1)(P1a)Write a program to create a class and implement a default, overloaded and copy Constructor.

Ans:

class P1a

{

private int data;

//Default Constructor

public P1a()

{

this.data = 10;

}

// Overloaded Constructor

public P1a(int data)

{

this.data = data;

}

// Copy Constructor

public P1a(P1a value)

{

this.data = value.data;

}

// Getter method to retrieve the value

public int GetData()

{

return data;

}

public static void main(String[] args)

{

// Using the default constructor

P1a obj1 = new P1a();

System.out.println("\nDefault Constructor - Data for obj1 : " + obj1.GetData());

// Using the overloaded constructor

P1a obj2 = new P1a(100);

System.out.println("\nOverloaded Constructor - Data for obj2 : " + obj2.GetData());

// Using the copy constructor

P1a obj3 = new P1a(obj2);

System.out.println("\nCopy Constructor - Data for obj3 : " + obj3.GetData());

}//main()

}//class P1a

2) (P1b)Write a program to create a class and implement the concepts of Method Overloading.

Ans:

class P1b

{

void Show()

{

System.out.println("Show() method - 1");

}//Show()

void Show(int a)

{

System.out.println("Show() method - 2");

}//Show()

public static void main(String [] args)

{

P1b m=new P1b();

m.Show(100);

}//main()

}//class P1b

3)(P1c) Write a program to create a class and implement the concepts of Static methods.

Ans:

import java.util.\*;

class P1c

{

//Static method

public static int Addition(int x, int y)

{

return x + y;

}//Addition

//Static method

public static int Subtraction(int x, int y)

{

return x - y;

}//Subtraction

//Non-Static method

public int Multiplication(int x, int y)

{

return x \* y;

}//Multiplication

//Non-Static method

public int Divison(int x, int y)

{

return x / y;

}//Divison

public static void main(String [] args)

{

int add, sub, mul, div, num1, num2;

Scanner sc=new Scanner(System.in);

System.out.println("------Data for Static and Non-Static Methods------");

System.out.print("Enter num1 : ");

num1=sc.nextInt();

System.out.print("\nEnter num2 : ");

num2=sc.nextInt();

System.out.println("\n------Calling a Static Methods------");

//Calling a static method

add=Addition(num1,num2);

System.out.println("Addition : "+ add);

//Calling a static method

sub=Subtraction(num1,num2);

System.out.println("\nSubtraction : "+ sub);

//Object creation to call non-static methods

P1c obj=new P1c();

System.out.println("\n------Calling a Non-Static Methods------");

//Calling a non-static method

mul=obj.Multiplication(num1,num2);

System.out.println("Multiplication : "+ mul);

//Calling a non-static method

div=obj.Divison(num1,num2);

System.out.println("\nDivison : "+ div);

}//main()

}//class P1c

4)(P2a)Write a program to implement the concepts of Inheritance and Method overriding.

Ans:

class A

{

void Show()

{

System.out.println("Show() - 1");

}//Show()

}

class B extends A

{

void Show()

{

//super.Show(); //When you want to call parent class method

System.out.println("Show() - 2");

}//Show()

}

class P2a

{

public static void main(String [] args)

{

B obj=new B();

obj.Show();

}//main()

}//class P2a

5)(Vehical.java) Write a program to implement the concepts of Abstract classes and methods. (File Name - Vehicle.java) Compile - javac Vehicle.java

Execute - java Bike

Ans:

Save the file - Vehicle.java

Compile - javac Vehicle.java

Execute - java Bike

\*/

//Abstraction

abstract class Vehicle //Vehicle is Parent Class

{

abstract void Start();

}//class Vehicle

//Inheritance

class Car extends Vehicle //Car is Child Class

{

void Start()

{

System.out.println("Car starts with a KEY");

}//Start()

}//class Car

class Bike extends Vehicle //Bike is Child Class

{

void Start()

{

System.out.println("Bike starts with a KICK");

}//Start()

public static void main(String [] args)

{

Car c=new Car();

c.Start();

Bike b=new Bike();

b.Start();

}//main()

}//class Bike

6)(P2c) Write a program to implement the concept of interfaces.

