Practical 1a

Create class and implement a default, overloaded and copy constructor

class Demo

{

public Demo()

{

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

}

public Demo(int n1,int n2)

{

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

}

public static void main(String[] args)

{

Demo d1=new Demo();

Demo d2=new Demo(10,20);

}

}

Practica 1b

Create class and implement the concept of method overloading

class Demo

{

public void add (int n1,int n2)

{

System.out.println(n1+n2);

}

public void add (double n1,double 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(11.5,13.6);

d.add(55,65,75);

}

}

Practical 1c

Implement the comcepts of 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(5,4);

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

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

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

}

}

Practical 2a

Implement concept of inheritance and method overloading

class vehical

{

public void run()

{

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

}

}

class Bike

{

public void run()

{

System.out.print("Blike is running");

}

public static void main(String[] args)

{

Bike obj=new Bike();

obj.run();

}

}

Practical 2b

Implement the concept of abstract classes and methods

abstract class calc

{

public abstract int sqr(int a);

public abstract int cube(int b);

public void show()

{

System.out.println("Hello");

}

}

class Demo1 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)

{

Demo1 d=new Demo1();

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

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

d.show();

}

}

Practical 2c \*

Write a program to implement concept of interface

interface Area

{

void show(int a,int b);

}

class Rect implements Area

{

public void show(int a,int b)

{

System.out.println("Area of rectangle:"+(a\*b));

}

}

class Circle implements Area

{

public void show(int a,int b)

{

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

}

}

class Demo2

{

public static void main(String[] args)

{

Rect r=new Rect();

r.show(10,20);

Circle c=new Circle();

c.show(30,40);

}

}

Practical 3a

Wirte a program to raise built-in exception and raise them as per the 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.out.println("Error: Division by zero.");

}

}

public static int divide(int a, int b)

{

return(a/b);

}

}

Practical 3b

Write a program to define user defined exceptions and raise them as per the reuirement.

class CustomException extends Exception

{

public CustomException(String message)

{

super(message);

}

}

public class Pr3b

{

public static void main(String[] args)

{

try

{

int age=-20;

if (age <0)

{

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

}

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

}

catch (CustomException e)

{

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

}

}

}

Practical 6\*

Create a swing application that randomly changes color on button click

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);

});

}

}

Practaical 7

Write programe for following layouts

import javax.swing.\*;

import java.awt.\*;

public class DemoFlowLayout

{

public static void main(String[] args)

{

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

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

frame.setSize(300, 100);

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

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

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

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

 panel.add(button1);

 panel.add(button2);

 panel.add(button3);

frame.add(panel);

frame.setVisible(true);

}

}

Practical 8

Write program to demonstrate the 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 9

Demonstrate the use of adapter class in 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 10

Demonstrate the use of anonymous inner class in event handling

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MyAnonymousInner

{

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);

});

}

}