**WEEK – 1**

**Exp. No: 1(a) Date:\_\_\_\_\_\_\_\_\_\_**

**Question:**

# Write a java program to find prime numbers between 1 to n.

# AIM:

To write a java program to find prime numbers between 1 to n.

**DESCRIPTION:**

🡪We are finding the prime numbers within the limit.

**🡪** Read the “n” value using scanner object  obj.nextInt()and store it in the variable n.

🡪The for loop iterates from i=2 to i=given number. then count assigned to 0, the inner loop finds the divisors of each i value, count

value represents no.of divisors. If count=2, then that number is a prime number.

**PROGRAM:**

import java.util.Scanner;

class PrimeNumbers{

public static void main(String[] args){

try{

System.*out*.println("\*\*\*\*\* PRIME NUMBERS \*\*\*\*\*");

Scanner obj = new Scanner(System.*in*);

System.*out*.print("\nEnter n Value:");

long n = obj.nextInt();

for (long i = 2; i <= n; i++){

boolean isprime = *isNumPrime*(i);

if (isprime){

System.*out*.print(i + " ");

}

}

}

catch (Exception e){

e.printStackTrace();

}

}

public static boolean isNumPrime(long number){

boolean result = true;

for (long i = 2; i <= number / 2; i++){

if ((number % i) != 0){

result = true;

}

else{

result = false;

break;

}

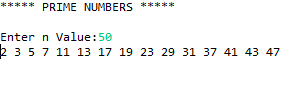
}

return result;

}

}

**OUTPUT:**



**RESULT:**

Java program to find prime numbers between 1 to n is successfully compiled and Executed.

**Exp. No: 1(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

# Write a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0. Read in a, b, c and use the quadratic formula.

**AIM:**

# To write a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0.

**DESCRIPTION:**

The standard form of a quadratic equation is: ax2 + bx + c = 0

Here, a, b, and c are real numbers and a can't be equal to 0.

We can calculate the root of a quadratic by using the formula:

x = (-b ± √(b2-4ac)) / (2a)

The ± sign indicates that there will be two roots:

root1 = (-b + √(b2-4ac)) / (2a)

root1 = (-b - √(b2-4ac)) / (2a)

The term b2-4ac is known as the determinant of a quadratic equation. It specifies the nature of roots.

That is,

if determinant > 0, roots are real and different

if determinant == 0, roots are real and equal

if determinant < 0, roots are complex complex and different.

**PROGRAM:**

import java.util.\*;

class QudraticRoots{

public static void main(String args[]){

int a,b,c,d,f=0;

Scanner sc=new Scanner(System.*in*);

System.*out*.println("\nEnter the values of a ,b ,c : ");

a=sc.nextInt();

b=sc.nextInt();

c=sc.nextInt();

d=(b\*b)-(4\*a\*c);

if(d==0){

System.*out*.println("Roots are real and Equal");

f=1;

}

else if(d>0){

System.*out*.println("Roots are real and UnEqual");

f=1;

}

else

System.*out*.println("Roots are imaginary");

if(f==1){

float r1=(float)(-b+Math.*sqrt*(d))/(2\*a);

float r2=(float)(-b-Math.*sqrt*(d))/(2\*a);

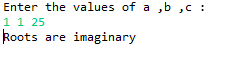
System.*out*.println("Roots are : "+r1+" ,"+r2);

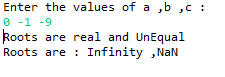
}

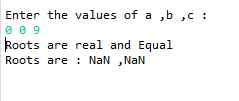
}

}

**OUTPUT:**







**RESULT:**

Java program that prints all real solutions to the quadratic equation ax2+bx+c=0 is successfully compiled and executed.

**Exp. No: 1(c) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

# Develop a Java application to generate Electricity bill. Create a class with the

**Following members: Consumer no., consumer name, previous month reading, current month reading, type of EB connection(i.e domestic or commercial). Commute the bill amount using the following tariff. If the type of the EB connection is domestic, calculate the amount to be paid as follows:**

* **First 100 units - Rs. 1 perunit**
* **101-200 units - Rs. 2.50 perunit**
* **201 -500 units - Rs. 4 perunit**
* **> 501 units - Rs. 6 perunit**

**If the type of the EB connection is commercial, calculate the amount to be paid as follows:**

* **First 100 units - Rs. 2 perunit**
* **101-200 units - Rs. 4.50 perunit**
* **201 -500 units - Rs. 6 perunit**
* **> 501 units - Rs. 7 perunit**

**AIM:**

# To develop a Java application to generate Electricity bill.

**DESCRIPTION:**

To compute the electricity bill :

If type of EB connection is Domestic:

* First 100 units - Rs. 1 per unit
* 101-200 units - Rs. 2.50 per unit
* 201 -500 units - Rs. 4 per unit
* > 501 units - Rs. 6 per unit

If the type of the EB connection is commercial:

* First 100 units - Rs. 2 perunit
* 101-200 units - Rs. 4.50 perunit
* 201 -500 units - Rs. 6 perunit
* > 501 units - Rs. 7 perunit

**PROGRAM:**

import java.util.Scanner;

class Consumer\_details

{

int ConsumerNo;

String ConsumerName;

int PrevReading;

int CurrReading;

String EBConn;

double Bill;

void input\_data()

{

Scanner sc = new Scanner(System.*in*);

System.*out*.println(" Enter Consumer Number: ");

ConsumerNo = sc.nextInt();

System.*out*.println(" Enter Consumer Name: ");

ConsumerName = sc.next();

System.*out*.println(" Enter Previous Units: ");

PrevReading = sc.nextInt();

System.*out*.println("Enter Current Units consumed:");

CurrReading = sc.nextInt();

System.*out*.println("Enter the types of EB Connection(domestic or commercial)");

EBConn = sc.next();

}

double calculate\_bill()

{

int choice;

if(EBConn.equals("domestic"))

choice=1;

else

choice=2;

switch(choice)

{

case 1:

if(CurrReading>=0 && CurrReading<=100)

Bill=CurrReading\*1;

else if(CurrReading>100 && CurrReading <= 200)

Bill=(100\*1)+((CurrReading-100)\*2.50);

else if(CurrReading>200 && CurrReading <= 500)

Bill=(100\*1)+(100\*2.50)+((CurrReading-200)\*4);

else

Bill=(100\*1)+(100\*2.50)+(300\*4)+((CurrReading-500)\*6);

break;

case 2:

if(CurrReading>=0 && CurrReading<=100)

Bill=CurrReading\*2;

else if(CurrReading>100 && CurrReading <= 200)

Bill=(100\*1)+((CurrReading-100)\*4.50);

else if(CurrReading>200 && CurrReading <= 500)

Bill=(100\*1)+(100\*2.50)+((CurrReading-200)\*6);

else

Bill=(100\*1)+(100\*2.50)+(300\*4)+((CurrReading-500)\*7);

break;

}

return Bill;

}

void display()

{

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

System.*out*.println("ELCTRICITY BILL");

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

System.*out*.println("Consumer Number: "+ConsumerNo);

System.*out*.println("Consumer Name: "+ConsumerName);

System.*out*.println("Consumer Previous Units: "+PrevReading);

System.*out*.println("Consumer Current Units: "+CurrReading);

System.*out*.println("Type of EBConnection: "+EBConn);

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

System.*out*.println("Total Amount(Rs.): "+Bill);

}}

class ElectricBill

{

public static void main (String[] args)

{

Consumer\_details b=new Consumer\_details();

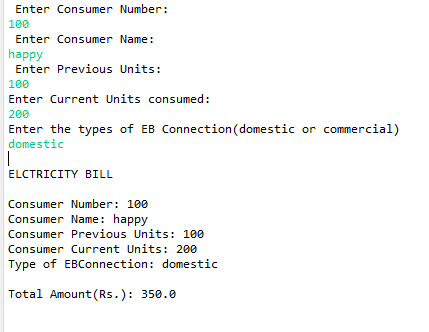
b.input\_data();

b.calculate\_bill();

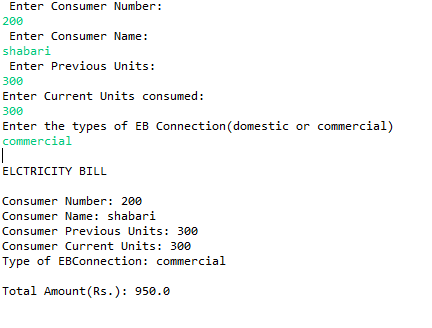
b.display();

}}

**OUTPUT 1:**



**OUTPUT 2:**



**RESULT:**

# Java application to generate Electricity bill is successfully developed.

**Exp. No: 1(d) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

# Write a Java program to multiply two given matrices.

**AIM:**

# To write a Java program to multiply two given matrices.

**DESCRIPTION:**

For matrix multiplication to take place, the number of columns of first matrix must be equal to the number of rows of second matrix. In our example, i.e.

c1 = r2

Also, the final product matrix is of size r1 x c2, i.e.

product[r1][c2]

**PROGRAM:**

class MatrixMultiplication {

static void printMatrix(int M[][],

int rowSize,

int colSize)

{

for (int i = 0; i < rowSize; i++) {

for (int j = 0; j < colSize; j++)

System.*out*.print(M[i][j] + " ");

System.*out*.println();

