**JAVA PROGRAMMING**

**PRACTICAL**



**SUBMITTED TO:**

**Mrs Ritika Arora**

**SUBMITTED BY:**

**Vidya Sharma**

**SG18821**

**IT-5TH Semester**

|  |
| --- |
| **PROGRAMS** |
| 1. **Write a program to print hello world.** |
| 1. **Write a program to add two integer numbers** |
| 1. **Write a program to obtain the product of two float numbers** |
| 1. **Write a Program to design a class using abstract methods and classes** |
| 1. **Create a simple class to find out the area and perimeter of rectangle and box using super and this keyword.** |
| 1. **Find the average and sum of N numbers.** |
| 1. **Check whether the number is palindrome or not** |
| 1. **Check whether the number is prime or not.** |
| 1. **Calculate Simple Interest** |
| 1. **Check whether the year is leap or not.** |
| 1. **Calculate the sum of digits of a number** |
| 1. **Write a Program To count vowels and consonants in a String** |
| 1. **Write a Program to Make smiley face using applets** |
| 1. **Write a Program to Make chessboard using applets** |
| 1. **Design a class bank without inheritance and show the functions (account number, name, balance, deposit, withdrawal).** |
| 1. **Write a Java program to get and display information (year, month, day, hour, minute) of a default calendar.** |
| 1. **Write a Java Program to Generate a random number up to 100 and print whether it is prime or not.** |
| 1. **Calculate the area of rectangle and a triangle.** |
| 1. **Write a Program for display the Checkboxes, Labels and Text Fields on an AWT** |
| 1. **Write a program to implement bubble sort in java** |
| 1. **Write a java package to show dynamic polymorphism and interface.** |
| 1. **Analog Clock using a Java applet** |
| 1. **Write a program of database connectivity using jdbc-odbc driver** |
| 1. **Write a java program that allows the user to draw lines, rectangles, filled rectangles and rounded rectangles.** |

**AIM: Write a program to print hello world.**

**Theory:**

Java is an object oriented language(OOP). In the program described below, a class called main is defined. Next static method is declared with arguments to be taken inside the method. And then using the System.out.println() statements we get the desired output. Where, System is a pre-defined class that java provides us, out is the static variable that represents output and println method used to print a line.

**Code:**

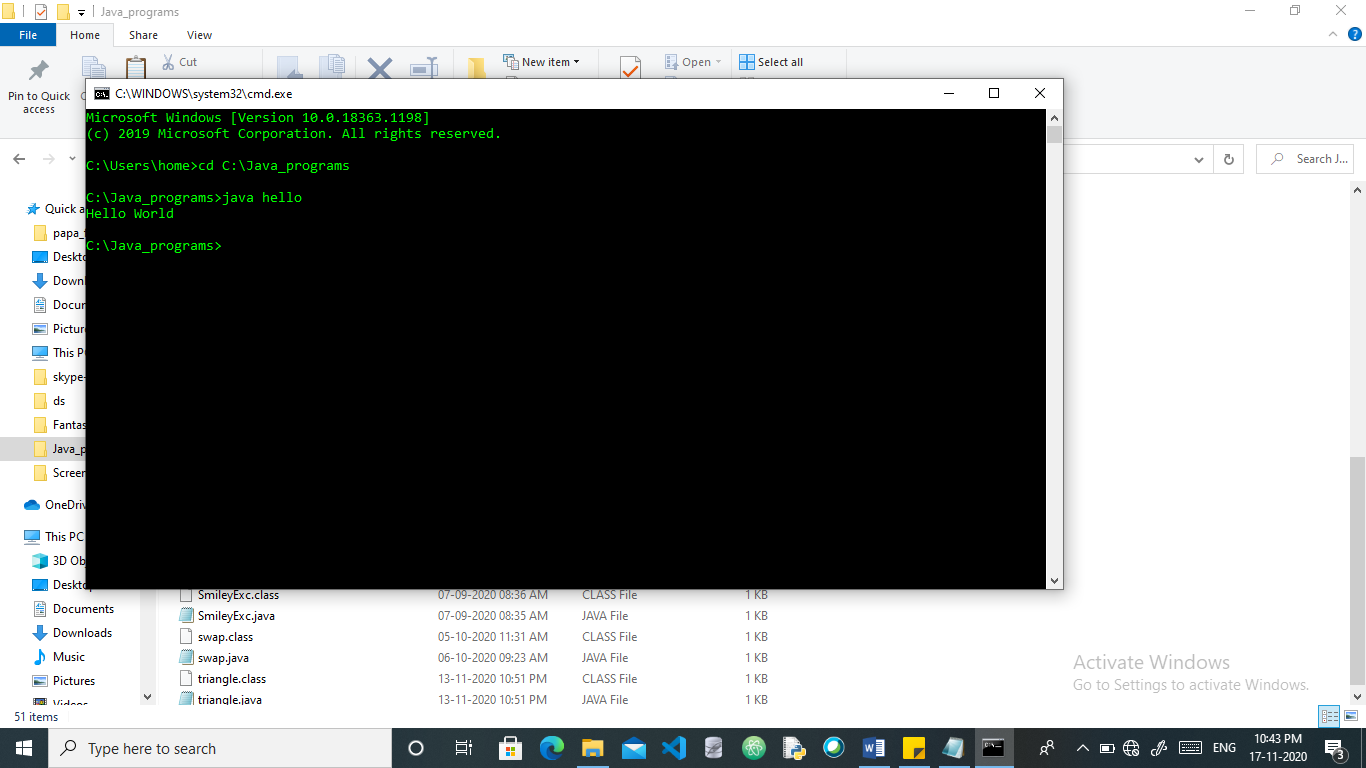
class hello {

public static void main (String [] args) {

System.out.println("Hello World");

}}

**Output:**



**AIM: Write a program to add two integer numbers**

**Theory:**

The below program is constructed using the java operators and the java variables. Here, two integer variables are declared and then by using the “+” operator (*Operators are symbols that perform operations on variables and values)*, operation is performed.2

**Code:**

class add {

public static void main (String [] args)

{

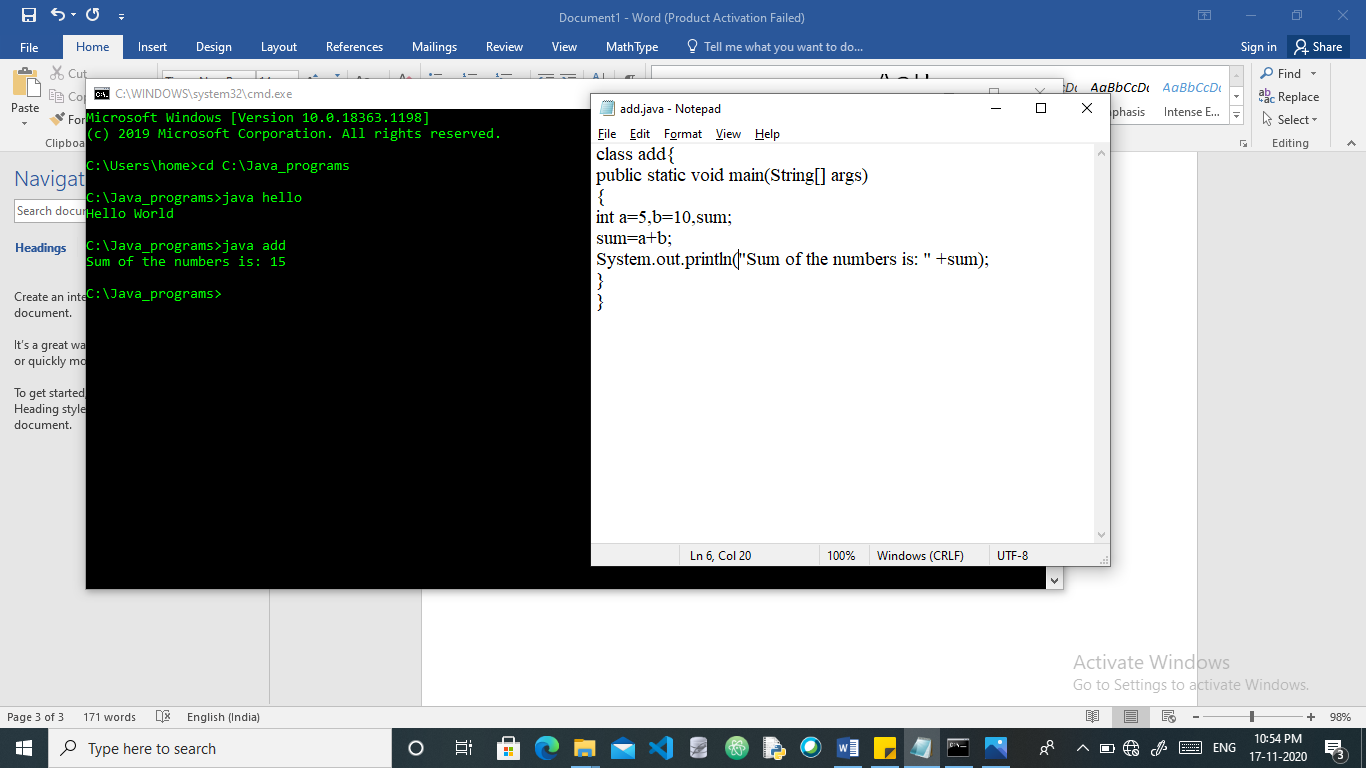
int a=5, b=10, sum;

sum=a+b;

System.out.println("Sum of the numbers is: " +sum);

}}

**Output:**



**AIM: Write a program to obtain the product of two float numbers**

**Theory:**

In this code, we use the concept of java data types and operators. Two floating point numbers are taken and then using operator “\*”, product of the numbers is obtained.

**Code:**

class product {

public static void main (String [] args) {

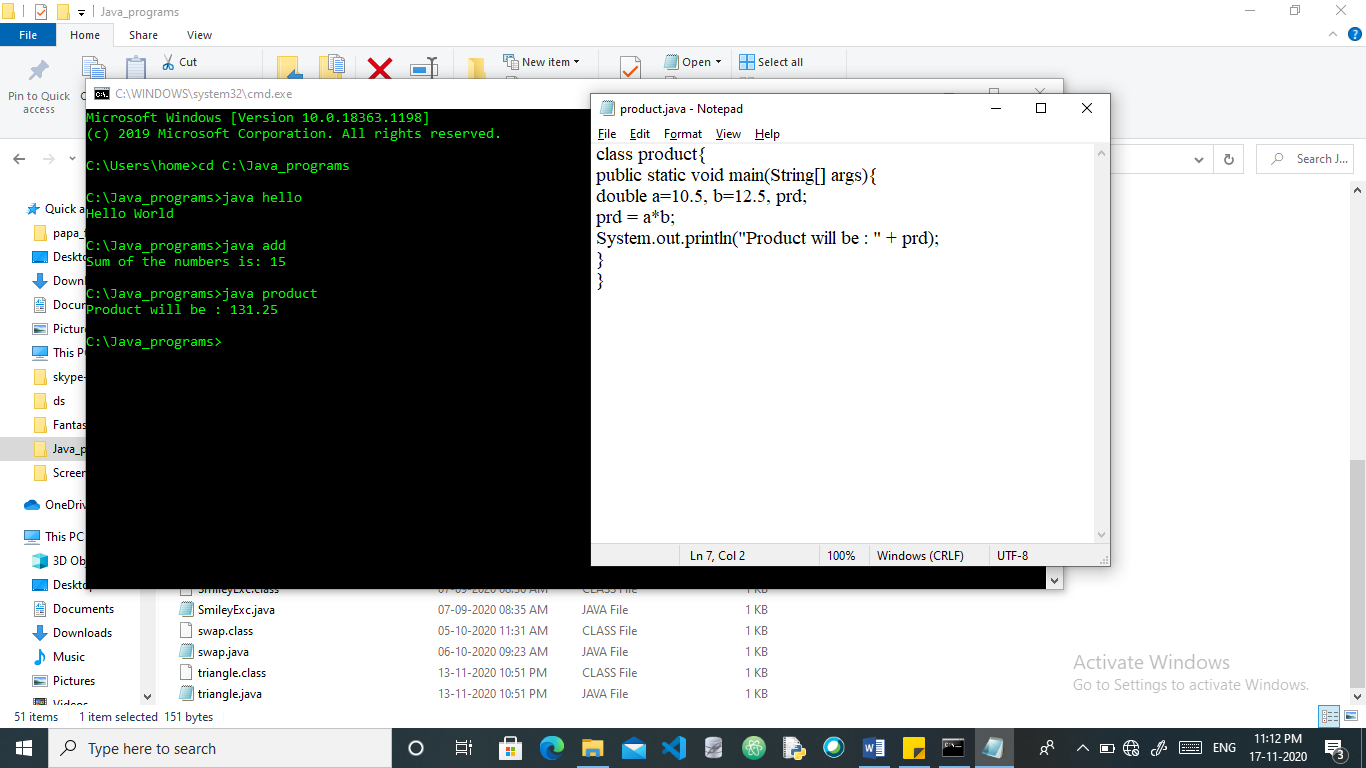
double a=10.5, b=12.5, prd;

prd = a\*b;

System.out.println("Product will be: " + prd);

}}

**Output:**



**AIM: Write a Program to design a class using abstract methods and classes**

**Theory:**

The abstract class in Java cannot be instantiated, we cannot create objects of abstract classes. We use the abstract keyword to declare an abstract class. Abstract method can only be used in an abstract class, and it does not have a body. The body is provided by the subclass.

