1a-Write a program to create a class and implement a default, overloaded and copy Constructor.

```
import java.util.*;
    class Student {
        int roll;
        String name;
        public Student() {
            roll = 101;
            name = "Anurag";        }
        public Student(int r, String n) {
            roll = r;
            name = n;        }
        public Student(Student s) {
            roll = s.roll;
            name = s.name;       }
        void display() {
            System.out.println("Roll No=" + roll + "\tName=" + name);
}      }
    class ConstructorDemo {
        public static void main(String args[]) {
            Student s1 = new Student();
            s1.display();
            Student s2 = new Student(111, "charvi");
            s2.display();
            Student s3 = new Student(s1);
            s3.display();      }      }
```

1b-Write a program to create a class and implement the concepts of Method Overloading.

```
class MethodOverload {
        public int sum(int x, int y) {
            return (x + y);       }
        public int sum(int x, int y, int z) {
            return (x + y + z);       }
        public double sum(double x, double y) {
            return (x + y);       }
        public static void main(String args[]) {
            MethodOverload m = new MethodOverload();
            System.out.println(m.sum(10, 20));
            System.out.println(m.sum(10, 20, 30));
            System.out.println(m.sum(10.5, 20.5));       }       }
```

1c-Write a program to create a class and implement the concepts of Static methods .

```
class Demo {
        void display() {
            System.out.println("A non-static method is called");     }
        static void show() {
            System.out.println("A static method is called");     }   }
    public class StaticDemo {
        public static void main(String args[]) {
            Demo obj = new Demo();
            obj.display();
            Demo.show();     }    }
```

2a-Write a program to implement the concepts of Inheritance and Method overriding.

```
class Base {
        void show() {
            System.out.println("Base class show() method invoked");      }
}
    class Derived extends Base { // Single level inheritance
        void show() {
            System.out.println("Derived class show() method invoked");
}   }
    class OverDemo {
        public static void main(String args[]) {
            Derived d = new Derived();
            d.show();    }    }
```

2c - Write a program to implement the concept of interfaces .

```
import java.util.*;
    interface A {
        void getA();    }
    interface B {
        void getB();    }
    class C implements A, B {
        public void getA() {
            System.out.println("getA() method is invoked");    }
        public void getB() {
            System.out.println("getB() method is invoked");   }   }
    class MultipleDemo {
        public static void main(String args[]) {
            C c1 = new C();
            c1.getA();
            c1.getB();    }   }
```

2b-Write a program to implement the concepts of Abstract classes and methods .

```
import java.util.*;
abstract class Shape {
        abstract void area();
    }
    class Rectangle extends Shape {
        void area() {
            double length, width;
            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter length of rectangle:");
            length = scanner.nextDouble();
            System.out.println("Enter width of rectangle:");
            width = scanner.nextDouble();
            System.out.println("Area of rectangle: " + (length * width));
        }      }
    class Circle extends Shape {
        void area() {
            double radius;
            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter radius of circle:");
            radius = scanner.nextDouble();
            System.out.println("Area of circle: " + (Math.PI * radius *
radius));
        }     }
    class AbstractDemo {
        public static void main(String args[]) {
            Rectangle rectangle = new Rectangle();
            rectangle.area();

            Circle circle = new Circle();
            circle.area();      }      }
```

3a - Write a program to raise built-in exceptions and raise them as per the requirements .

```
import java.io.*;
    public class ExceptionDemo {
        public static void main(String args[]) throws IOException {
            int n1 = 10, n2 = 0;
            int a[] = {1, 2, 3};
            int d1, d2;
            System.out.println("Handling Arithmetic Exception:");
            try {
                d1 = n1 / n2;
            } catch (ArithmeticException e) {
                System.out.println("Division by Zero exception: " + e);     }
            System.out.println("Handling Array Index Out Of Bounds
Exception");
            try {
                d2 = a[0] / a[3];
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Division by array index out of bound
exception: " + e);    }    }    }
```