}

}

static void multiplyMatrix(

int row1, int col1, int A[][],

int row2, int col2, int B[][])

{

int i, j, k;

System.*out*.println("\nMatrix A:");

*printMatrix*(A, row1, col1);

System.*out*.println("\nMatrix B:");

*printMatrix*(B, row2, col2);

if (row2 != col1) {

System.*out*.println(

"\nMultiplication Not Possible");

return;

}

int C[][] = new int[row1][col2];

for (i = 0; i < row1; i++) {

for (j = 0; j < col2; j++) {

for (k = 0; k < row2; k++)

C[i][j] += A[i][k] \* B[k][j];

}

}

System.*out*.println("\nResultant Matrix:");

*printMatrix*(C, row1, col2);

}

public static void main(String[] args)

{

int row1 = 4, col1 = 3, row2 = 3, col2 = 4;

int A[][] = { { 3,3,3 },{ 8, 8, 8 },{ 3, 3, 3 },{ 8, 8, 8 } };

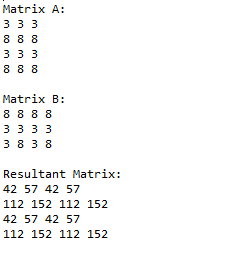
int B[][] = { { 8, 8, 8,8 },{ 3, 3, 3, 3 },{ 3, 8, 3, 8 } };

*multiplyMatrix*(row1, col1, A,row2, col2, B);

}

}

**OUTPUT:**

****

**RESULT:**

# Java program to multiply two given matrices is successfully writtened and executed.

**WEEK – 2**

**Exp. No: 2(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

* 1. **Write Java program on use of inheritance, preventing inheritance using final, abstract classes**.

**AIM:**

* 1. To write Java program on use of inheritance, preventing inheritance using final, abstract classes.

**DESCRIPTION:**

* **Inheritance:**

The class which inherits the properties of other is known as subclass and the class whose properties are inherited is known as super class.

**extends** is the keyword used to inherit the properties of a class.

* **Preventing Inheritance:**

When a class is declared as final then it cannot be subclassed i.e. no any other class can extend it.

* **Abstract classes:**

A class which is declared with the abstract keyword is known as an abstract class in [Java](https://www.javatpoint.com/java-tutorial).

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

**Program on Inheritance :**

class Parent

{

public void p1()

{

System.*out*.println("I am parent class method");

}

}

public class Child extends Parent {

public void c1()

{

System.*out*.println("I am a Child class method");

}

public static void main(String[] args)

{

Child obj = new Child();

obj.c1();

obj.p1();

}

}

**OUTPUT:**

****

**Program on preventing Inheritance :**

final class Parent{

void show()

{

System.out.println("I am parent 1");

}

}

class Child extends Parent{

void show()

{

System.out.println("I am child of parent 1");

}

public static void main(String[] args) {

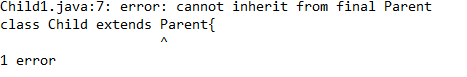
Child obj = new Child();

obj.show();

}

}

**OUTPUT:**

****

**Program on abstract classes:**

abstract class Bike{

abstract void school();

}

class Children extends Bike{

void school(){

System.*out*.println("Going to school");

}

public static void main(String args[]){

Bike obj = new Children();

obj.school();

}

}

**OUTPUT:**

****

**RESULT:**

Java program on use of inheritance, preventing inheritance using final, abstract classes is successfully compiled and executed.

**Exp. No: 2(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write Java program on dynamic binding, differentiating method overloading and overriding.**

**AIM:**

To write Java program on dynamic binding, differentiating method overloading and overriding.

**DESCRIPTION:**

* **Dynamic binding**

When type of the object is determined at run-time, it is known as dynamic binding.

* **Method Overloading**

In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading.

* **Method Overriding**

The same method is defined in both the superclass and the subclass, then the method of the subclass class overrides the method of the superclass. This is known as method overriding.

**Program on Dynamic Binding:**

class Newmobile{

void fingerPrint()

{System.*out*.println("I don't have fingerprint");}

}

class Mobile extends Newmobile{

void fingerPrint(){System.*out*.println("I have Finger print Sensor");}

public static void main(String args[]){

Newmobile a=new Mobile();

a.fingerPrint();

} }

**OUTPUT:**

****

**Program on differentiating method overloading:**

class Adder{

static int add(int a,int b){

return a+b;

}

static int add(int a,int b,int c){

return a+b+c;

}

static int add(int a,int b,int c,int d){

return a+b+c+d;

}

}

class Overloading{

public static void main(String[] args){

System.*out*.println("Method OverLoading");

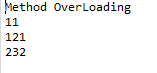
System.*out*.println(Adder.*add*(3,8));

System.*out*.println(Adder.*add*(11,10,100));

System.*out*.println(Adder.*add*(33,88,111));

}}

**OUTPUT:**

****

**Program on method overriding**

class Car{

void run(){

System.*out*.println("Car is running");

}

}

class Overriding extends Car{

void run(){

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

}

public static void main(String args[]){

Overriding obj = new Overriding();

obj.run();

}

}

**OUTPUT:**

****

**RESULT:**

Java program on dynamic binding, differentiating method overloading and overriding is executed successfully.