Ans:

Interfaces

- An interface in Java is a blueprint of a class. It has static constants and abstract methods.

- The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

\*/

//Interface

interface Bank

{

float ROI();

}//interface Bank

interface SavingsAccount

{

void Account();

}//interface SavingsAccount

class LenaBank implements Bank, SavingsAccount

{

public float ROI()

{

return 8.2F;

}//ROI()

public void Account()

{

System.out.println("\nSavings Account");

}//Account()

}//class LenaBank

public class P2c

{

public static void main(String [] args)

{

Bank b1=new LenaBank();

System.out.println("LenaBank - "+b1.ROI());

SavingsAccount a1=new LenaBank();

a1.Account();

}//main()

}//class P2c

7)(Onedimensional.java)Write a programs to demonstrate the concept of one dimensional array

Ans:

class Onedimensional

{

public static void main(String[] args)

{

int[] a = new int[3];

a[0]=10;

a[1]=20;

a[2]=30;

System.out.println("One Dimensional array: ");

System.out.println(a[0]);

System.out.println(a[1]);

System.out.println(a[2]);

}

}

8)(P3b)Write a program to define user defined exceptions and raise them as per the requirements.

Ans:

import java.util.\*;

class NumberNotInRange extends Exception

{

NumberNotInRange(String str)

{

//Call the constructor of parent class Exception

super(str);

}//NumberNotInRange

}//class NumberNotInRange

class P3b

{

public static void main(String [] args)

{

int num;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number:");

num=sc.nextInt();

try

{

if(num<=100 && num>=1)

{

System.out.println("Your inputs are correct");

}//if

else

{

throw new NumberNotInRange("The number should be in the range of 1 to 100");

}//else

}//try

catch(NumberNotInRange e)

{

System.out.println(e.getMessage());

}//catch

}//main()

}//class P3b

9)(Company.java) Write a program to demonstrate the concept of data hiding and encapsulation .

Ans:

class Employee {

private int empId;

public void setEmpId(int eid) {

empId = eid;

}

public int getEmpId() {

return empId;

}

}

class Company {

public static void main(String[] args) {

Employee e = new Employee();

e.setEmpId(101);

System.out.println(e.getEmpId());

}

}

10) Write a program to demonstrate arithmetic exceptions .

Ans:

//10)(ArthExcep.java) Write a program to demonstrate arithmetic exceptions .

import java.util.Scanner;

class ArthExcep

{

public static void main(String[] args)

{

int a,b,c;

Scanner sc=new Scanner(System.in);

System.out.println("Enter a : ");

a=sc.nextInt();

System.out.println("Enter b : ");

b=sc.nextInt();

System.out.println("\na= "+a);

System.out.println("\nb= "+b);

try

{

c = a/b;

System.out.println("\nc= "+c);

}

catch(ArithmeticException e)

{

System.out.println("\nPlease enter a non zero number");

}

}

}

11)(Multicatch.java)Write a program to demonstrate multiple catch blocks along with finally

Ans:

public class Multicatch

{

public static void main(String[] args)

{

try

{

int[] numbers = {1,2,3};

int result = numbers[4];

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

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("ArrayIndexOutOfBoundsException "+e.getMessage());

}

catch(ArithmeticException e)

{

System.out.println(" Caught ArithmeticException "+e.getMessage());

}

finally

{

System.out.println("finally block executed");

}

}

}

12)(T1.java)Write a program to design a thread by extending a thread class

Ans:

class MyThread extends Thread

{

public void run()

{

for(int i=0;i<=5;i++)

{

System.out.print("MyThread class");

}

}

}

class T1

{

public static void main(String[] args)

{

MyThread t = new MyThread();

t.start();

for(int i=0;i<=5;i++)

{

System.out.println("T1 class");

}

}

}

13)(T2.java)Write a program to implement the runnable interface and create a thread class

Ans:

class MyRunnable implements Runnable

{

public void run()

{

for(int i=0;i<5;i++)

{

System.out.println("My Runnable Class");

}

}

}

class T2

{

public static void main(String[] args)

{

MyRunnable r = new MyRunnable();