**Code:**

abstract class Animal {

abstract void makeSound ();

}

class Dog extends Animal {

public void makeSound () {

System.out.println("GOLDEN RETRIEVER");

}}

class Cat extends Animal {

public void makeSound () {

System.out.println("Nico Nico Ni");

}}

class abstractclass {

public static void main (String [] args) {

Dog d1 = new Dog ();

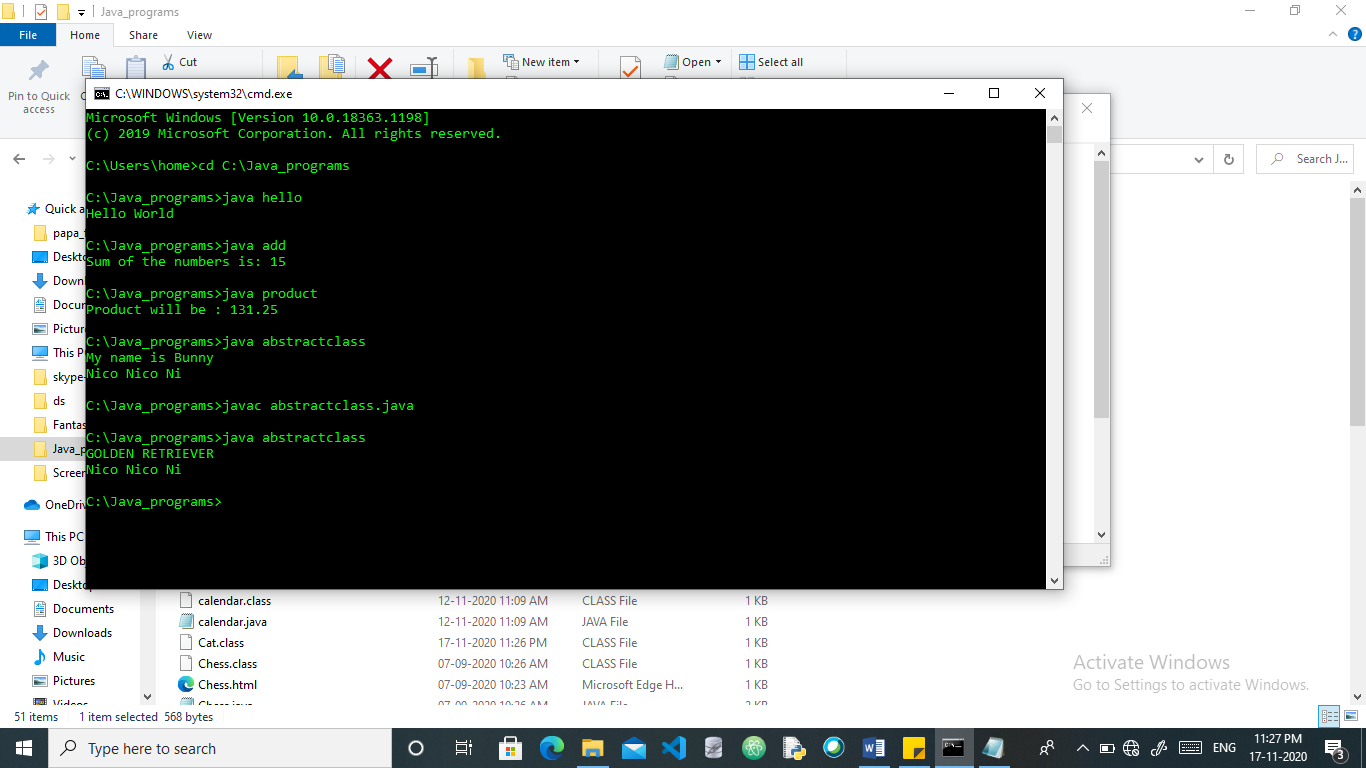
d1. makeSound ();

Cat c1 = new Cat ();

c1. makeSound ();

}}

**Output:**



**AIM: Create a simple class to find out the area and perimeter of rectangle and box using super and this keyword.**

**Theory:**

Super keyword is used to access methods of the parent class while this is used to access methods of the current class. This is a reserved keyword in java i.e., we can’t use it is an identifier, this is used to refer current-class’s instance as well as static members. Super can be used to invoke immediate parent class method.

**Code:**

class measure {

double length = 17.2;

double breadth = 8.5;

measure ()

{

this. length = 20.5;

this. breadth = 81.8;

double area = length\*breadth;

System.out.println(“Area of rectangle: " + area);

double peri = 2\*(length+breadth);

System.out.println("Perimeter of rectangle: " + peri);

}}

public class cal\_are\_per extends measure {

cal\_are\_per ()

{

super ();

double height = 3.6;

double surface\_area = 2\*(height\*breadth) +2\*(height\*length) +2\*(breadth\*length);

double perimeter = 4\*(length +breadth + height);

System.out.println("perimeter of box: "+ perimeter);

System.out.println("surface area of box: " + surface\_area);

}

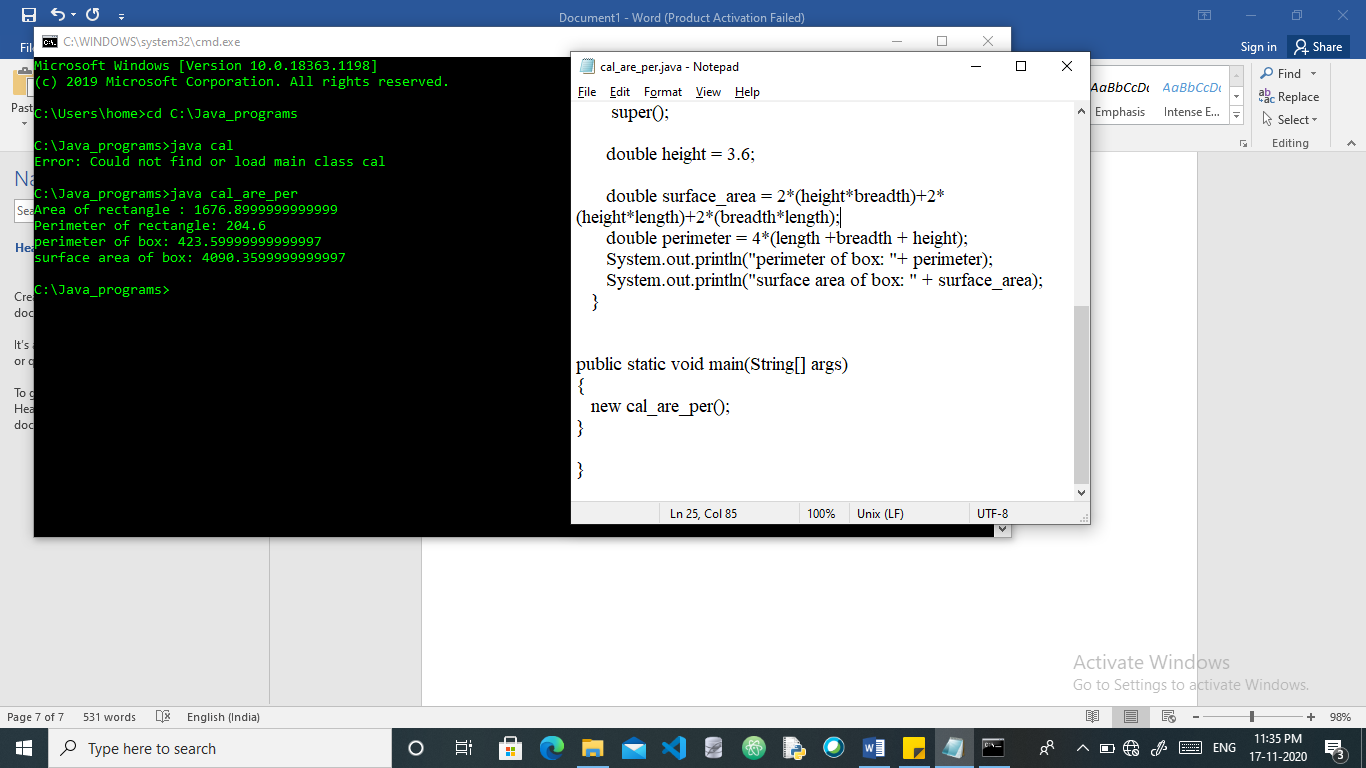
public static void main (String [] args)

{

new cal\_are\_per ();

}}

**Output:**



**AIM: Find the average and sum of N numbers.**

**Theory:**

The average is the outcome from the sum of the numbers divided by the count of the numbers being averaged. Either scanner class can be used to read the input from the user or using int or float integers and then calculate the average. Scanner class is used to get the user input, and it’s found in java.util.package. Below, variables are declared, with the help of Scanner class input is taken from user and average is calculated.

**Code:**

import static java.lang.Float.sum;

import java.util.Scanner;

public class Main {

public static void main(String[] args)

{

int n, count = 1;

float xF, averageF, sumF = 0;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the value of n");

n = sc.nextInt();

while (count <= n)

{

System.out.println("Enter the "+count+" number?");

xF = sc.nextInt();

sumF += xF;

++count;

}

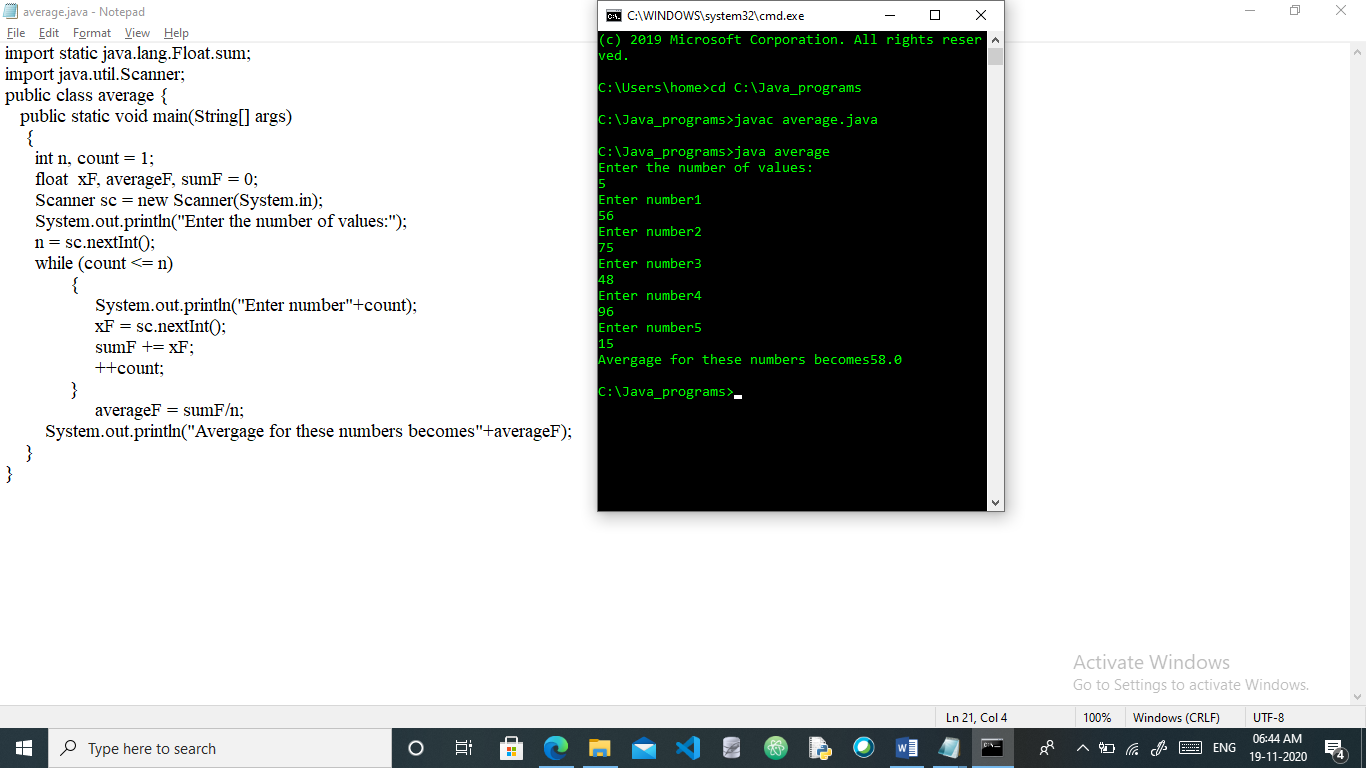
averageF = sumF/n;

System.out.println("The Average is"+averageF);

}

}

**Output:**



**AIM : Check whether the number is palindrome or not**

**Theory:**

A Palindrome Number is a number that is same after reverse. Get the number, hold the number in temporary variable. Reverse and compare the temporary number with reversed number. If both are same, number is palindrome. The number as shown is stored in the temp variable. If the number stored is same as the reversed the number becomes a palindrome.