**Exp. No: 2(c) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Develop a java application to implement currency converter (Dollar to INR, EURO to INR,Yen) using Interfaces**

**AIM:**

To develop a java application to implement currency converter (Dollar to INR, EURO to INR,Yen) using Interfaces

**DESCRIPTION:**

Create an interface to develop java application to implement currency coversions.Inside an interface declare methods DollarToInr(),

EuroTOInr(),EuroToYen. Implement this Interface in a class and write suitable calculations to implement conversions.

**PROGRAM:**

import java.util.Scanner;

interface Currency{

void DollarToInr();

void EuroToInr();

void EuroToYen();

}

class Conversions implements Currency{

Scanner sc = new Scanner(System.*in*);

double inr;

double Dollar;

double euro;

double yen;

public void DollarToInr(){

System.*out*.println("Enter DOLLARS to convert into INR");

Dollar=sc.nextDouble();

inr=Dollar\*73.162;

System.*out*.println("DOllar "+Dollar+" equal to INR = "+inr);

}

public void EuroToInr(){

System.*out*.println("Enter Euro to convert into INR");

euro=sc.nextDouble();

inr=euro\*88.65;

System.*out*.println("EURO "+euro+" equal to INR = "+inr);

}

public void EuroToYen(){

System.*out*.println("Enter Euro to convert into YEN");

euro=sc.nextDouble();

yen=euro\*126.144;

System.*out*.println("euro "+euro+" equal to YEN = "+yen);

}

}

class Currency\_converter{

public static void main(String[] args) {

Scanner s = new Scanner(System.*in*);

int choice,ch;

Conversions con = new Conversions();

do{

System.*out*.println("CURRENCY CONVERTER");

System.*out*.println("--------------------------\n");

System.*out*.println("1.Dollar to INR ");

System.*out*.println("2.Euro to INR ");

System.*out*.println("3.Euro to YEN ");

System.*out*.println("Enter ur choice");

choice=s.nextInt();

switch(choice){

case 1: con.DollarToInr();

break;

case 2: con.EuroToInr();

break;

case 3: con.EuroToYen();

}

System.*out*.println("Enter 0 to quit and 1 to continue ");

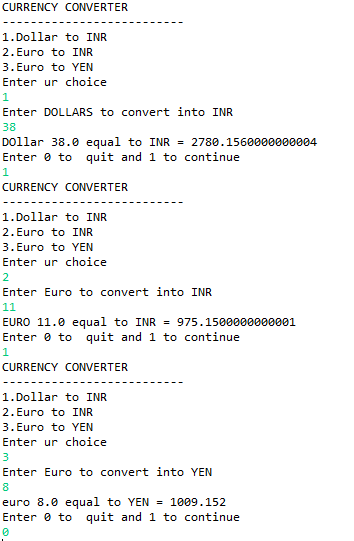
ch=s.nextInt();

}while(ch==1);

}

}

**OUTPUT:**

****

**RESULT:**

Java application to implement currency converter (Dollar to INR, EURO to INR,Yen) using Interfaces is successfully developed.

**WEEK – 3**

**Exp. No: 3(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write Java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it’s not a duplicate of any number already read display the complete set of unique values input after the user enters each new value.**

**AIM:**

To write Java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it’s not a duplicate of any number already read display the complete set of unique values input after the user enters each new value.

**DESCRIPTION:**

Use a one-dimensional array to solve the following problem:Write an application that inputs five numbers, each of which is between 10 and 100, inclusive. As each number is read, display it only if it is not a duplicate of a number already read.

