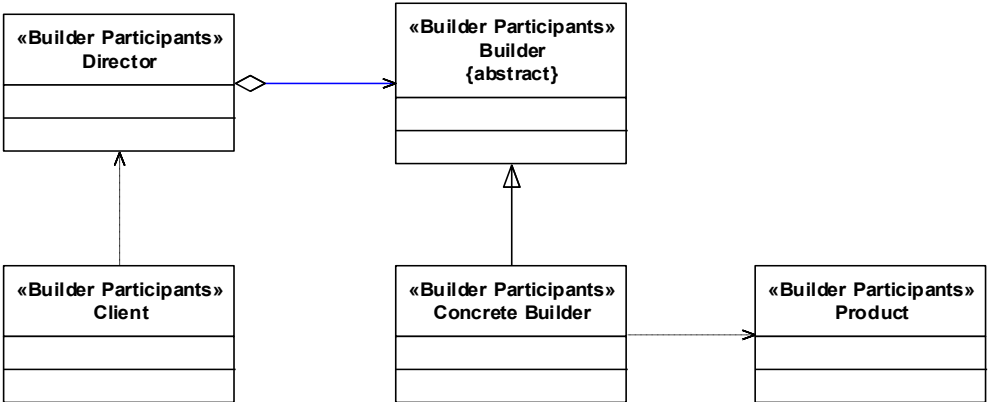
        jp.add( new CoolDecorator(
            CButton = new JButton("Cbutton")));

        jp.add( new SlashDecorator(new CoolDecorator(
            DButton = new JButton("Dbutton"))));

        jp.add(Quit = new JButton("Quit"));
        Quit.addActionListener(this);
        setSize(new Dimension(200,100));

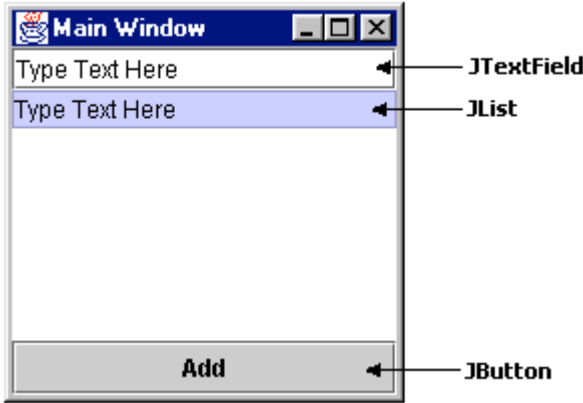
        setVisible(true);
        Quit.requestFocus();
    }
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
    static public void main(String argv[])
    {
        new slashWindow();
    }
}
```

Question 4

4a)	<p>With the aid of some simple UML diagrams, describe the difference between the Factory Pattern, Abstract Factory Pattern and the Factory method pattern.</p> <p style="text-align: right;">[12 Marks]</p>
4b)	<p>What is the Builder Pattern? Why would you use it. Illustrate your answer with two real-world examples.</p> <p style="text-align: right;">[4 Marks]</p>
4c)	<p>Given the following UML diagram, explain briefly the role of each participant in the Builder Pattern.</p>  <pre> classDiagram class Director { <<Builder Participants>> } class Builder { <<Builder Participants>> <<abstract>> } class Client { <<Builder Participants>> } class ConcreteBuilder { <<Builder Participants>> } class Product { <<Builder Participants>> } Director o--> Builder Client -- > Director ConcreteBuilder -- > Builder ConcreteBuilder --> Product </pre> <p style="text-align: right;">[5 Marks]</p>
4d)	<p>Describe four consequences of the Builder Pattern.</p> <p style="text-align: right;">[4 Marks]</p>

(25 marks)

Question 5

5a)	<p>The Chain of Responsibility Pattern does not have to use a linear chain. What does this statement mean?</p> <p>What, if any, implications are there if a non-linear chain is used?</p> <p style="text-align: right;">[4 Marks]</p>
5b)	<p>When you build a Java user interface, you provide controls – menu items, buttons, check boxes, and so on – to allow the user to tell the program what to do. When a user selects one of these controls, the program receives an ActionEvent which it must trap by implementing the ActionListener interfaces (actionPerformed). This code can get quite cumbersome if there are many controls that make up the user interface.</p> <p>Describe with the aid of some sample code one method that can be used to reduce the amount of coding in the actionPerformed method by forwarding specific commands to specific user interface controls.</p> <p style="text-align: right;">[10 Marks]</p>
5c)	<p>Write a program that allows the user to add items to a JList control by typing text into a JTextField and clicking on an add button. Your program should take advantage of the Model View Controller (MVC) architecture.</p> <p>In developing your program you must incorporate the following classes</p> <ul style="list-style-type: none"> • MainWindow extends JFrame • ListData extends AbstractListModel <div style="text-align: center;">  </div> <p style="text-align: right;">[11 Marks]</p>

(25 marks)