WAJP to invoke parent class instance variable using super keyword.

class Animal{

String color="white";

}

class Dog extends Animal{

String color="black";

void printColor()

{

System.out.println(color);

System.out.println(color);

}

}

class TestSuper1{

public static void main(String args[]){

Dog d=new Dog();

d.printColor();

}

}

WAJP to invoke parent class method using super keyword.

class Animal{

void eat(){

System.out.println("eating...");

}

}

class Dog extends Animal{

void eat()

{

System.out.println("eating bread...");

}

void bark(){

System.out.println("barking...");

}

void work(){

eat();

bark();

}

}

class TestSuper2{

public static void main(String args[]){

Dog d=new Dog();

d.work();

}}

WAJP to invoke parent class construct using super keyword.

class Animal

{

Animal()

{

System.out.println("animal is created");

}

}

class Dog{

Dog()

{

super();

System.out.println("dog is created");

}

}

class TestSuper3

{

public static void main(String args[])

{

Dog d=new Dog();

}

}

WAJP to invoke super keyword implicitly

class Animal{

Animal(){System.out.println("animal is created");}

}

class Dog extends Animal{

Dog(){

System.out.println("dog is created");

}

}

class TestSuper4{

public static void main(String args[]){

Dog d=new Dog();

}}

WAJP toprint the details of an employee using super keyword

class Person{

int id;

String name;

Person(int id,String name){

this.id=id;

this.name=name;

}

}

class Emp extends Person{

float salary;

Emp(int id,String name,float salary){

super(id,name);//reusing parent constructor

this.salary=salary;

}

void display(){System.out.println(id+" "+name+" "+salary);}

}

class TestSuper5{

public static void main(String[] args){

Emp e1=new Emp(1,"ankit",45000f);

e1.display();

}}

WAJP to implement interface.

interface A

{

void show();

void config();

}

class B

{

public void show()

{

System.out.println("in show");

}

public void config()

{

System.out.println("in config");

}

}

public class DemoInterface

{

public static void main(String args[])

{

//A obj;

B obj=new B();

obj.show();

obj.config();

}

}

WAJP to implement using override

import java.io.\*;

interface Vehicle {

// all are the abstract methods.

void changeGear(int a);

void speedUp(int a);

void applyBrakes(int a);

}

class Bicycle implements Vehicle{

int speed;

int gear;

// to change gear

@Override

public void changeGear(int newGear){

gear = newGear;

}

// to increase speed

@Override

public void speedUp(int increment){

speed = speed + increment; //3

}

// to decrease speed

@Override

public void applyBrakes(int decrement){

speed = speed - decrement; //3-1

}

public void printStates() {

System.out.println("speed: " + speed

+ " gear: " + gear);

}

}

class Bike implements Vehicle {

int speed;

int gear;

// to change gear

@Override

public void changeGear(int newGear){

gear = newGear;

}

// to increase speed

@Override

public void speedUp(int increment){

speed = speed + increment; //4

}

// to decrease speed

@Override

public void applyBrakes(int decrement){

speed = speed - decrement; //3

}

public void printStates() {

System.out.println("speed: " + speed

+ " gear: " + gear);

}

}

class GFG {

public static void main (String[] args) {

// creating an instance of Bicycle

// doing some operations

Bicycle bicycle = new Bicycle();

bicycle.changeGear(2);

bicycle.speedUp(3);

bicycle.applyBrakes(1);

System.out.println("Bicycle present state :");

bicycle.printStates();

// creating instance of the bike.