**Code:**

class Palindrome{

public static void main(String args[]){

int r,sum=0,temp;

int n=454;//It is the number variable to be checked for palindrome

temp=n;

while(n>0){

r=n%10; //getting remainder

sum=(sum\*10)+r;

n=n/10; }

if(temp==sum)

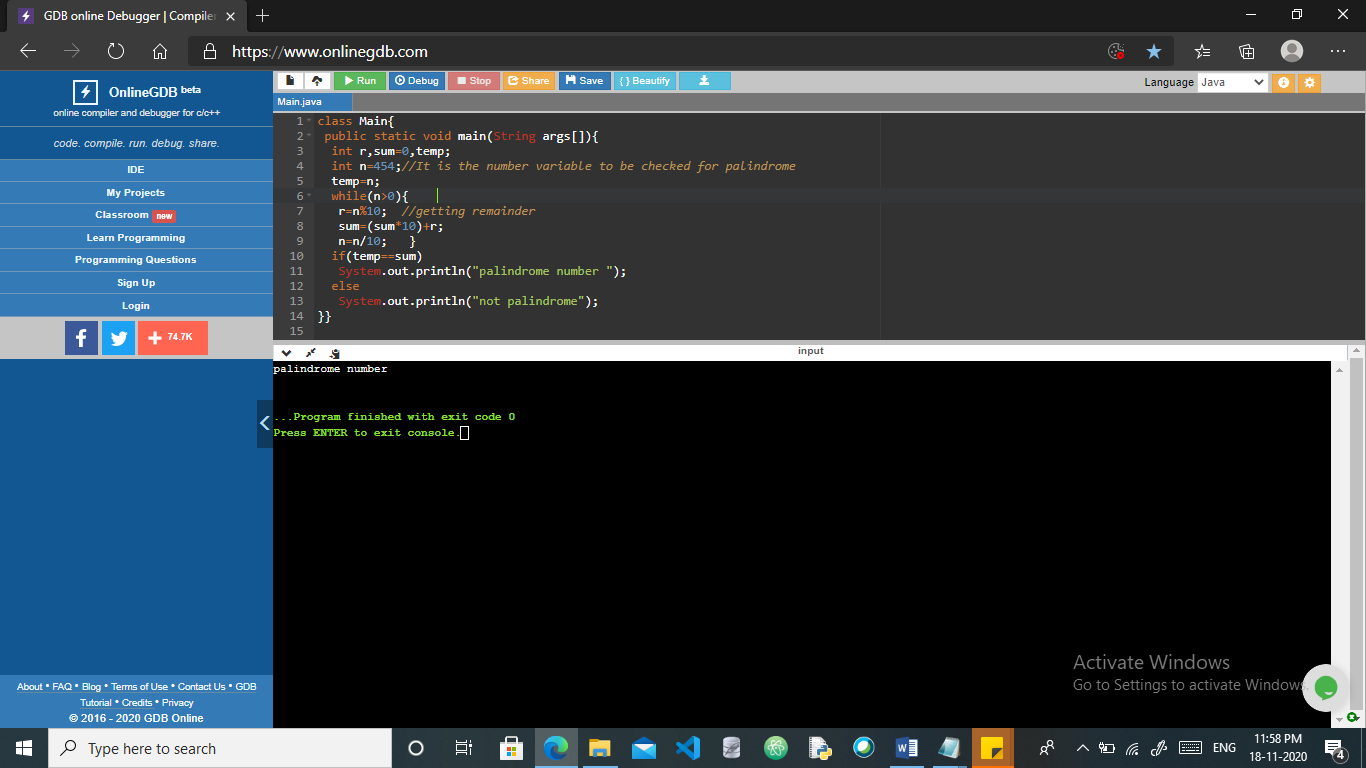
System.out.println("palindrome number ");

else

System.out.println("not palindrome");

}}

**Output:**



**AIM : Check whether the number is prime or not.**

**Theory:**

If number is divisible from 2 to one less than that number, then the number is not prime number otherwise it will be a prime number. So here, if the modulus of the

**Code:**

class prime{

public static void main(String[] args){

int a=15;

if (a%i==0){

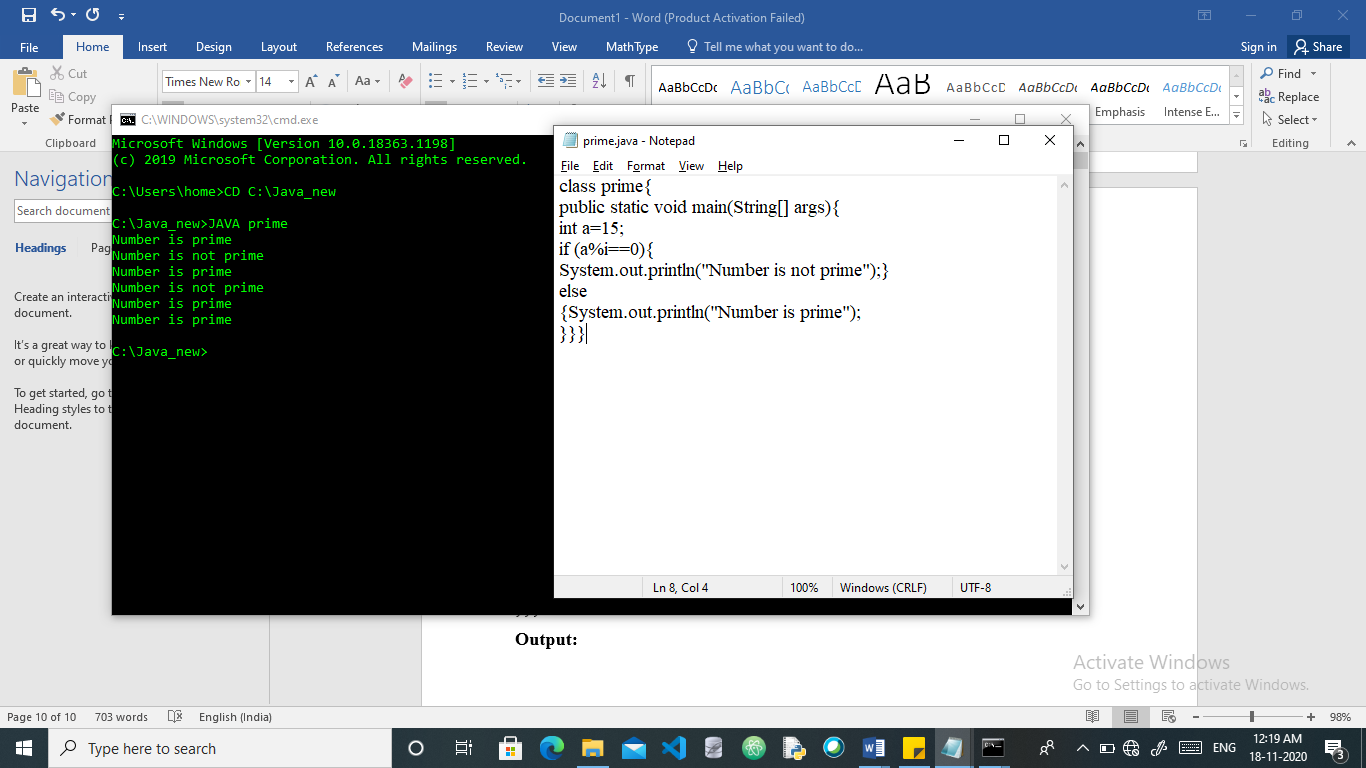
System.out.println("Number is not prime");}

else

{System.out.println("Number is prime");

}}}

**Output:**



**AIM:** **Calculate Simple Interest**

**Theory:**

The formula behind this Simple Interest calculation is Simple Interest = **(Principal Amount \* Rate of Interest \* Number of years) / 100** This Java program allows the user to enter the Principal Amount, total Number of years, and Interest Rate.

**Code:**

import java.util.Scanner;

public class SI{

public static void main(String args[])

{

float p, r, t, sinterest;

Scanner scan = new Scanner(System.in);

System.out.print("Enter the Principal : ");

p = scan.nextFloat();

System.out.print("Enter the Rate of interest : ");

r = scan.nextFloat();

System.out.print("Enter the Time period : ");

t = scan.nextFloat();

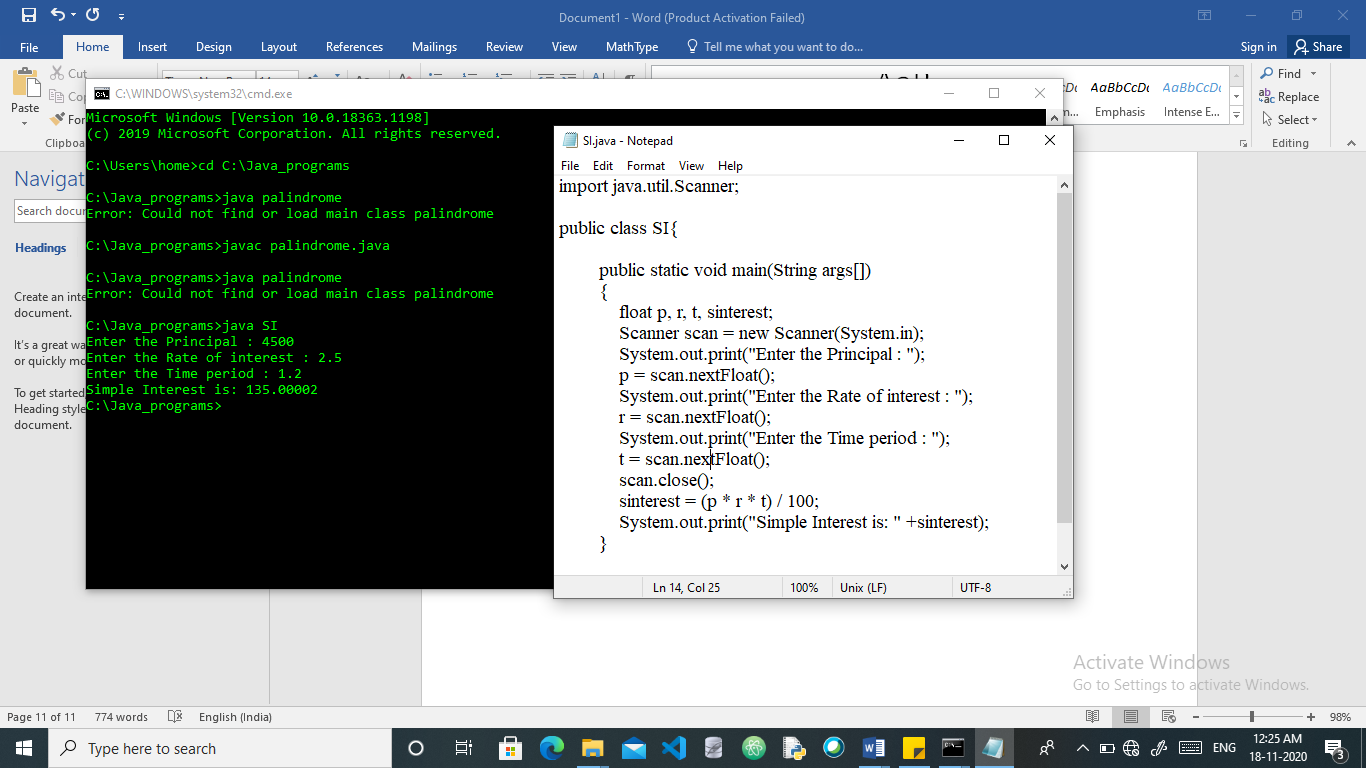
scan.close();

sinterest = (p \* r \* t) / 100;

System.out.print("Simple Interest is: " +sinterest);

}}

**Output:**



**AIM : Check whether the year is leap or not.**

**Theory:**

 This is checked using an if else statement. A leap year is exactly divisible by 4 except for century years (years ending with 00). The century year is a leap year only if it is perfectly divisible by 400. Scanner class is used to take input from the user, here the year. Then it is modulus with4,100 and 400.