**PROGRAM:**

import java.util.Scanner;

public class Duplicate{

public static void main(String[ ] args){

int a[]={0,0,0,0,0},t,i,j,s=0,r=0;

Scanner z=new Scanner(System.*in*);

System.*out*.println("Enter 5 unique values between 10 & 100 ");

for(j=0;j<5;j++){

t=z.nextInt();

if(t>=10&&t<=100){

for(i=0;i<r;i++){

if(a[i]==t)

s++;

}

if(s>0){

System.*out*.println("Duplicate value found retry");

s--;

j--;

continue;

}

else{

a[j]=t;

r++;

} }

else {

System.*out*.println("value must be in between 10 & 100");

j--;

} }

System.*out*.print("The five unique values are ");

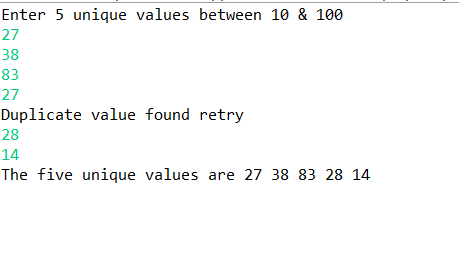
for(i=0;i<5;i++){

System.*out*.print(a[i]+" ");

} }

}

**OUTPUT:**

****

**RESULT:**

Java program that inputs 5 numbers, each between 10 and 100 inclusive is compiled and executed.

**Exp. No: 3(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.**

**AIM:**

To write a program to create an abstract class named Shape that contains two integers and an empty method named print Area().Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.

**DESCRIPTION:**

Create an abstract class named Shape that contains two integers named a,b and an empty method named print Area().

Create three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape.

Each one of the classes contains only the method print Area () inside the print\_Area() method

* To calculate Area of rectangle

area\_rect=a\*b

🡪 To calculate the Area of triangle

area\_tri=(int) (0.5\*a\*b);

🡪 To calculate the Area of circle

area\_circle=(int) (3.14\*a\*a);

**PROGRAM:**

abstract class shape{

int a=11,b=3;

abstract public void print\_area();

}

class rectangle extends shape {

public int area\_rect;

public void print\_area(){

area\_rect=a\*b;

System.*out*.println("The area of rectangle is:"+area\_rect);

}}

class triangle extends shape {

int area\_tri;

public void print\_area() {

area\_tri=(int) (0.5\*a\*b);

System.*out*.println("The area of triangle is:"+area\_tri);

} }

class circle extends shape {

int area\_circle;

public void print\_area() {

area\_circle=(int) (3.14\*a\*a);

System.*out*.println("The area of circle is:"+area\_circle);

}}

public class Geometricalshapes {

public static void main(String[] args) {

rectangle r=new rectangle();

r.print\_area();

triangle t=new triangle();

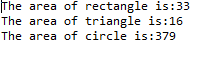
t.print\_area();

circle r1=new circle();

r1.print\_area();

} }

**OUTPUT:**

****

**RESULT:**

Successfully printed the area of the given shapes.

**Exp. No: 3(c) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a Java program to read the time intervals (HH:MM) and to compare system time if the system Time between your time intervals print correct time and exit else try again to repute the same thing. By using String Toknizer class.**

**AIM:**

To write a Java program to read the time intervals (HH:MM) and to compare system time if the system Time between your time intervals print correct time and exit else try again to repute the same thing. By using String Tokenizer class.

**DESCRIPTION:**

A StringTokenizer object internally maintains a current position within the string to be tokenized. Some operations advance this current position past the characters processed.  
A token is returned by taking a substring of the string that was used to create the StringTokenizer object.

**PROGRAM:**

import java.util.\*;

public class Tokens{

public static void main(String args[]){

String mydelim = " : ";

String mystr = "Software : Developer : Good : At : Programming";

StringTokenizer geeks3 = new StringTokenizer(mystr, mydelim);

int count = geeks3.countTokens();

System.*out*.println("Number of tokens : " + count + "\n");

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

System.*out*.println("token at [" + i + "] : " + geeks3.nextToken());

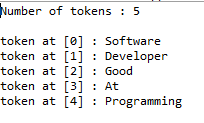
while (geeks3.hasMoreTokens())

System.*out*.println(geeks3.nextToken());

}

}

**OUTPUT:**



**RESULT:**

To write a Java program to read the time intervals (HH:MM) and to compare system time if the system Time between your time intervals print correct time and exit else try again to repute the same thing. By using String Tokenizer class is complied and executed successfully.

**WEEK-4**

**Exp. No: 4(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

# Write a Java program to implement user defined exception handling

**AIM:**

# To write a Java program to implement user defined exception handling.

**DESCRIPTION:**

Creating our own Exception that is known as user-defined exception. Java custom exceptions are used to customize the exception according to user need.

We pass the string to the constructor of the super class- Exception which is obtained using “getMessage()” function on the object created.

**PROGRAM:**

class MyException extends Exception{

public MyException(String s){

super(s);

}

}

public class User\_exceptions{

public static void main(String args[]){

try{

throw new MyException("msg exception");

}

catch (MyException ex){

System.*out*.println("Caught");

System.*out*.println(ex.getMessage());

}

}

}

**OUTPUT:**

****

**RESULT:**

To write a Java program to implement user defined exception handling is compiled successfully.