Bike bike = new Bike();

bike.changeGear(10);

bike.speedUp(14);

bike.applyBrakes(3);

System.out.println("Bike present state :");

bike.printStates();

}

}

4. Write a Java program on Random Access File class to perform different read and write operations.

import java.io.\*;

public class RandomAccessFileExample {

public static void main(String[] args) {

try {

// Create a RandomAccessFile object with read-write mode

RandomAccessFile file = new RandomAccessFile("data.txt", "rw");

// Write data to the file

String data1 = "Hello";

String data2 = "World";

file.writeUTF(data1);

file.writeUTF(data2);

// Move the file pointer to the beginning of the file

file.seek(0);

// Read data from the file

String readData1 = file.readUTF();

String readData2 = file.readUTF();

System.out.println("Data read from file:");

System.out.println(readData1);

System.out.println(readData2);

// Move the file pointer to the ending of the file

file.seek(file.length());

// Append new data to the file

String newData = "Java!";

file.writeUTF(newData);

// Move the file pointer to the beginning of the file

file.seek(0);

// Read data from the file again after appending

readData1 = file.readUTF();

readData2 = file.readUTF();

String readData3 = file.readUTF();

System.out.println("Data read from file after appending:");

System.out.println(readData1);

System.out.println(readData2);

System.out.println(readData3);

// Close the file

file.close();

} catch (IOException e) {

System.out.println("An error occurred: " + e.getMessage());

e.printStackTrace();

}

}

}

6. Write a program to synchronize the threads acting on the same object. [Consider the example of any reservations like railway, bus, movie ticket booking, etc.]

class TicketBooking {

private int availableTickets = 10;

public synchronized void bookTicket(String customerName, int numberOfTickets) {

System.out.println("Tickets available: " + availableTickets);

System.out.println(customerName + " is trying to book " + numberOfTickets + " ticket(s).");

if (numberOfTickets <= availableTickets) {

System.out.println("Booking " + numberOfTickets + " ticket(s) for " + customerName + "...");

availableTickets -= numberOfTickets;

System.out.println("Booking successful for " + customerName + ". Tickets remaining: " + availableTickets);

} else {

System.out.println("Not enough tickets available for " + customerName + ". Tickets remaining: " + availableTickets);

}

}

}

class CustomerThread extends Thread {

private TicketBooking bookingSystem;

private String customerName;

private int numberOfTickets;

public CustomerThread(TicketBooking bookingSystem, String customerName, int numberOfTickets) {

this.bookingSystem = bookingSystem;

this.customerName = customerName;

this.numberOfTickets = numberOfTickets;

}

@Override

public void run() {

bookingSystem.bookTicket(customerName, numberOfTickets);

}

}

public class SynchronizedBookingExample {

public static void main(String[] args) {

TicketBooking bookingSystem = new TicketBooking();

CustomerThread customer1 = new CustomerThread(bookingSystem, "Alice", 4);

CustomerThread customer2 = new CustomerThread(bookingSystem, "Bob", 5);

CustomerThread customer3 = new CustomerThread(bookingSystem, "Charlie", 2);

customer1.start();

customer2.start();

customer3.start();

}

}

8. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the +, -,\*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class calculator extends JFrame implements ActionListener

{

private String s="";

private int i;

private int result;

private JButton x0;

private JButton x1;

private JButton x2;

private JButton x3;

private JButton x4;

private JButton x5;

private JButton x6;

private JButton x7;

private JButton x8;

private JButton x9;

private JButton x11;

private JButton x12;

private JButton x13;

private JButton x14;

private JButton x15;

private JButton x16;

private JButton xc;

private JTextField xt;

public calculator()

{

setLayout(new FlowLayout());

xt=new JTextField(10);

add(xt);

x1=new JButton("1");

add(x1);

x2=new JButton("2");

add(x2);

x3=new JButton("3");

add(x3);

x4=new JButton("4");

add(x4);

x5=new JButton("5");

add(x5);

x6=new JButton("6");

add(x6);

x7=new JButton("7");

add(x7);

x8=new JButton("8");

add(x8);

x9=new JButton("9");

add(x9);

x0=new JButton("0");

add(x0);

x11=new JButton("+");

add(x11);

x12=new JButton("-");

add(x12);

x13=new JButton("\*");

add(x13);

x14=new JButton("/");

add(x14);

x15=new JButton("%");

add(x15);

x16=new JButton("=");

add(x16);

xc=new JButton("C");

add(xc);

x1.addActionListener(this);

x2.addActionListener(this);

x3.addActionListener(this);

x4.addActionListener(this);

x5.addActionListener(this);

x6.addActionListener(this);

x7.addActionListener(this);

x8.addActionListener(this);

x9.addActionListener(this);

x0.addActionListener(this);

x11.addActionListener(this);

x12.addActionListener(this);

x13.addActionListener(this);

x14.addActionListener(this);

x15.addActionListener(this);

x16.addActionListener(this);

xc.addActionListener(this);