**Code:**

import java.util.Scanner;

public class leap {

public static void main(String[] args){

int year;

System.out.println("Enter an Year :: ");

Scanner sc = new Scanner(System.in);

year = sc.nextInt();

if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))

System.out.println("Yeah! its leap");

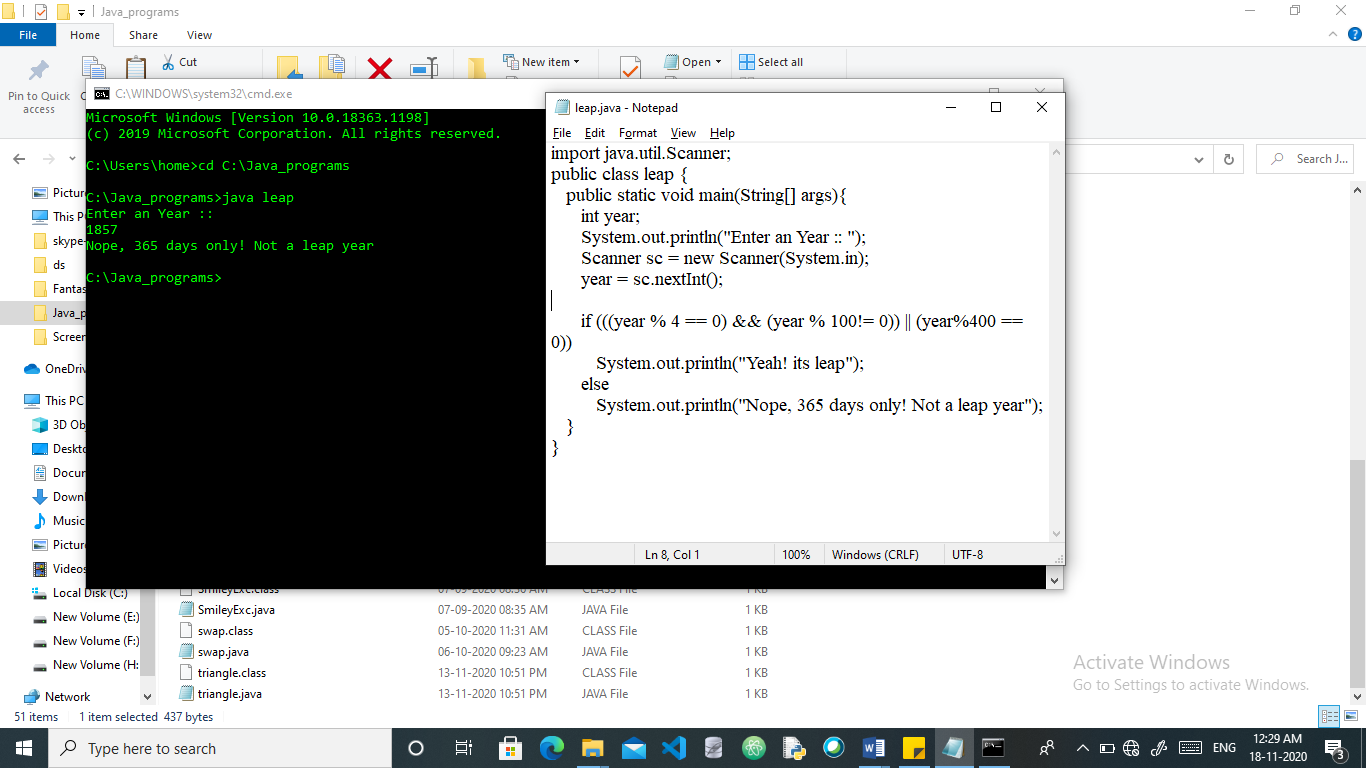
else

System.out.println("Nope, 365 days only! Not a leap year");

}

}

**Output:**



**AIM : Calculate the sum of digits of a number**

**Theory :**

In this Java program, we have used a while loop to find the sum of digits of a given integer number. While loop is a pre-test loop where the expression is evaluated then only statements are executed. It uses a test expression to control the loop. Before every iteration of the loop, the test expression is evaluated.

We can find the last digit of the number and then add it to the sum variable using one line. It will reduce the number of line of the method which is using while loop.

**Code:**

import java.util.Scanner;

public class Main {

public static int digitSum(int number) {

int lastDigit = 0;

int sum = 0;

while(number != 0) {

lastDigit = number % 10;

sum = sum + lastDigit;

number = number / 10;

}

return sum;

}

public static void main(String[] args) {

int number = 0;

int sumOfDigits = 0;

Scanner scan = new Scanner(System.in);

System.out.print("Enter an integer number::");

number = scan.nextInt();

sumOfDigits = digitSum(number);

System.out.println("The sum of digits of" +

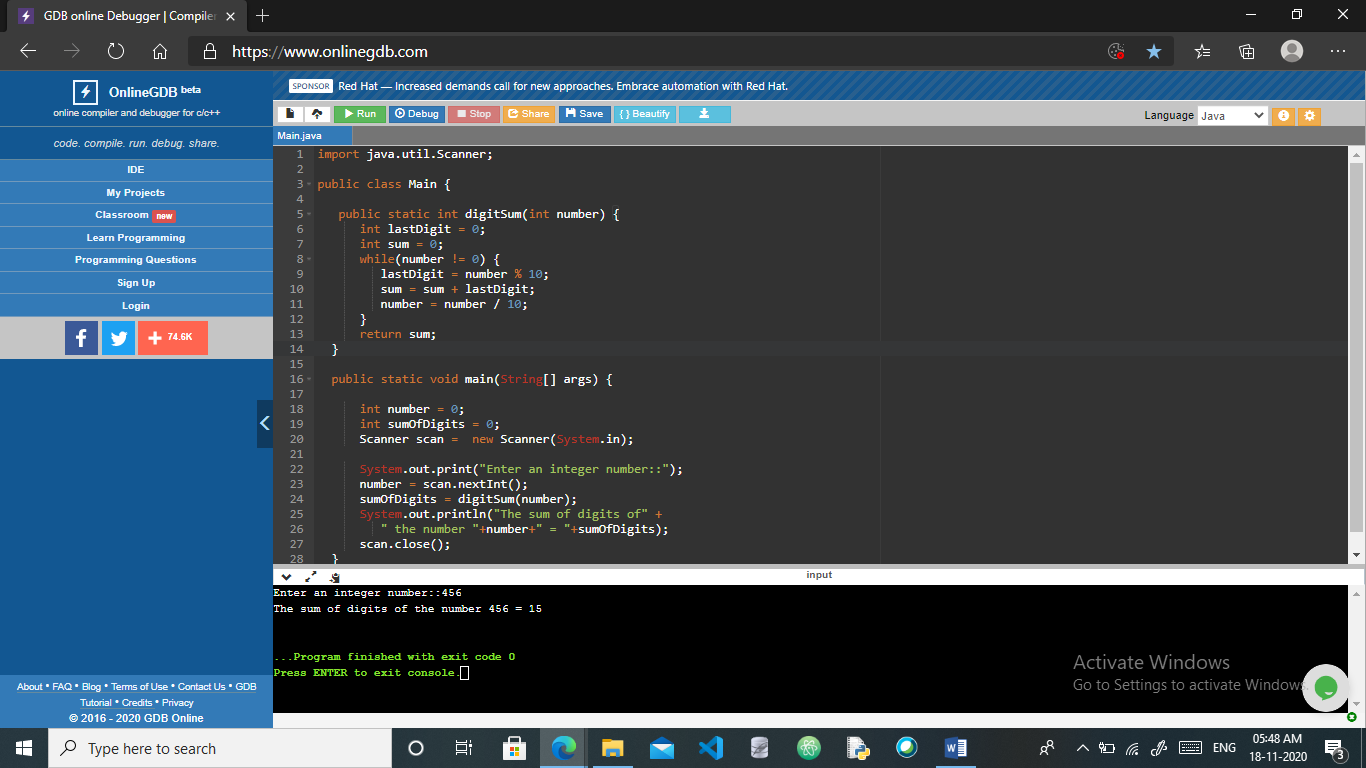
" the number "+number+" = "+sumOfDigits);

scan.close();

}

}

**Output:**



**AIM: Write a Program to count vowels and consonants in a String**

**Theory:**

To count vowels and consonants in a String, iterate over each character of the string and check if the character is vowel or consonant. If the character is vowel, increment vowel count, or if the character is consonant, increment consonant count.

**Code:**

public class CountVowelConsonant {

    public static void main(String[] args) {

        String str = "apple is fruit.";

              String alpha = str.toLowerCase().replaceAll("[^a-z]", "");

        int vowels = 0;

        int consonants = 0;

        for (char ch: alpha.toCharArray()) {

            if(ch == 'a' || ch=='e' || ch == 'i' || ch=='o' || ch == 'u')

                vowels++;

            else

                consonants++;

        }

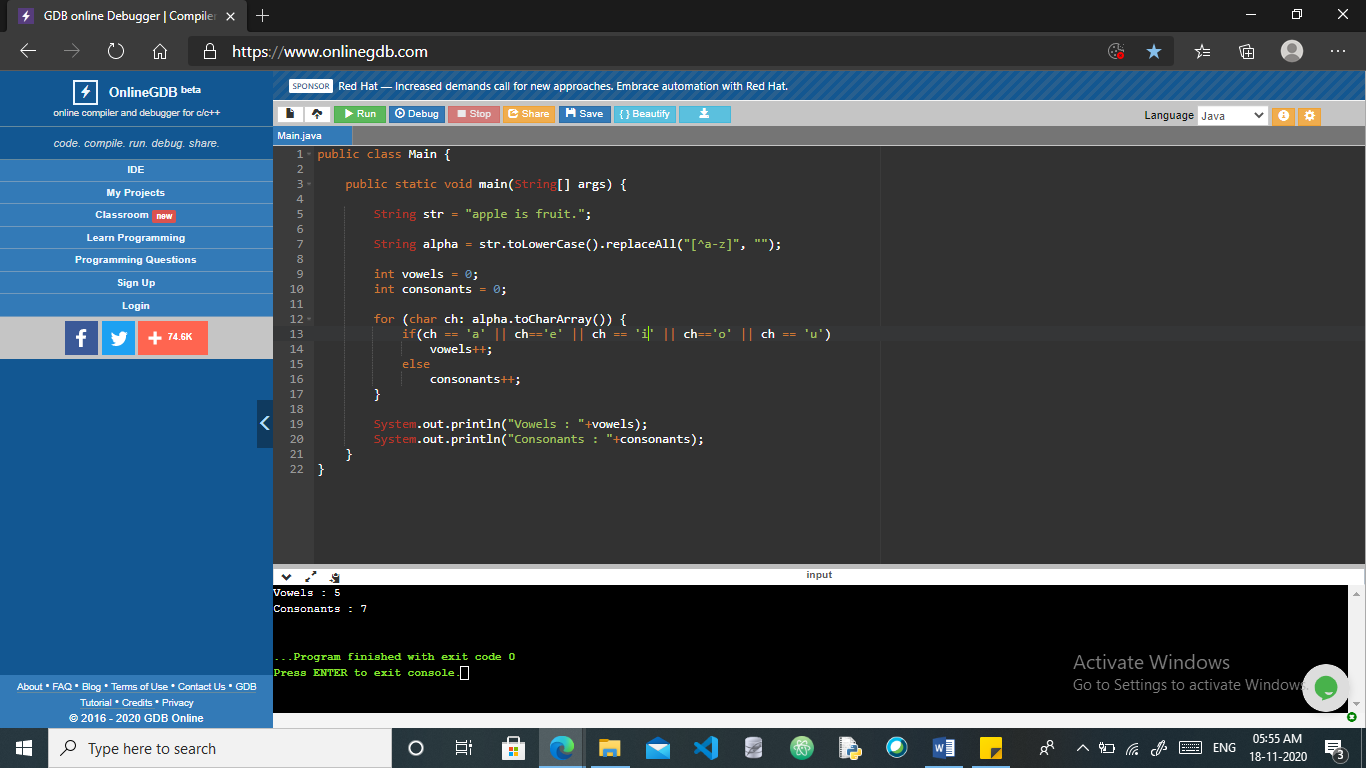
        System.out.println("Vowels : "+vowels);

        System.out.println("Consonants : "+consonants);

    }

}

**Output:**



**AIM: Write a Program to Make smiley face using applets**

**Theory:**

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side. It works at client side so less response time. Every applet is an extension of the *java.applet. Applet class*. The base Applet class provides methods that a derived Applet class may call to obtain information and services from the browser context. An applet may be invoked by embedding directives in an HTML file and viewing the file through an applet viewer or Java-enabled browser. The <applet> tag is the basis for embedding an applet in an HTML file.