3b - Write a program to define user defined exceptions and raise them as per the requirements .

```
import java.util.*;
    class AgeNotMatchException extends Exception {
        AgeNotMatchException(String msg) {
            super(msg);        }     }
    class Student {
        private String name;
        private int age;
        public Student(String name, int age) {
            this.name = name;
            this.age = age;
            try {
                if (age < 15 || age > 20) {
                    String msg = "Age is not between 15 and 20";
    AgeNotMatchException ae = new AgeNotMatchException(msg);
                    throw ae;                }
            } catch (AgeNotMatchException e) {
                e.printStackTrace();            }        }
        public void display() {
            System.out.println("Name of the Student: " + this.name);
            System.out.println("Age of the Student: " + this.age);          }
}
    class MyExceptionDemo {
        public static void main(String args[]) {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the name of the student:");
            String name = sc.next();
            System.out.println("Enter the age of the student (should be
between 15 and 20 inclusive):");
            int age = sc.nextInt();
            Student obj = new Student(name, age);
            obj.display();        }    }
```

4 - Write a java application to demonstrate 5 bouncing balls of different colors using threads.

```
import java.applet.Applet;
    import java.awt.Color;
    import java.awt.Graphics;
    class Ball {
        int x, y, radius, dx, dy;
        Color ballColor;
        public Ball(int x, int y, int radius, int dx, int dy, Color ballColor) {
            this.x = x;
            this.y = y;
            this.radius = radius;
            this.dx = dx;
            this.dy = dy;
            this.ballColor = ballColor;        }     }
    public class BouncingBall extends Applet implements Runnable {
        Ball redBall, blackBall, greenBall, blueBall, pinkBall;
        public void init() {
            redBall = new Ball(200, 200, 20, 2, 10, Color.red);
            blackBall = new Ball(160, 190, 20, 4, 8, Color.black);
            greenBall = new Ball(120, 180, 20, 6, 6, Color.green);
            blueBall = new Ball(80, 170, 20, 8, 4, Color.blue);
            pinkBall = new Ball(40, 160, 20, 10, 2, Color.pink);
            Thread t = new Thread(this);
            t.start();        }
        public void paint(Graphics g) {
            g.setColor(redBall.ballColor);
            g.fillOval(redBall.x, redBall.y, redBall.radius, redBall.radius);
            g.setColor(blackBall.ballColor);
            g.fillOval(blackBall.x,         blackBall.y,         blackBall.radius,
blackBall.radius);
            g.setColor(greenBall.ballColor);
            g.fillOval(greenBall.x,         greenBall.y,         greenBall.radius,
greenBall.radius);
            g.setColor(blueBall.ballColor);
            g.fillOval(blueBall.x,          blueBall.y,          blueBall.radius,
blueBall.radius);
            g.setColor(pinkBall.ballColor);
            g.fillOval(pinkBall.x,          pinkBall.y,          pinkBall.radius,
pinkBall.radius);       }
        public void run() {
            while (true) {
                try {
                    displacementOperation(redBall);
                    displacementOperation(blackBall);
                    displacementOperation(greenBall);
                    displacementOperation(blueBall);
                    displacementOperation(pinkBall);
                    Thread.sleep(20);
                    repaint();
                } catch (Exception e) {
        }           }       }
        public void displacementOperation(Ball ball) {
            if (ball.y >= 400 || ball.y <= 0) {
                ball.dy = -ball.dy;          }
            if (ball.x >= 400 || ball.x <= 0) {
                ball.dx = -ball.dx;       }
            ball.y = ball.y - ball.dy;
            ball.x = ball.x - ball.dx;       }     }
```