}

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==x1)

{

s=s+"1";

xt.setText(s);

}

if(e.getSource()==x2)

{

s=s+"2";

xt.setText(s);

}

if(e.getSource()==x3)

{

s=s+"3";

xt.setText(s);

}

if(e.getSource()==x4)

{

s=s+"4";

xt.setText(s);

}

if(e.getSource()==x5)

{

s=s+"5";

xt.setText(s);

}

if(e.getSource()==x6)

{

s=s+"6";

xt.setText(s);

}

if(e.getSource()==x7)

{

s=s+"7";

xt.setText(s);

}

if(e.getSource()==x8)

{

s=s+"8";

xt.setText(s);

}

if(e.getSource()==x9)

{

s=s+"9";

xt.setText(s);

}

if(e.getSource()==x0)

{

s=s+"0";

xt.setText(s);

}

if(e.getSource()==xc)

{

s="";

xt.setText(s);

}

if(e.getSource()==x11)

{

int a=Integer.parseInt(xt.getText());

result=a;

i=1;

s="";

xt.setText(s);

}

if(e.getSource()==x12)

{

int a=Integer.parseInt(xt.getText());

result=a;

i=2;

s="";

xt.setText(s);

}

if(e.getSource()==x13)

{

int a=Integer.parseInt(xt.getText());

result=a;

i=3;

s="";

xt.setText(s);

}

if(e.getSource()==x14)

{

int a=Integer.parseInt(xt.getText());

result=a;

i=4;

s="";

xt.setText(s);

}

if(e.getSource()==x15)

{

int a=Integer.parseInt(xt.getText());

result=a;

i=5;

s="";

xt.setText(s);

}

if(e.getSource()==x16)

{

int b=Integer.parseInt(xt.getText());

if(i==1)

result=result+b;

if(i==2)

result=result-b;

if(i==3)

result=result\*b;

if(i==4)

result=result/b;

if(i==5)

result=result%b;

xt.setText(""+result);

}

}

public static void main(String args[])

{

calculator x=new calculator();

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

x.setSize(500,500);

x.setVisible(true);

}

}

9. Write a Java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired. [Use Adapter classes]

import java.awt.event.\*;

import javax.swing.\*;

import java.awt.\*;

class experiment9 extends Frame implements MouseListener,MouseMotionListener

{

private int x;

private int y;

private String s="hello";

experiment9()

{

addMouseListener(this);

addMouseMotionListener(this);

}

public void mousePressed(MouseEvent me)

{

x=200;

y=200;

s="mouse pressed";

repaint();

}

public void mouseMoved(MouseEvent me)

{

x=200;

y=200;

s="mouse Moved";

repaint();

}

public void mouseEntered(MouseEvent me)

{

x=200;

y=200;

s="mouse Entered";

repaint();

}

public void mouseExited(MouseEvent me)

{

x=200;

y=200;

s="mouse Exited";

repaint();

}

public void mouseClicked(MouseEvent me)

{

x=200;

y=200;

s="mouse Clicked";

repaint();

System.exit(0);

}

public void mouseDragged(MouseEvent me)

{

x=200;

y=200;

s="mouse Dragged";

repaint();

}

public void mouseReleased(MouseEvent me)

{

x=200;

y=200;

s="mouse Released";

repaint();

}

public void paint(Graphics g)

{

g.drawString(s,x,y);

}

public static void main(String args[])

{

experiment9 x=new experiment9();

// x.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

x.setSize(500,500);

x.setVisible(true);

}

}