**Code:**

import java.applet.Applet;

import java.awt.\*;

public class Smiley extends Applet {

public void paint(Graphics g) {

g.setColor(Color.yellow);

g.fillOval(20,20,150,150);

g.setColor(Color.black);

g.fillOval(50,60,15,25);

g.fillOval(120,60,15,25);

int x[] = {95,85,106,95};

int y[] = {85,104,104,85};

g.drawPolygon(x, y, 4);

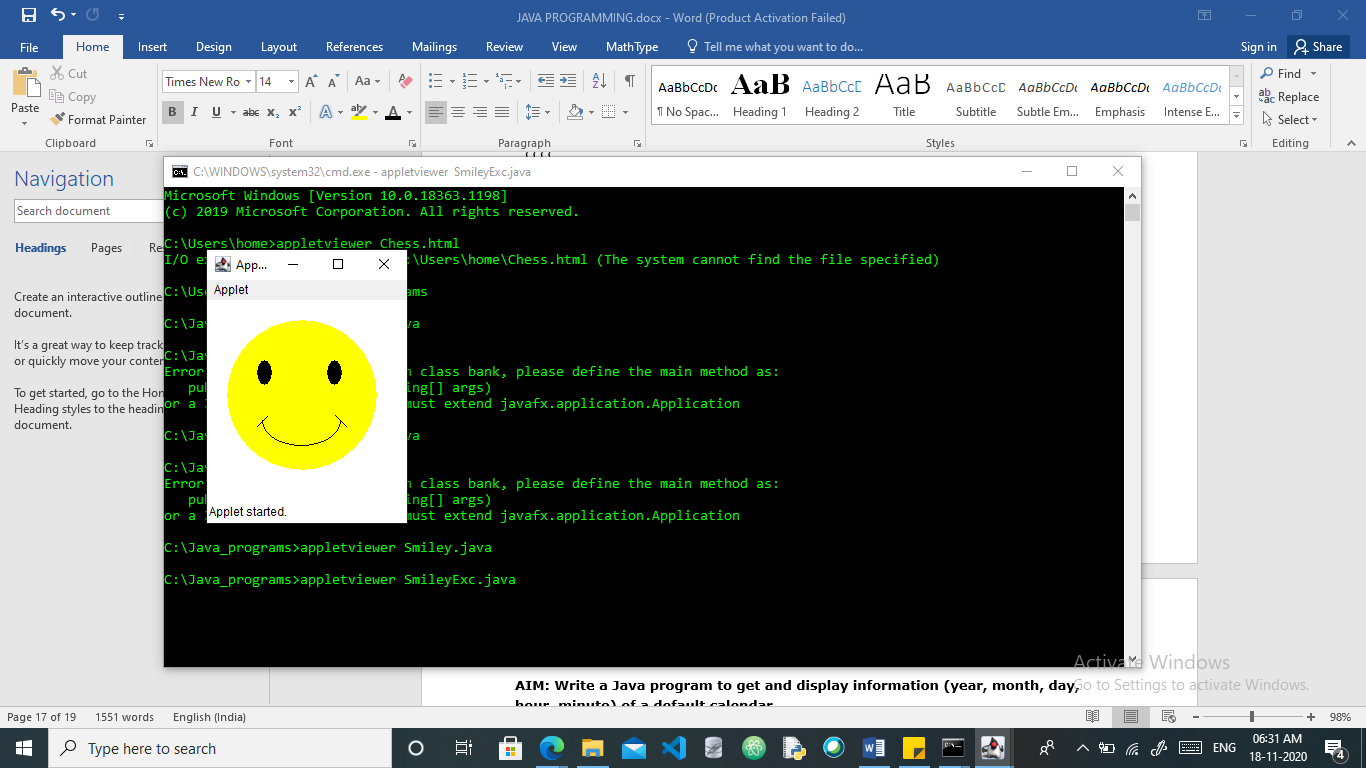
g.drawArc(55,95,78,50,0,-180);

g.drawLine(50,126,60,116);

g.drawLine(128,115,139,126);

}}

**Output:**



**AIM: Write a Program to Make chessboard using applets**

**Theory:**

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.

Create a rectangle with length and breadth of 20 unit each, with 10 rows and columns of chess.

As soon as even position occurs in row and column change the color of a rectangle with BLACK, else it will be WHITE.

Two files are created one .java and other .html. The .html file embed the .java file on web.

**Code:**

**Chess.java**

import java.awt.\*;

import java.applet.\*;

public class Chess extends Applet {

static int N = 10;

public void paint(Graphics g)

{ int x, y;

for (int row = 0; row < N; row++) {

for (int col = 0; col < N; col++) {

x = row \* 20;

y = col \* 20;

if ((row % 2 == 0) == (col % 2 == 0))

g.setColor(Color.BLACK);

else

g.setColor(Color.WHITE);

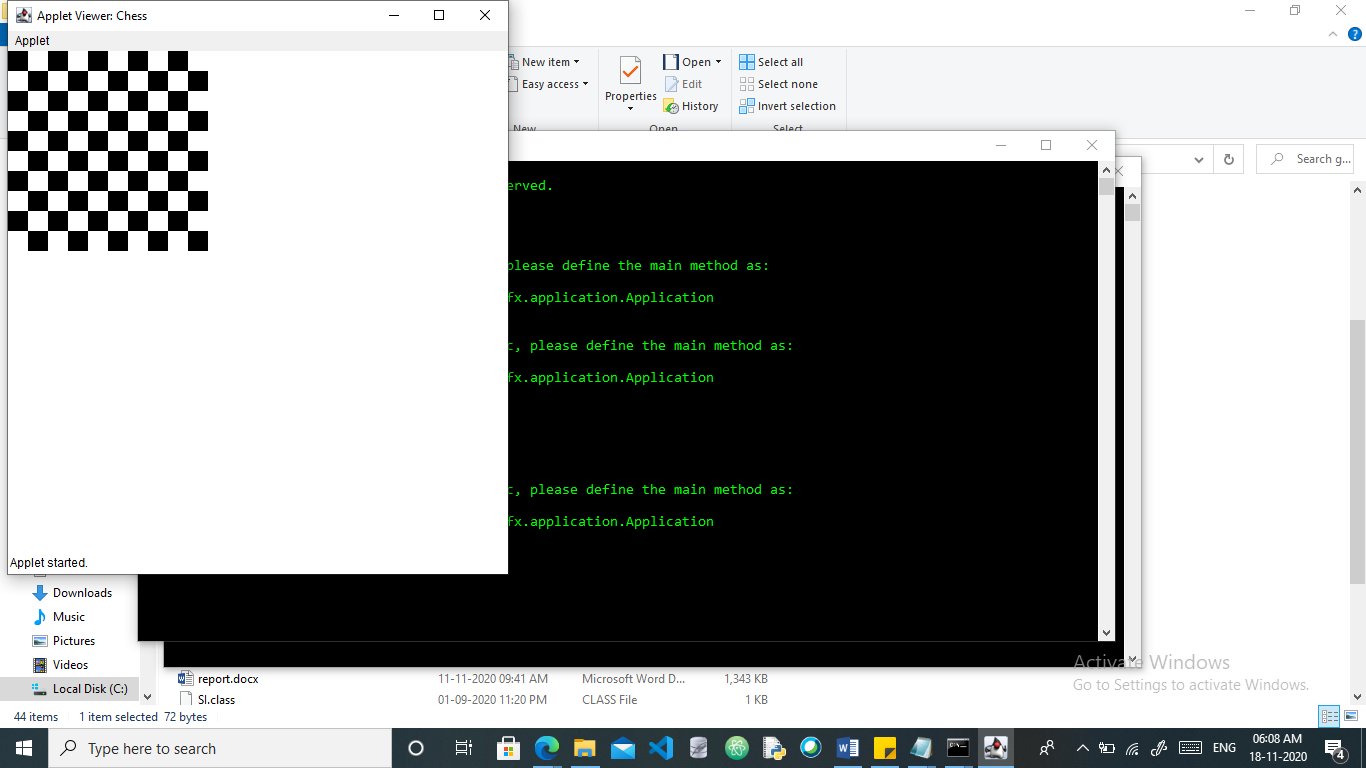
g.fillRect(x, y, 20, 20);

}}}}

**Chess.html**

<applet code="Chess" archive="Chess.java" WIDTH=500 HEIGHT=500></applet>

**Output :**



**AIM: Design a class bank without inheritance and show the functions (account number, name, balance, deposit, withdrawal).**

**Theory:**

Functions account, deposit, withdraw, balance, interest are initialized and then are called at the driver code shown. Objects are created and then the required values are passed as arguments for them. Giving the required results.

**Code:**

class bank

{

private double balance;

private double interest;

public SavingsAccount()

{

balance = 0;

interest = 0;

}

public SavingsAccount(double initialBalance, double initialInterest)

{

balance = initialBalance;

interest = initialInterest;

}

public void deposit(double amount)

{

balance = balance + amount;

}

public void withdraw(double amount)

{

balance = balance - amount;

}

public void addInterest()

{

balance = balance + balance \* interest;

}

public double getBalance()

{

return balance;

}

}

public class SavingsAccountTester

{

public static void main(String[] args)

{

SavingsAccount VidyasSavings = new SavingsAccount(10000, 0.10);

VidyasSavings.withdraw(2250);

VidyasSavings.deposit(4500);

VidyasSavings.addInterest();

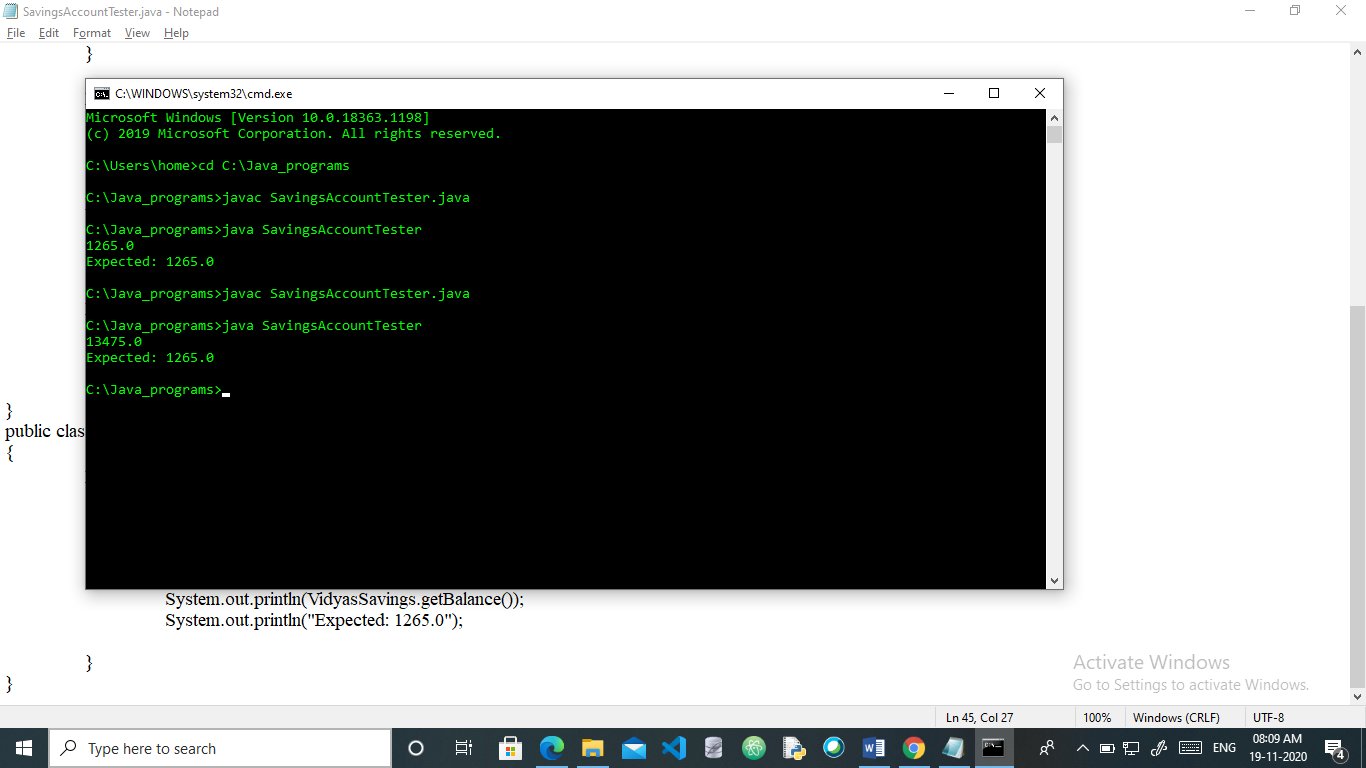
System.out.println(VidyasSavings.getBalance());

