Pract 1a:- constructor overloading

class Demo

{

public Demo()

{

System.out.println("This is default constructor");

}

public Demo(int n1,int n2)

{

System.out.println("Multiplication="+(n1\*n2));

}

}

class B

{

public static void main(String args[])

{

Demo d1=new Demo();

Demo d2=new Demo(10,20);

}

}

Pract 1b:- method overloading

class Demo

{

public void add(int n1,int n2)

{

System.out.println(n1+n2);

}

public void add(float n1,float n2)

{

System.out.println(n1+n2);

}

public void add(int n1,int n2,int n3)

{

System.out.println(n1+n2+n3);

}

}

public class addition

{

public static void main(String args[])

{

Demo d=new Demo();

d.add(10,20);

d.add(115,136);

d.add(55,65,75);

}

}

Pract 1c:- static method

class DemoStaticMethods

{

public static int add(int a,int b)

{

return a+b;

}

public static int subtract(int a,int b)

{

return a-b;

}

}

public class Pr1c

{

public static void main(String args[])

{

int sum=DemoStaticMethods.add(8,4);

int difference=DemoStaticMethods.subtract(7,6);

System.out.println("Sum: "+sum);

System.out.println("Difference: "+difference);

}

}

Pract 2a:- method overriding

class vehicle

{

public void run()

{

System.out.println("Vehicle is running");

}

}

class Bike extends vehicle

{

public void run()

{

System.out.println("Bike is running");

}

public static void main(String args[])

{

Bike obj=new Bike();

obj.run();

}

}

Pract 2b:- abstract class method.

abstract class Calc

{

public abstract int sqr (int a);

Public abstract int cube ( int b);

public void show()

{

System.out.println("Hello");

}

}

class Demo extends Calc

{

public int sqr(int a)

{

return a\*a;

}

public int cube(int b)

{

return b\*b\*b;

}

Public Static void main (string args[])

{

Demo d = new Demo();

System.out.println( d.sqr (5));

System.out.println(d.cube (10));

d.show();

}

}

Pract 2c:- interface.

interface area

{

void show(int a,int b);

}

class rect implements area

{

public vid show(int a,int b)

{

System.out.println("Area of rectanglr=" +(a\*b));

}

}

class circle implements area

{

public void show(int a,int b)

{

System.out.println("Area of circle=" +(3.14\*a\*a));

}

}

class pr2c

{

public static void main(String args[])

{

rect r=new rect();

r.show(10,20);

circle c=new circle();

c.show(30,10);

}

}

Practical 3a

Raise built in exception as per requirements .

public class Pr3a

{

public static void main (String [] args)

{

try

{

int result =divide(10,0);

System.out.println("result:"+result);

}

catch(ArithmeticException e)

{

System.err.println("error:division by zero");

}

}

public static int divide(int a,int b)

{

return a/b;

}

}

Practical 3b

User defined exception and raise them as per the requirement.

class CustomException extends Exception {

public CustomException(String message) {

super(message);

}

}

public class Pr3b {

public static void main(String[] args) {

try {

int age = -20;

// Check if age is negative

if (age < 0) {

throw new CustomException("Age cannot be negative.");

}

System.out.println("Age: " + age);

} catch (CustomException e) {

System.err.println("Error: " + e.getMessage());

}

}

}

Practical 9

Adapter class event handling .

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class MyAdapterClass {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("MouseAdapter Demo");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel label = new JLabel();

JPanel panel = new JPanel();

panel.setPreferredSize(new Dimension(300, 200));

panel.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

label.setText("Mouse Clicked at (" + e.getX() + "," + e.getY() + ")");

}

});

panel.add(label);

frame.add(panel);

frame.pack();

frame.setVisible(true);

});

}

}

Practical 8

ACTION EVENT WITH BUTTON CLICK

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class BtnClkDemo {

public static void main(String args[]) {

JFrame frame = new JFrame("Button Click Demo");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JButton button = new JButton("Click Me");

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(frame, "Button Clicked");

}

});

frame.getContentPane().add(button);

frame.pack();

frame.setVisible(true);