**Exp. No: 4(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write Java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it’s not a duplicate of any number already read display the complete set of unique values input after the user enters each new value.**

**AIM:**

To write Java program that inputs 5 numbers, each between 10 and 100 inclusive. As each number is read display it only if it’s not a duplicate of any number already read display the complete set of unique values input after the user enters each new value.

**DESCRIPTION:**

Use a one-dimensional array to solve the following problem:Write an application that inputs five numbers, each of which is between 10 and 100, inclusive. As each number is read, display it only if it is not a duplicate of a number already read.

**PROGRAM:**

import java.util.Scanner;

public class Duplicate{

public static void main(String[ ] args){

int a[]={0,0,0,0,0},t,i,j,s=0,r=0;

Scanner z=new Scanner(System.*in*);

System.*out*.println("Enter 5 unique values between 10 & 100 ");

for(j=0;j<5;j++){

t=z.nextInt();

if(t>=10&&t<=100){

for(i=0;i<r;i++){

if(a[i]==t)

s++;

}

if(s>0){

System.*out*.println("Duplicate value found retry");

s--;

j--;

continue;

}

else{

a[j]=t;

r++;

}

}

else {

System.*out*.println("value must be in between 10 & 100");

j--;

}

}

System.*out*.print("The five unique values are ");

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

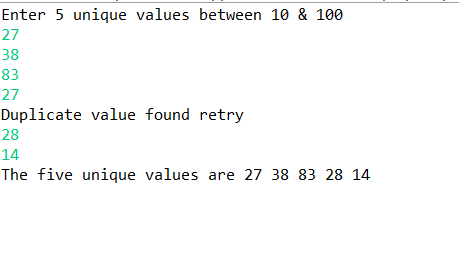
System.*out*.print(a[i]+" ");

}

}

}

**OUTPUT:**

****

**RESULT**:

Program that inputs 5 numbers, each between 10 and 100 inclusive is compiled successfully.

**WEEK-5**

**Exp. No: 5(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a Java program that creates a user interface to perform integer division. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 and Num2 were not integers, the program would throw a Number Format Exception. If Num2 were zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box**

**AIM:**

To write a Java program that creates a user interface to perform integer division.

**DESCRIPTION:**

The AWT supports a rich assortment of graphics methods. All graphics are drawn relative to a window. this can the main window of an applet, a child window of an applet, or a stand alone application window. The origin of each window is at the top-left corner and is 0,0cordinates are specified in pixels. All output to a window takes place through a graphics context.

**PROGRAM:**

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.applet.\*;

import java.lang.String;

import java.util.Scanner;

public class Week10 extends Frame implements ActionListener{

Dialog d;

TextField t1,t2,t3;

Button comp;

public Week10(){

setLayout(new FlowLayout());

setSize(500,500);

t1=new TextField(10);

t2=new TextField(10);

t3=new TextField(10);

comp=new Button("Compute ");

add(new Label("Enter a: "));

add(t1);

add(new Label("Enter b: "));

add(t2);

add(new Label("Result : "));

add(t3);

add(comp);

comp.addActionListener(this);

setVisible(true);

}

public void actionPerformed(ActionEvent ae){

if(ae.getSource()==comp){

try{

t3.setText(Integer.toString((Integer.parseInt(t1.getText().trim()))/(Integer.parseInt(t2.getText().trim()))));

}

catch(ArithmeticException aex){

Dia d1=new Dia("Arithmetic Exception");

d1.setVisible(true);

}

catch(NumberFormatException nfe){

Dia d2=new Dia("Number Format Exception ");

d2.setVisible(true);

}

}

}

public static void main(String ar[]){

new Week10();

}

}

class Dia extends Dialog implements ActionListener{

Button cancel;

Dia(String str){

super(new Frame(),str,true);

cancel=new Button("Cancel");

setLayout(new FlowLayout());

setSize(300,200);

add(new Label("Press the Button"));

add(cancel);

cancel.addActionListener(this);

}

public void actionPerformed(ActionEvent ae){

setVisible(true);

}

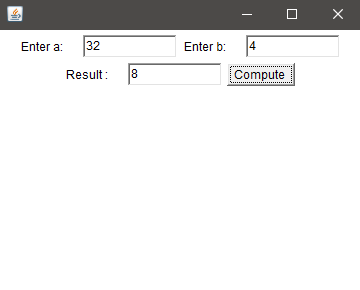
public void paint(Graphics g){

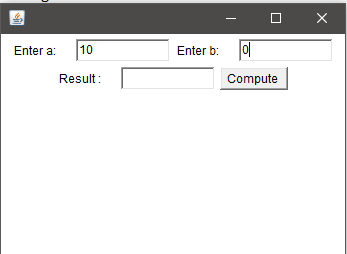
g.drawString("Exception Occured ",10,70);