System.out.println("Expected: 1265.0");

}

}

**Output:**



**AIM: Write a Java program to get and display information (year, month, day, hour, minute) of a default calendar.**

**Theory:**

The idea is use to get() method of Main class to get the day,month,year,hour and minute. The get() method takes one parameter and returns the value of the passed field.

**Code:**

import java.util.\*;

public class Main {

public static void main(String[] args)

{

Calendar cal = Calendar.getInstance();

System.out.println();

System.out.println("Year: " + cal.get(Calendar.YEAR));

System.out.println("Month: " + cal.get(Calendar.MONTH));

System.out.println("Day: " + cal.get(Calendar.DATE));

System.out.println("Hour: " + cal.get(Calendar.HOUR));

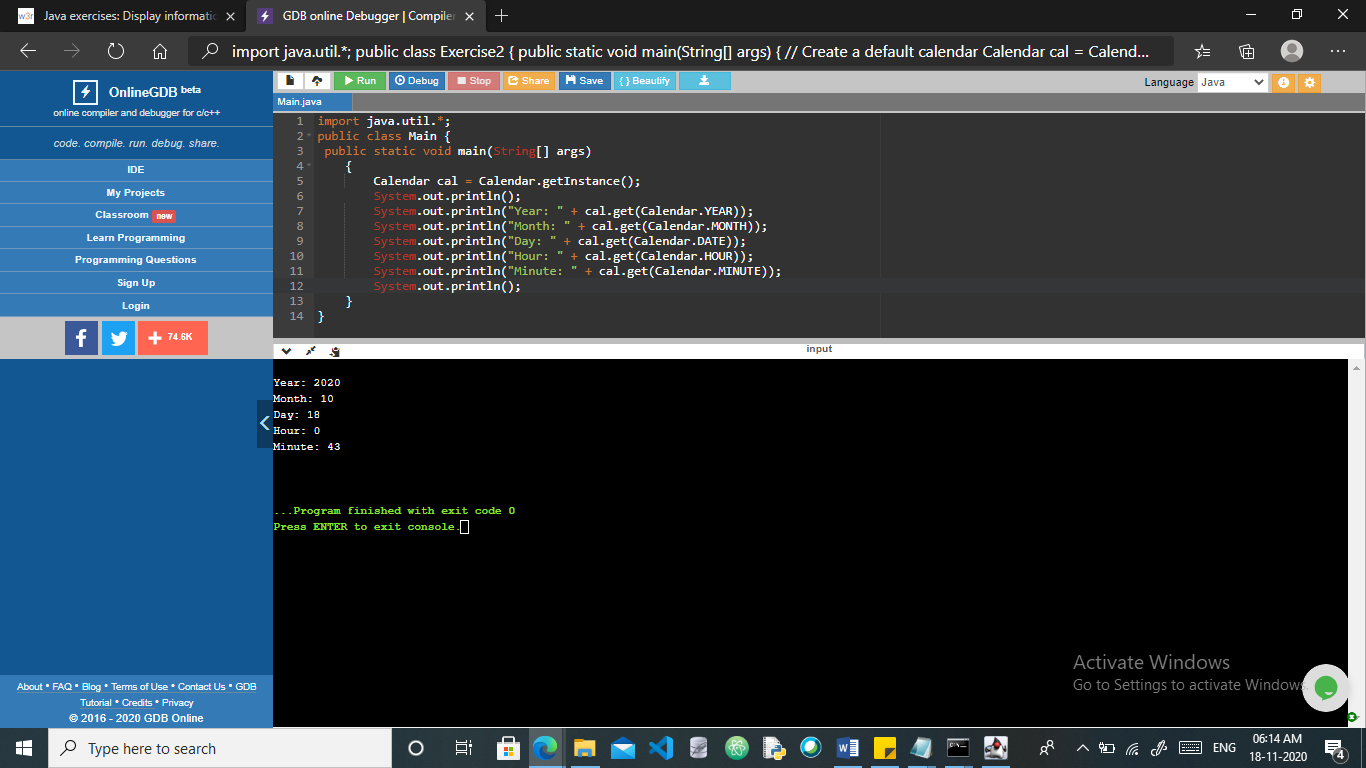
System.out.println("Minute: " + cal.get(Calendar.MINUTE));

System.out.println();

}

}

**Output :**



**AIM: Write a Java Program to Generate a random number up to 100 and print whether it is prime or not.**

**Theory:**

The program used the concept of generating a Random number and also checking its primeness. First a random number is generated upto 100, using the Random() method. Then that number is stored in an int variable. Then condition is checked for that variable and outputs if number is prime or not.

**Code:**

import java.util.\*;

class generaterandom{

public static void main(String[] args){

int counter,flag=0,m=0,var=0;

Random rnum=new Random();

System.out.println("Random number is : ");

for(counter=1;counter<=1;counter++)

var=rnum.nextInt(100);

System.out.println(var);

m=var/2;

if(var==0||var==1){

System.out.println("is not prime");

}else{

for(int i=2;i<m;i++)

if(var%i==0){

System.out.println("is not prime");

flag=1;

break;

}}

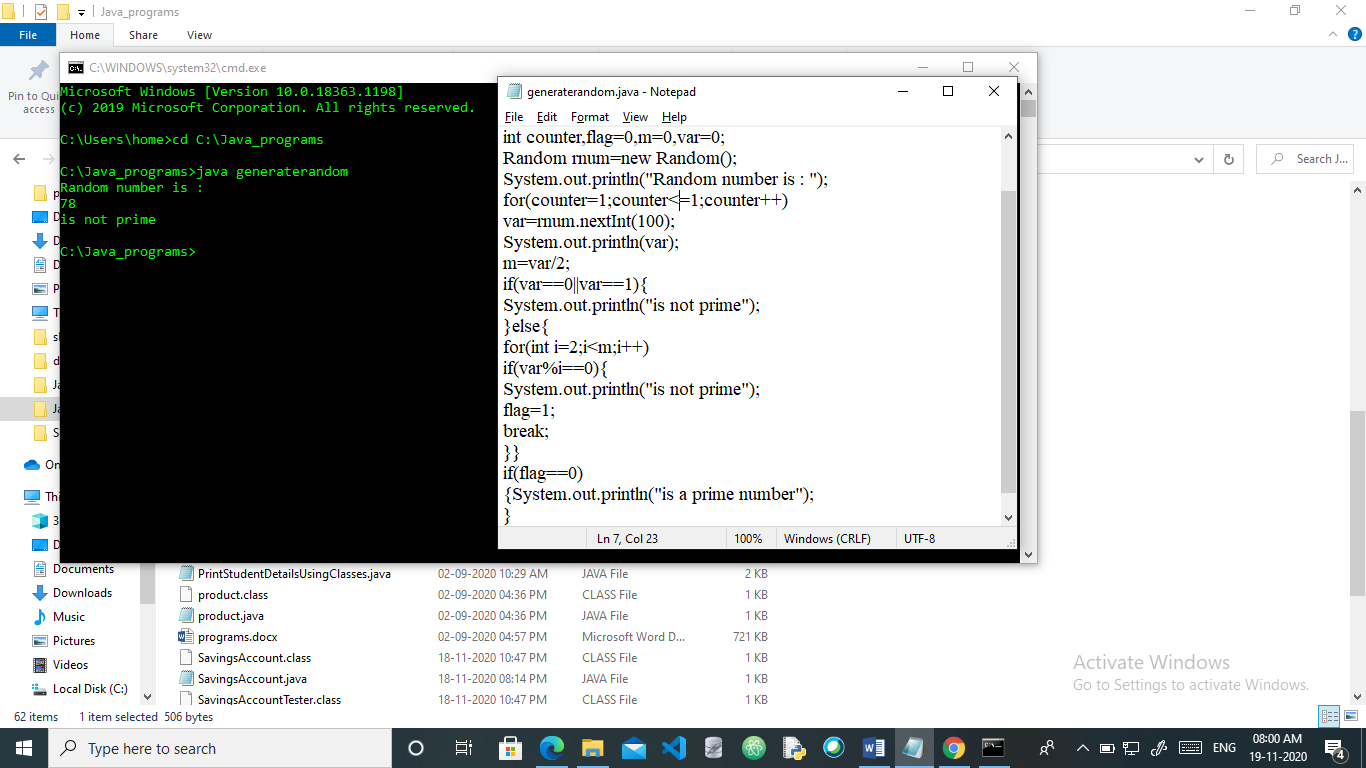
if(flag==0)

{System.out.println("is a prime number");

}

}}

**Output:**



**AIM: Calculate the area of rectangle and a triangle.**

**Theory:**

A class named AREA is created. Inside it area of rectangle and triangle is calculated by first taking values for triangle and then for the rectangle.

**Code:**

class AREA {

public static void main (String args []) {

double base = 15.0;

double height = 25.0;

double area=(base\*height)/2;

System.out.println("Area of triangle is:"+area);

double length = 15.0;

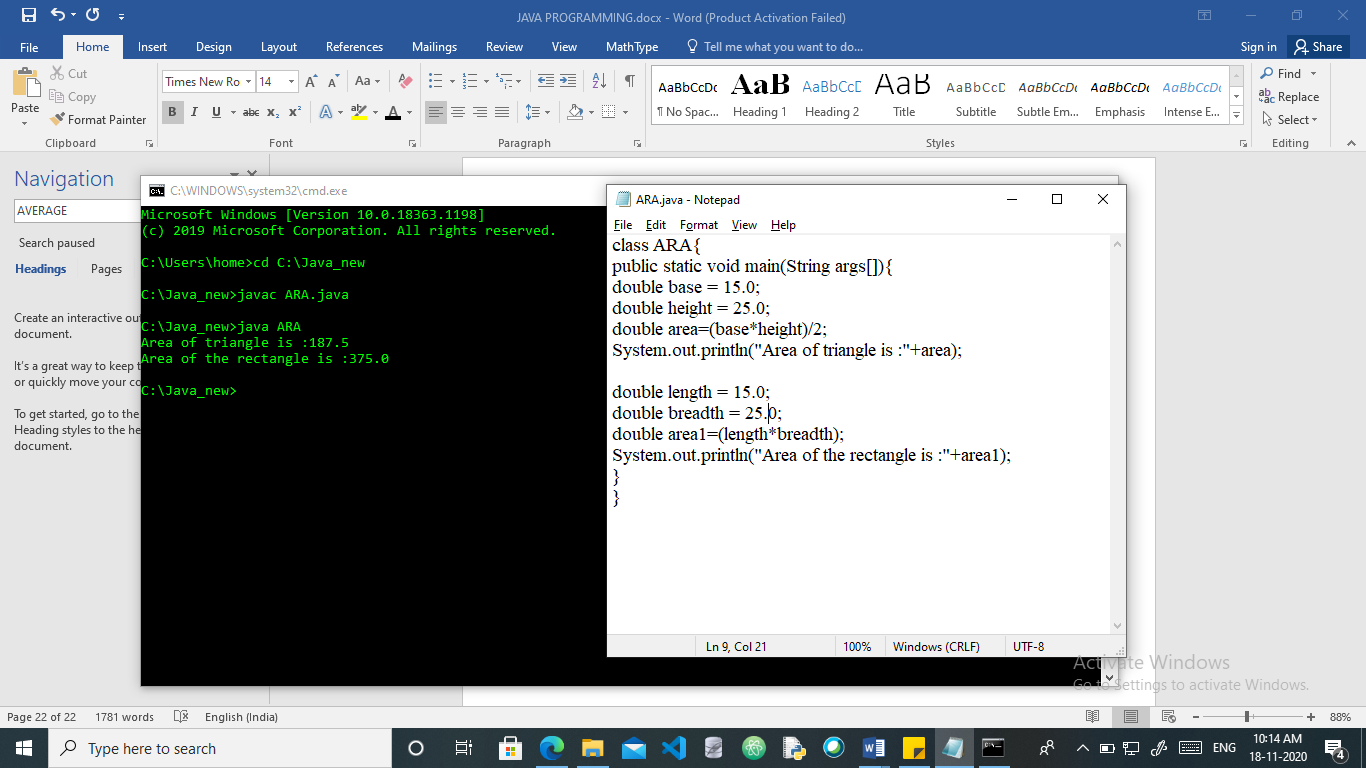
double breadth = 25.0;

double area1=(length\*breadth);