}

}

PRACTICAL 10

ANONYMOUS INNER CLASS IN EVENT HANDELING

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class amn {

public static void main(String args[]) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Anonymous Inner Class");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setPreferredSize(new Dimension(300, 200));

JButton button = new JButton("Click Here");

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JOptionPane.showMessageDialog(frame, "Button Clicked!");

}

});

panel.add(button);

frame.add(panel);

frame.pack();

frame.setVisible(true);

});

}

}

Practical 6

SWING (CHANGE COLOR)

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Random;

public class ChangeColor extends JFrame

{

private JPanel colorPanel;

private JButton changeColorButton;

public ChangeColor()

{

setTitle("Random Color Changer");

setSize(300,200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

colorPanel = new JPanel();

changeColorButton = new JButton("Change Color");

add(colorPanel,BorderLayout.CENTER);

add(changeColorButton,BorderLayout.SOUTH);

changeColorButton.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

changeColor();

}

});

}

private void changeColor()

{

Random random = new Random();

Color randomColor = new Color(random.nextInt(256),random.nextInt(256),random.nextInt(256));

colorPanel.setBackground(randomColor);

}

public static void main(String args[])

{

SwingUtilities.invokeLater(()->{

ChangeColor app = new ChangeColor();

app.setVisible(true);

});

}

}

PRACTICAL 4

MULTITHREADING

import javax.swing.\*;

import java.awt.\*;

import java.util.ArrayList;

import java.util.Random;

public class BouncingBalls extends JPanel implements Runnable {

public static final int WIDTH = 800;

public static final int HEIGHT = 600;

private static final int NUM\_BALLS = 5;

private ArrayList<Ball> balls;

public BouncingBalls() {

balls = new ArrayList<>();

Random random = new Random();

for (int i = 0; i < NUM\_BALLS; i++) {

int x = random.nextInt(WIDTH);

int y = random.nextInt(HEIGHT);

int xSpeed = random.nextInt(5) + 1;

int ySpeed = random.nextInt(5) + 1;

Color color = new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256));

balls.add(new Ball(x, y, xSpeed, ySpeed, color));

}

}

@Override

public void run() {

while (true) {

for (Ball ball : balls) {

ball.move();

}

repaint();

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

for (Ball ball : balls) {

ball.draw(g);

}

}

public static void main(String args[]) {

JFrame frame = new JFrame("Bouncing Balls");

BouncingBalls bouncingBalls = new BouncingBalls();

frame.add(bouncingBalls);

frame.setSize(WIDTH, HEIGHT);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

Thread thread = new Thread(bouncingBalls);

thread.start();

}

public void setBalls(ArrayList<Ball> balls) {

this.balls = balls;

}

}

class Ball {

private int x, y, xSpeed, ySpeed;

private Color color;

public Ball(int x, int y, int xSpeed, int ySpeed, Color color) {

this.x = x;

this.y = y;

this.xSpeed = xSpeed;

this.ySpeed = ySpeed;

this.color = color;

}

public void move() {

x += xSpeed;

y += ySpeed;

if (x < 0 || x > BouncingBalls.WIDTH) {

xSpeed = -xSpeed;

}

if (y < 0 || y > BouncingBalls.HEIGHT) {

ySpeed = -ySpeed;

}

}

public void draw(Graphics g) {

g.setColor(color);

g.fillOval(x, y, 20, 20);

}

}

Practical 7

Flow layout example

import javax.swing.\*;

import java.awt.\*;

public class DemoFlowLayout {

public static void main(String[] args) {

// Create a JFrame (main window)

JFrame frame = new JFrame("FlowLayout Example");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(300, 100);

// Create a JPanel with FlowLayout

JPanel panel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

// Create three buttons

JButton button1 = new JButton("Button 1");

JButton button2 = new JButton("Button 2");

JButton button3 = new JButton("Button 3");

// Add the buttons to the panel

panel.add(button1);

panel.add(button2);

panel.add(button3);

frame.add(panel); // Add the panel to the frame

frame.setVisible(true); // Make the JFrame visible

}

}