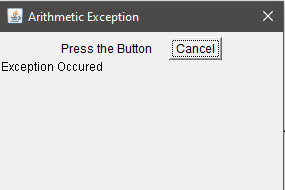
}

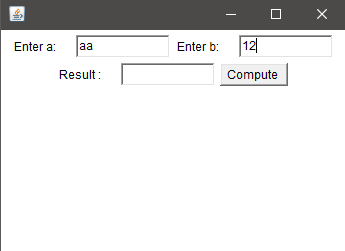
}

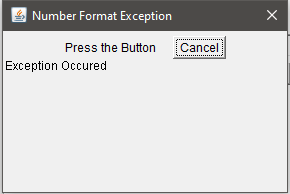
**OUTPUT:**

****

****

****

****

****

**RESULT:**

Java program that creates a user interface to perform integer division is crerated successfully.

**Exp. No: 5(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a Java program that creates three threads. First thread displays ―Good Morning!! every one second, the second thread displays ―Hello!! every two seconds and the third thread**

**displays ―Welcome!! every three seconds.**

**AIM:**

# To write a Java program that creates three threads. First thread displays ―Good Morning!! every one second, the second thread displays ―Hello!! every two seconds and the third thread displays ―Welcome! every three seconds.

**DESCRIPTION:**

**PROGRAM:**

class ChildThread implements Runnable{

Thread t;

ChildThread(String name){

t = new Thread(this, name);

t.start();

}

public void run(){

for(int i=1;i<=5;i++){

try{

if(t.getName().equals("First Thread")){

Thread.*sleep*(1000);

System.*out*.println(t.getName()+": Good Morning");

}

else if(t.getName().equals("Second Thread")){

Thread.*sleep*(2000);

System.*out*.println(t.getName()+": Hello");

}

else{

Thread.*sleep*(3000);

System.*out*.println(t.getName()+": Welcome");

}

}

catch(InterruptedException e){

System.*out*.println(t.getName()+" is interrupted");

}

}

}

}

class Three\_Threads{

public static void main(String args[]){

ChildThread one = new ChildThread("First Thread");

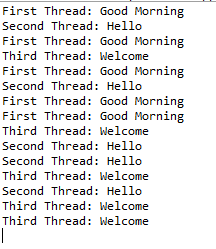
ChildThread two = new ChildThread("Second Thread");

ChildThread three = new ChildThread("Third Thread");

}

}

**OUTPUT:**

****

**RESULT:**

Java program that creates three threads is successfully created.

**WEEK-6**

**Exp. No: 6(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a java program to split a given text file into n parts. Name each part as the name of the original file followed by .part where n is the sequence number of the part**

**AIM:**

To write a java program to split a given text file into n parts. Name each part as the name of the original file followed by part where n is the sequence number of the part file.

**DESCRIPTION:**

**PROGRAM:**

import java.io.PrintWriter;

import java.util.Scanner;

import java.io.\*;

class Text {

public static void main(String args[]) throws IOException {

Scanner input=new Scanner(new File("D:\\filesjava\\writing.txt"));

String line=input.nextLine();

String[] tokens=line.split("[.,?!]");

for(int i=0;i<tokens.length;i++)

System.*out*.println("text.part"+i+" "+tokens[i]);

} }

**OUTPUT:**

****

**RESULT:**

Java program to split a given text file into n parts is executed successfully.

**Exp. No: 6(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

# Write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file inbytes

**AIM:**

# To Write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes.

**DESCRIPTION:**

**PROGRAM:**

import java.io.\*;

class filedemo {

public static void p(String str) {

System.*out*.println(str);

}

public static void analyze(String s) {

File f=new File("D:\\filesjava");

if(f.exists()) {

*p*(f.getName()+" is a file");

*p*(f.canRead()?" is readable":" is not readable");

*p*(f.canWrite()?" is writable":" is not writable");

*p*("Filesize:"+f.length()+" bytes");

*p*("File last mdified:"+f.lastModified());

}

if(f.isDirectory()) {

*p*(f.getName()+" is directory");

*p*("List of files");

String dir[]=f.list();

for(int i=0;i<dir.length;i++)

*p*(dir[i]);

} } }

public class FileInfo {

public static void main(String rr[])throws IOException {

filedemo fd=new filedemo();

BufferedReader br=new BufferedReader(new InputStreamReader(System.*in*));

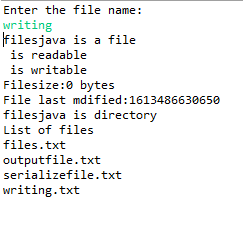
System.*out*.println("Enter the file name:");

String s=br.readLine();

fd.*analyze*(s);

} }

**OUTPUT:**

****

**RESULT:**

Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes is executed successfully.