System.out.println("Area of the rectangle is:"+area1);

}}

**Output:**



**AIM: Write a Program for display the Checkboxes, Labels and Text Fields on an AWT**

**Theory: pussgrc library managemt borrow,storage, login, function,**

**AWT** stands for **Abstract Window Toolkit**. It is a platform dependent API for creating Graphical User Interface (GUI) for java programs. The [object](https://www.javatpoint.com/object-and-class-in-java) of Label class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly.   
The [object](https://www.javatpoint.com/object-and-class-in-java) of a Text Field class is a text component that allows the editing of a single line text. It inherits Text Component

class. The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

**Code:**

import java.awt.\*;

public class Main

{

Main(){

Frame f= new Frame("Representing Checkbox,labels,textfield!!");

Checkbox checkbox1 = new Checkbox("Male");

checkbox1.setBounds(150,120, 50,50);

Checkbox checkbox2 = new Checkbox("Female", true);

checkbox2.setBounds(150,150, 50,50);

f.add(checkbox1);

f.add(checkbox2);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

TextField t1;

t1=new TextField("Enter your name!");

t1.setBounds(250,300, 400,130);

f.add(t1);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

Label l1;

l1=new Label("Select one option");

l1.setBounds(150,200, 200,60);

f.add(l1);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

public static void main(String args[])

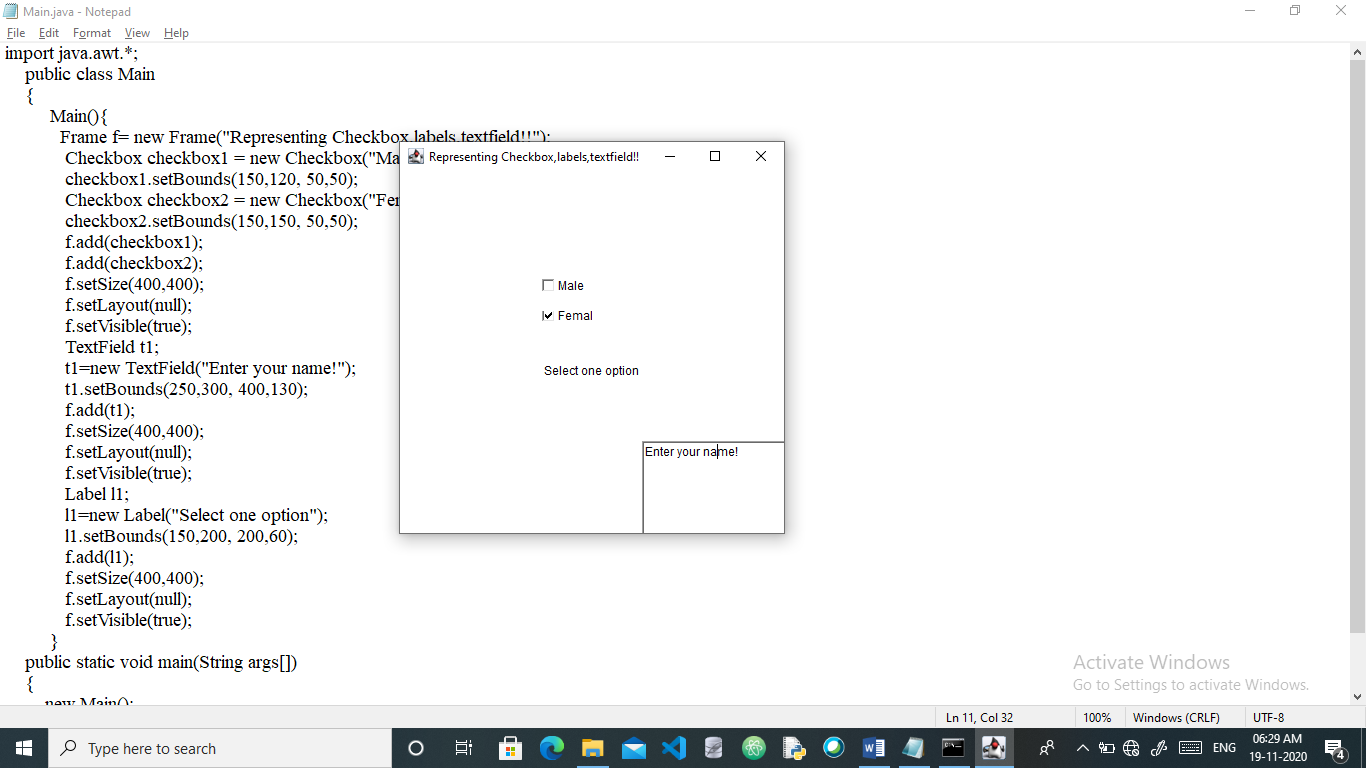
{

new Main();

}

}

**Output**



**AIM: Write a program to implement bubble sort in java**

**Theory:**

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of Ο(n2) where **n** is the number of items. In bubble sort algorithm, array is traversed from first element to last element. Here, current element is compared with the next element. If current element is greater than the next element, it is swapped.

**Code:**

public class Bubble {

static void bubbleSort(int[] arr) {

int n = arr.length;

int temp = 0;

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

for(int j=1; j < (n-i); j++){

if(arr[j-1] > arr[j]){

temp = arr[j-1];

arr[j-1] = arr[j];

arr[j] = temp;

}

}}}

public static void main(String[] args) {

int arr[] ={52,58,2,5,41};

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

System.out.print(arr[i] + " "); }

System.out.println();

bubbleSort(arr);

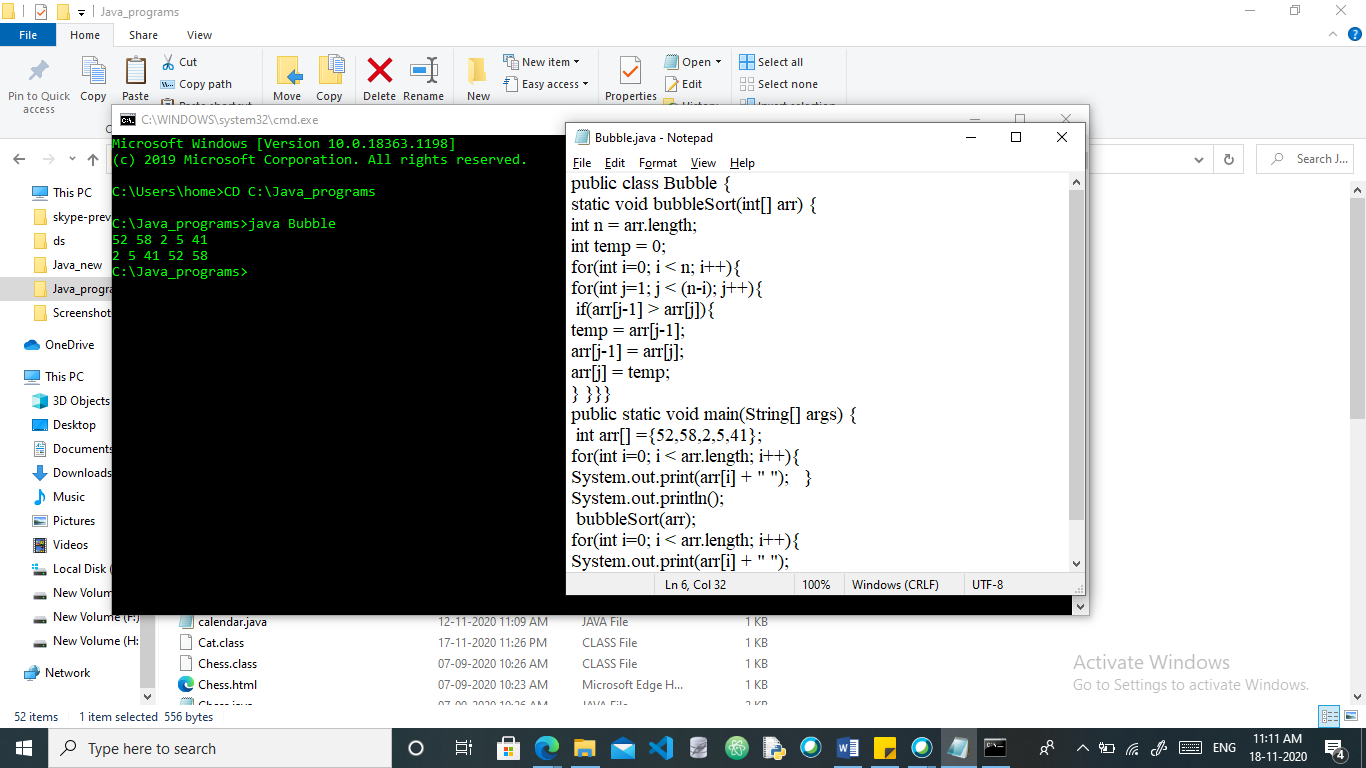
for(int i=0; i < arr.length; i++){

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

}

}}}

**Output:**



**AIM: Write a java package to show dynamic polymorphism and interface.**

**Theory:**

**Dynamic polymorphism** is a process or mechanism in which a call to an overridden method is to resolve at runtime rather than compile-time. It is also known as [**runtime polymorphism**](https://www.javatpoint.com/runtime-polymorphism-in-java) or **dynamic method dispatch**. We can achieve dynamic polymorphism by using the overriding. In this process, an overridden method is called through a reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

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.