Thread t = new Thread(r);

t.start();

for(int i=0;i<5;i++)

{

System.out.println("T2 class");

}

}

}

14)(P1)AWT-by extending frame class

Ans:

import java.awt.\*;

class P1 extends Frame

{

P1()

{

//Button class

Button b=new Button("Click Me");

//Setting button position

/\*setBounds(int x, int y, int width, int height) - Specifies the size of the frame and the location of the upper left corner x axis to the right and y axis to the bottom\*/

b.setBounds(30,100,80,30);

//Adding button into frame

add(b);

//Frame size - 300 width and 300 height

setSize(300,300);

//No layout manager

setLayout(null);

//Now frame will be visible, by default not visible

setVisible(true);

}

public static void main(String args[])

{

P1 p=new P1();

}

}

15)(P5)AWT-text area

Ans:

import java.awt.\*;

class P5

{

public static void main(String args[])

{

Frame f= new Frame();

TextArea t=new TextArea();

t.setBounds(10,30,300,200);

f.add(t);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

}

16)(P3)AWT-labels and buttons

Ans:

import java.awt.\*;

class P3

{

public static void main(String args[])

{

Frame f= new Frame("Labels and Buttons");

//Labels

Label l1,l2;

l1=new Label("Label 1");

l1.setBounds(50,100, 100,30);

l2=new Label("Label 2");

l2.setBounds(50,150, 100,30);

f.add(l1);

f.add(l2);

//Buttons

Button b1, b2;

b1=new Button("Button 1");

b1.setBounds(160,100, 100,30);

b2=new Button("Button 2");

b2.setBounds(160,150, 100,30);

f.add(b1);

f.add(b2);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

}

17)(P2) AWT-creating object of frame class

Ans:

import java.awt.\*;

class P2

{

P2()

{

//Frame class

Frame f=new Frame();

Button b=new Button("Click Me");

b.setBounds(30,50,80,30);

f.add(b);

f.setSize(300,300);

f.setLayout(null);

f.setVisible(true);

}

public static void main(String args[])

{

P2 p=new P2();

}

}

18)(P11)AWT-java flow layout

Ans:

/\*The FlowLayout is used to arrange the components in a line, one after another (in a flow).

It is the default layout of applet or panel.

Fields of FlowLayout class

public static final int LEFT

public static final int RIGHT

public static final int CENTER\*/

import java.awt.\*;

class P11

{

public static void main(String[] args)

{

Frame f=new Frame();

Button b1=new Button("1");

Button b2=new Button("2");

Button b3=new Button("3");

Button b4=new Button("4");

Button b5=new Button("5");

f.add(b1);

f.add(b2);

f.add(b3);

f.add(b4);

f.add(b5);

//Setting flow layout of right alignment

f.setLayout(new FlowLayout(FlowLayout.RIGHT));

f.setSize(300,300);

f.setVisible(true);

}

}

19)(P10)AWT-grid layout

Ans:

//Java GridLayout

/\*The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.\*/

import java.awt.\*;

class P10

{

public static void main(String[] args)

{

Frame f=new Frame();

Button b1=new Button("1");

Button b2=new Button("2");

Button b3=new Button("3");

Button b4=new Button("4");

Button b5=new Button("5");

Button b6=new Button("6");

Button b7=new Button("7");

Button b8=new Button("8");

Button b9=new Button("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);

//Setting grid layout of 3 rows and 3 columns

f.setLayout(new GridLayout(3,3));

f.setSize(300,300);

f.setVisible(true);

}

}

20)(P9)Write programs for the following

layouts : Border Layout

Ans:

//Java BorderLayout

/\*The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. The BorderLayout provides five constants for each region:

public static final int NORTH

public static final int SOUTH

public static final int EAST

public static final int WEST

public static final int CENTER

\*/

import java.awt.\*;

class P9

{

public static void main(String[] args)

{

Frame f=new Frame();

Button b1=new Button("NORTH");

Button b2=new Button("SOUTH");

Button b3=new Button("EAST");

Button b4=new Button("WEST");

Button b5=new Button("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(500,500);

f.setVisible(true);

}

}