**WEEK-7**

**Exp. No: 7(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a java program that displays the number of characters, lines and words in a text file**

**AIM:**

To Write a java program that displays the number of characters, lines and words in a text file.

**PROCEDURE / DESCRIPTION:**

**PROGRAM:**

import java.io.\*;

class ReadingFile {

public static void main(String args[]) {

try {

int lines=0,chars=0,words=0;

int code=0;

FileInputStream fis = new FileInputStream("D:\\filesjava\\writing.txt");

while(fis.available()!=0) {

code = fis.read();

if(code!=10)

chars++;

if(code==32)

words++;

if(code= =13) {

lines++;

words++;

} }

System.*out*.println("No.of characters = "+chars);

System.*out*.println("No.of words = "+(words+1));

System.*out*.println("No.of lines = "+(lines+1));

fis.close();

}

catch(FileNotFoundException e) {

System.*out*.println("Cannot find the specified file...");

}

catch(IOException i) {

System.*out*.println("Cannot read file...");

} } }

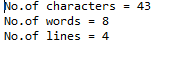
**OUTPUT:**

**Content in writing.txt**

hello!

programming world!

I am developer.

****

**RESULT:**

Java program that displays the number of characters, lines and words in a text fileis displayed successfully.

**Exp. No: 7(b) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a java program that reads a file and displays the file on the screen with line number before each line**

**AIM:**

To write a java program that reads a file and displays the file on the screen with line number before each line.

**PROCEDURE / DESCRIPTION:**

**PROGRAM:**

import java.util.\*;

import java.io.\*;

class DisplayingFile{

public static void main(String args[])throws IOException{

int j=1; char ch;

Scanner scr=new Scanner(System.*in*);

System.*out*.print("\nEnter File name: ");

String str=scr.next();

FileInputStream f=new FileInputStream(str);

System.*out*.println("\nContents of the file are");

int n=f.available();

System.*out*.print(j+": ");

for(int i=0;i<n;i++){

ch=(char)f.read();

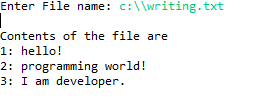
System.*out*.print(ch);

if(ch=='\n'){

System.*out*.print(++j+": ");

}}} }

**OUTPUT:**

****

**RESULT:**

Java program that reads a file and displays the file on the screen with line number before each line is successfully executed.

**WEEK-8**

**Exp. No: 8(a) Date:\_\_\_\_\_\_\_\_\_\_\_\_\_**

**QUESTION:**

**Write a Java program that correctly implements producer consumer problem using the concept of inter thread communication**

**AIM:**

To write a Java program that correctly implements producer consumer problem using the concept of inter thread communication.

**DESCRIPTION:**

In computing, the producer-consumer problem is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

* The producer’s job is to generate data, put it into the buffer, and start again.
* At the same time, the consumer is consuming the data (i.e. removing it from the buffer), one piece at a time.

**PROGRAM:**

import java.io.\*;

class Thread1{

int n;

boolean valueset=false;

synchronized int get(){

if (!valueset)

try{

wait();

}

catch (Exception e){

System.*out*.println("Excepton occur at : "+e);

}

System.*out*.println("get" +n);

try{

Thread.*sleep*(1000);

}

catch (Exception e){

System.*out*.println("Excepton occur at : "+e);

}

valueset=false;

notify();

return n;

}

synchronized int put(int n){

if (valueset)

try{

wait();

}

catch (Exception e){

System.*out*.println("Excepton occur at : "+e);

}

this.n=n;

valueset=true;

System.*out*.println("put"+n);

try{

Thread.*sleep*(1000);

}

catch (Exception e){

System.*out*.println("Excepton occur at : "+e);

}

notify();

return n;

}

}

class Producer implements Runnable{

Thread1 t;

Producer(Thread1 t){

this.t=t;

new Thread(this,"Producer").start();

}

public void run(){

int i=0;

while (true){

t.put(i++);

}

}

}

class Consumer implements Runnable{

Thread1 t;

Consumer(Thread1 t){

this.t=t;

new Thread(this,"Consumer").start();

}

public void run(){

int i=0;

while (true){

t.get();

}

}

}

class ProducerConsumer{

public static void main(String[] args) throws IOException{

Thread1 t=new Thread1();

new Producer(t);

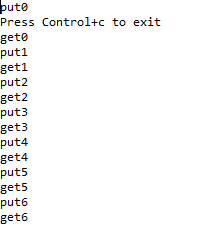
new Consumer(t);

System.*out*.println("Press Control+c to exit");

}

}

**OUTPUT:**

****

**RESULT:**

Java program that correctly implements producer consumer problem using the concept of inter thread communication is executed successfully.