**Code:**

class Language {

public void displayInfo() {

System.out.println("Common English Language");

}

}

class Java extends Language {

@Override

public void displayInfo() {

System.out.println("Java Programming Language");

}

}

class Main {

public static void main(String[] args) {

// create an object of Java class

Java j1 = new Java();

j1.displayInfo();

// create an object of Language class

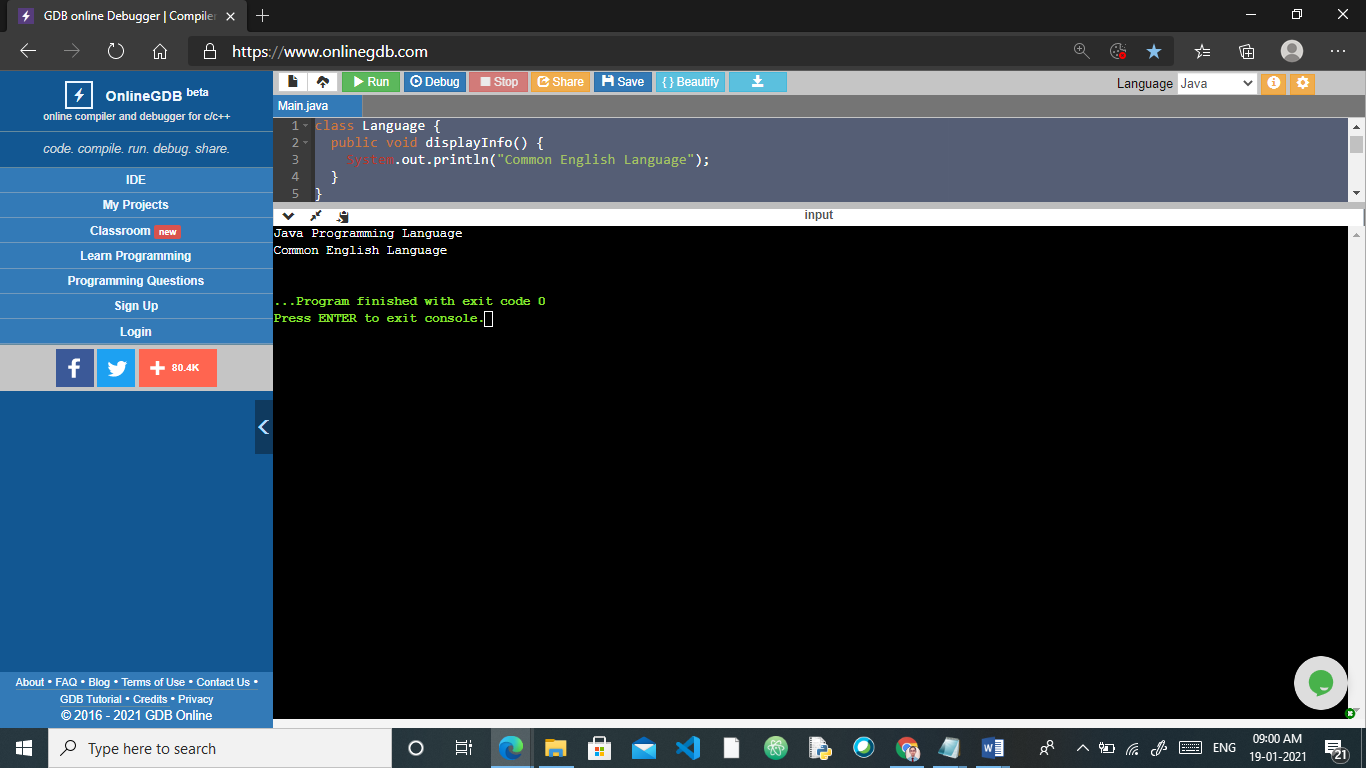
Language l1 = new Language();

l1.displayInfo();

}

}

**OUTPUT:**



**AIM: Write a java program to develop an analog Clock**

**Code:**

import java.applet.\*;

import java.awt.\*;

import java.util.\*;

import java.text.\*;

public class MyClock extends Applet implements Runnable {

int width, height;

Thread t = null;

boolean threadSuspended;

int hours=0, minutes=0, seconds=0;

String timeString = "";

public void init() {

width = getSize().width;

height = getSize().height;

setBackground( Color.black );

}

public void start() {

if ( t == null ) {

t = new Thread( this );

t.setPriority( Thread.MIN\_PRIORITY );

threadSuspended = false;

t.start();

}

else {

if ( threadSuspended ) {

threadSuspended = false;

synchronized( this ) {

notify();

}

}

}

}

public void stop() {

threadSuspended = true;

}

public void run() {

try {

while (true) {

Calendar cal = Calendar.getInstance();

hours = cal.get( Calendar.HOUR\_OF\_DAY );

if ( hours > 12 ) hours -= 12;

minutes = cal.get( Calendar.MINUTE );

seconds = cal.get( Calendar.SECOND );

SimpleDateFormat formatter

= new SimpleDateFormat( "hh:mm:ss", Locale.getDefault() );

Date date = cal.getTime();

timeString = formatter.format( date );

// Now the thread checks to see if it should suspend itself

if ( threadSuspended ) {

synchronized( this ) {

while ( threadSuspended ) {

wait();

}

}

}

repaint();

t.sleep( 1000 ); // interval specified in milliseconds

}

}

catch (Exception e) { }

}

void drawHand( double angle, int radius, Graphics g ) {

angle -= 0.5 \* Math.PI;

int x = (int)( radius\*Math.cos(angle) );

int y = (int)( radius\*Math.sin(angle) );

g.drawLine( width/2, height/2, width/2 + x, height/2 + y );

}

void drawWedge( double angle, int radius, Graphics g ) {

angle -= 0.5 \* Math.PI;

int x = (int)( radius\*Math.cos(angle) );

int y = (int)( radius\*Math.sin(angle) );

angle += 2\*Math.PI/3;

int x2 = (int)( 5\*Math.cos(angle) );

int y2 = (int)( 5\*Math.sin(angle) );

angle += 2\*Math.PI/3;

int x3 = (int)( 5\*Math.cos(angle) );

int y3 = (int)( 5\*Math.sin(angle) );

g.drawLine( width/2+x2, height/2+y2, width/2 + x, height/2 + y );

g.drawLine( width/2+x3, height/2+y3, width/2 + x, height/2 + y );

g.drawLine( width/2+x2, height/2+y2, width/2 + x3, height/2 + y3 );

}

public void paint( Graphics g ) {

g.setColor( Color.gray );

drawWedge( 2\*Math.PI \* hours / 12, width/5, g );

drawWedge( 2\*Math.PI \* minutes / 60, width/3, g );

drawHand( 2\*Math.PI \* seconds / 60, width/2, g );

g.setColor( Color.white );

g.drawString( timeString, 10, height-10 );

}

}

**Analog.html**

<html>

<body>

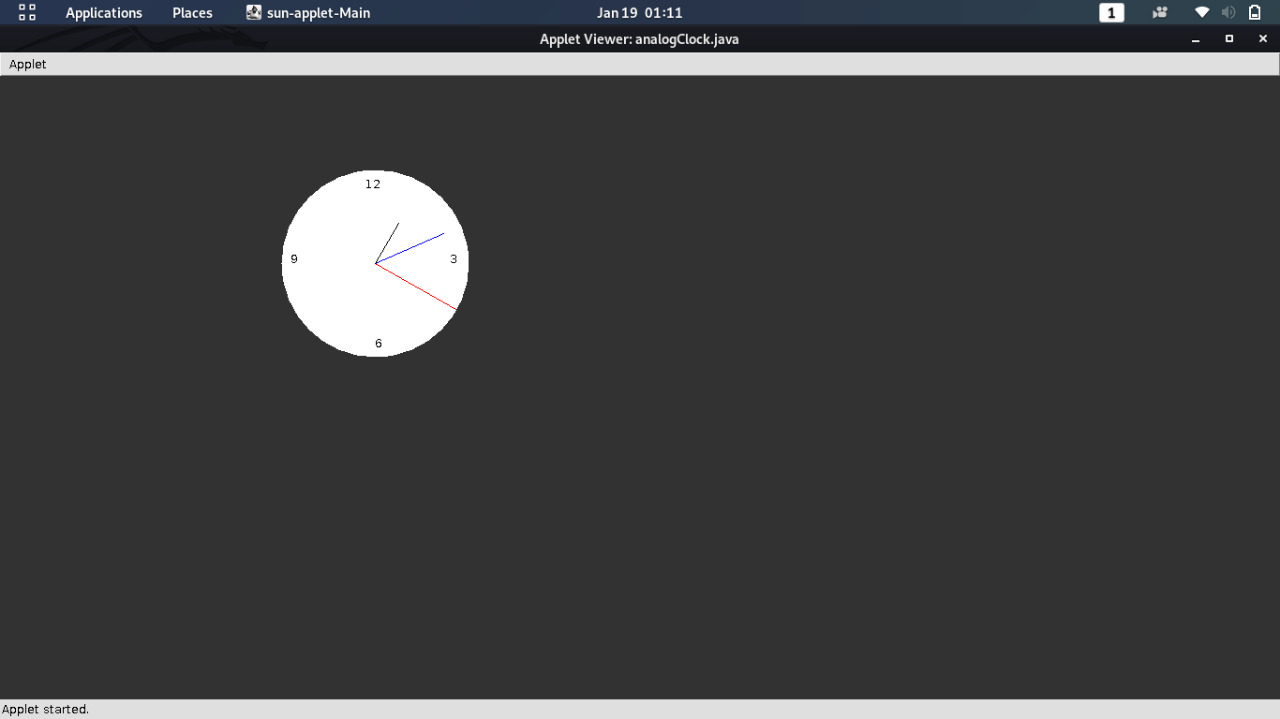
<applet code="analogClock.class" width="300" height="300">

</applet>

</body>

</html>

**OUTPUT:**



**AIM: Write a program of database connectivity using jdbc-odbc driver**

**Code:**

**To create a table in the mysql database:-**

create database sonoo;

use sonoo;

create table emp(id int(10),name varchar(40),age int(3));

**Java Program**

import java.sql.\*;

class MysqlCon{

public static void main(String args[]){

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(

"jdbc:mysql://localhost:3306/java","root","root");

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from emp");

while(rs.next())

System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));

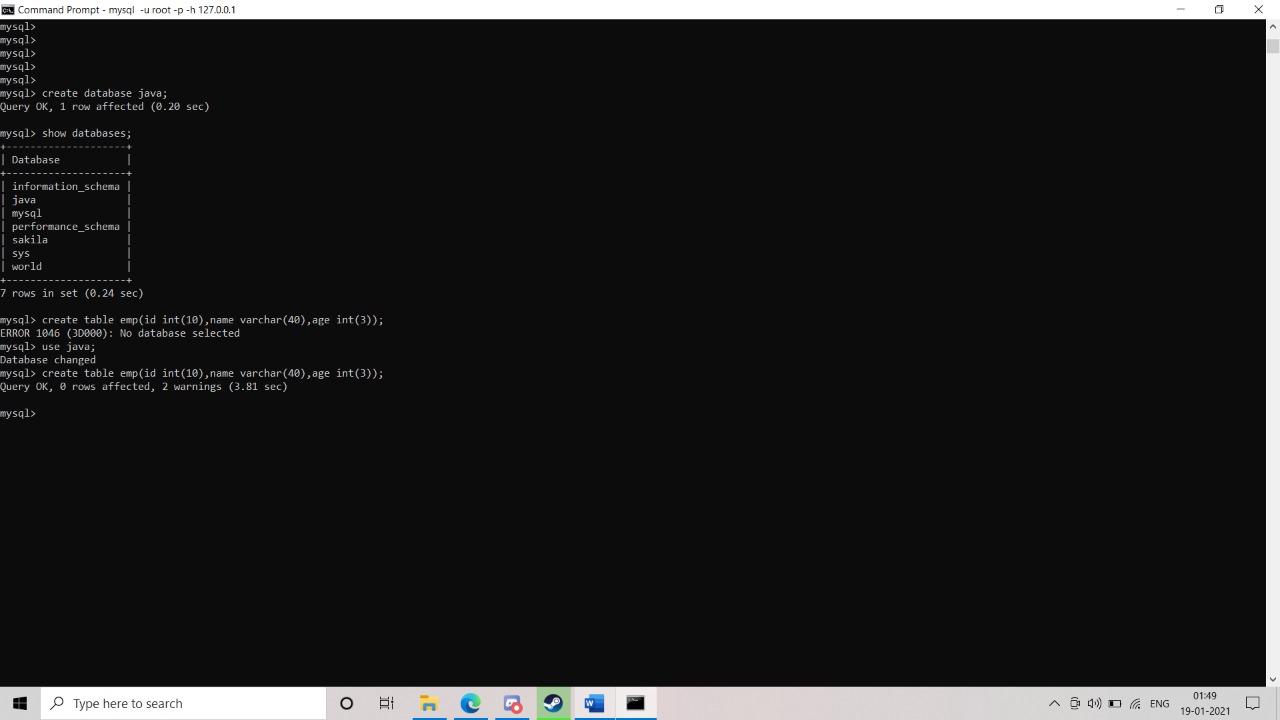
con.close();

}catch(Exception e){ System.out.println(e);}

}

}

**Output:**



**AIM: Write a java program that allows the user to draw lines, rectangles, filled rectangles and rounded rectangles.**

**Code:**

import java.applet.\*;

import java.awt.\*;

public class Shapes extends Applet{

int x=300,y=100,r=50;

public void paint(Graphics g){

g.setColor(Color.green);

g.drawLine(30,300,200,10);

g.setColor(Color.blue);

g.drawRect(400,50,200,100);

g.setColor(Color.red);

g.drawRoundRect(10,10,50,100,10,10);

g.setColor(Color.green);

g.fillRoundRect(200,20,50,100,10,10);

}}

**Myapplet.html**

<html>

<body>

<applet code="Shapes.class" width="300" height="300">

</applet>

</body>

</html>

**OUTPUT:**