6a - Create a swing application that randomly changes color on button click.

```
import java.awt.*;
    import java.awt.event.*;
    import javax.swing.*;
    public class ColourChangeApp extends JFrame implements
ActionListener {
        JButton b1;
        Container c;
        public ColourChangeApp() {
            c = getContentPane();
            c.setLayout(new FlowLayout());
            b1 = new JButton("Change colour");
            b1.addActionListener(this);
            c.add(b1);
            setSize(400, 400);
            setTitle("Colour Changing Window");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setVisible(true);        }
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == b1) {
                int R = (int) (Math.random() * 100) % 255;
                int G = (int) (Math.random() * 100) % 255;
                int B = (int) (Math.random() * 100) % 255;
                Color mycolor = new Color(R, G, B);
                c.setBackground(mycolor);            }       }
        public static void main(String args[]) {
            new ColourChangeApp();       }     }
```

6c - Create a Swing application to demonstrate use of scrollpane to change its color selected using colour chooser.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class ScrollPaneApp extends JFrame implements ActionListener {
JScrollPane sp;
 JButton b;
 JTextArea ta;
ScrollPaneApp() {
setSize(500, 500);
setTitle("Colour Change ScrollPane Application");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new FlowLayout());
ta = new JTextArea("This is Textarea in ScrollPane", 2, 20);
sp = new JScrollPane(ta);
sp.setPreferredSize(new Dimension(200, 80));
sp.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_
SCROLLBAR_ALWAYS);
sp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL
_SCROLLBAR_ALWAYS);   add(sp);
b = new JButton("Change Color");   add(b);
sp.setBorder(BorderFactory.createLineBorder(Color.RED));
b.addActionListener(this);
setVisible(true);   }
public void actionPerformed(ActionEvent e) {
Color initialColor = Color.BLUE;
Color selectedColor = JColorChooser.showDialog(this, "Select a Background Color", initialColor);
sp.setBackground(selectedColor);        }
public static void main(String[] args) {
new ScrollPaneApp();        }     }
```

7a - Flow Layout

```
import java.awt.*;
    import javax.swing.*;
    public class MyFlowLayout {
       JFrame f = new JFrame();
       MyFlowLayout() {
          JButton b1 = new JButton("1");
          JButton b2 = new JButton("2");
          JButton b3 = new JButton("3");
          JButton b4 = new JButton("4");
          JButton b5 = new JButton("5");
          f.add(b1);
          f.add(b2);
          f.add(b3);
          f.add(b4);
          f.add(b5);
          f.setLayout(new FlowLayout(FlowLayout.RIGHT));
          f.setSize(300, 300);
          f.setVisible(true);        }
       public static void main(String args[]) {
          new MyFlowLayout();        }     }
```

7b - Grid Layout

```
import java.awt.*;
    import javax.swing.*;
    public class MyGridLayout {
       MyGridLayout() {
          JFrame f = new JFrame();
          JButton b1 = new JButton("1");
          JButton b2 = new JButton("2");
          JButton b3 = new JButton("3");
          JButton b4 = new JButton("4");
          JButton b5 = new JButton("5");
          JButton b6 = new JButton("6");
          JButton b7 = new JButton("7");
          JButton b8 = new JButton("8");
          JButton b9 = new JButton("9");
          f.add(b1);
          f.add(b2);
          f.add(b3);
          f.add(b4);
          f.add(b5);
          f.add(b6);
          f.add(b7);
          f.add(b8);
          f.add(b9);
          f.setLayout(new GridLayout(3, 3));
          f.setSize(300, 300);
          f.setVisible(true);        }
       public static void main(String args[]) {
          new MyGridLayout();        }     }
```

7c - Border Layout

```
import java.awt.*;
    import javax.swing.*;
    public class MyBorderLayout {
       MyBorderLayout() {
          JFrame f = new JFrame();
          JButton b1 = new JButton("NORTH");
          JButton b2 = new JButton("SOUTH");
          JButton b3 = new JButton("EAST");
          JButton b4 = new JButton("WEST");
          JButton b5 = new JButton("CENTER");
          f.add(b1, BorderLayout.NORTH);
          f.add(b2, BorderLayout.SOUTH);
          f.add(b3, BorderLayout.EAST);
          f.add(b4, BorderLayout.WEST);
          f.add(b5, BorderLayout.CENTER);
          f.setSize(300, 300);
          f.setVisible(true);        }
       public static void main(String args[]) {
          new MyBorderLayout();        }     }
```

8a - Events: Write programs to demonstrate the following events:

```
import java.awt.*;
    import java.awt.event.*;
    public class ActionEventExample implements ActionListener {
       TextField tf;
       Button b;
       Frame f;
       ActionEventExample() {
          f = new Frame("ActionEvent Example");
          tf = new TextField();
          tf.setBounds(50, 50, 200, 20);
          b = new Button("Click Here");
          b.setBounds(50, 100, 60, 30);
          b.addActionListener(this);
          f.add(b);
          f.add(tf);
          f.setSize(400, 400);
          f.setLayout(null);
          f.setVisible(true);        }
       public void actionPerformed(ActionEvent e) {
          tf.setText("Welcome to VIVA College");        }
       public static void main(String[] args) {
          new ActionEventExample();        }     }
```

8b – MouseEvent

```
import java.awt.*;
    import java.awt.event.*;
    public class MouseEventExample extends Frame implements MouseListener {
       Label l;
       MouseEventExample() {
          l = new Label();
          l.setBounds(20, 50, 100, 20);
          add(l);
          setSize(300, 300);
          setLayout(null);
          setVisible(true);
          addMouseListener(this);     }
       public void mouseClicked(MouseEvent e) {
          l.setText("Mouse Clicked");       }
       public void mouseEntered(MouseEvent e) {
          l.setText("Mouse Entered");        }
       public void mouseExited(MouseEvent e) {
          l.setText("Mouse Exited");        }
       public void mousePressed(MouseEvent e) {
          l.setText("Mouse Pressed");        }
       public void mouseReleased(MouseEvent e) {
          l.setText("Mouse Released");        }
       public static void main(String args[]) {
          new MouseEventExample();        }     }
```

8c – KeyEvent

```
import java.awt.*;
    import java.awt.event.*;
    public class KeyEventExample extends Frame implements KeyListener {
       Label l;
       TextArea area;
       KeyEventExample() {
          l = new Label();
          l.setBounds(20, 50, 100, 20);
          area = new TextArea();
          area.setBounds(20, 80, 300, 300);
          add(l);
          add(area);
          setSize(400, 400);
          setLayout(null);
          setVisible(true);
          area.addKeyListener(this);        }
       public void keyPressed(KeyEvent e) {
          l.setText("Key Pressed");        }
       public void keyReleased(KeyEvent e) {
          l.setText("Key Released");        }
       public void keyTyped(KeyEvent e) {
          l.setText("Key Typed");        }
       public static void main(String[] args) {
          new KeyEventExample();        }     }
```

9 - Demonstrate the use of Adapter Class in Event Handling .

```
import java.awt.*;
    import java.awt.event.*;
    public class AdapterExample extends Frame {
       AdapterExample() {
          addWindowListener(new MyInnerClass());
          setSize(300, 300);
          setLayout(null);
          setVisible(true);
          setTitle("Adapter Class example");        }
       class MyInnerClass extends WindowAdapter {
          public void windowClosing(WindowEvent e) {
             dispose();          }       }
       public static void main(String[] args) {
          new AdapterExample();        }     }
```

10 - Demonstrate the use of Anonymous Inner Class in Event Handling

```
import java.awt.*;
    import java.awt.event.*;
    public class AnonymousAdapter extends Frame {
       AnonymousAdapter() {
          addMouseListener(new MouseAdapter() {
             public void mouseClicked(MouseEvent e) {
                Graphics g = getGraphics();
                g.setColor(Color.BLUE);
                g.fillOval(e.getX(), e.getY(), 30, 30);              }
          });
          setSize(300, 300);
          setLayout(null);
          setVisible(true);        }
       public static void main(String[] args) {
          new AnonymousAdapter();        }     }
```