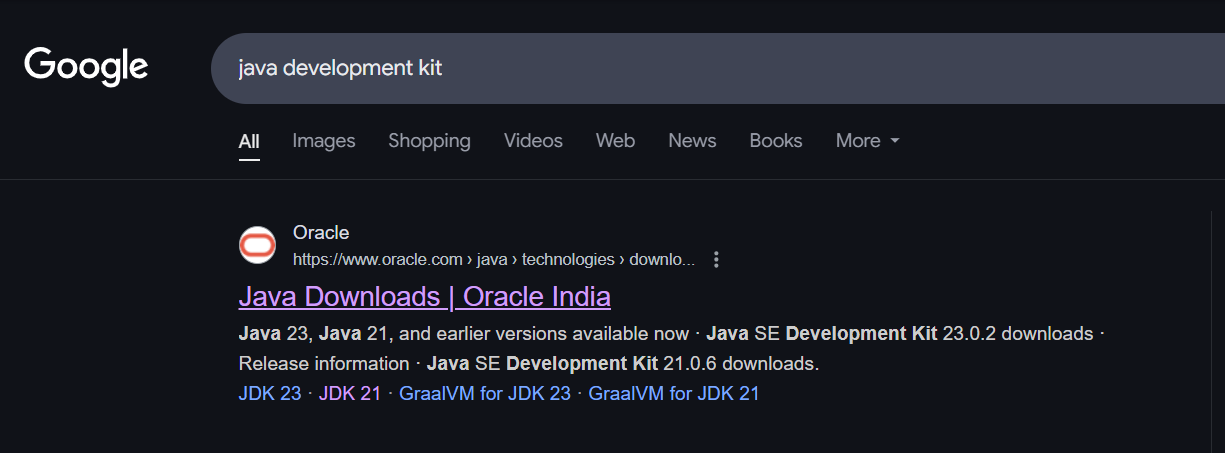
WEEK-1

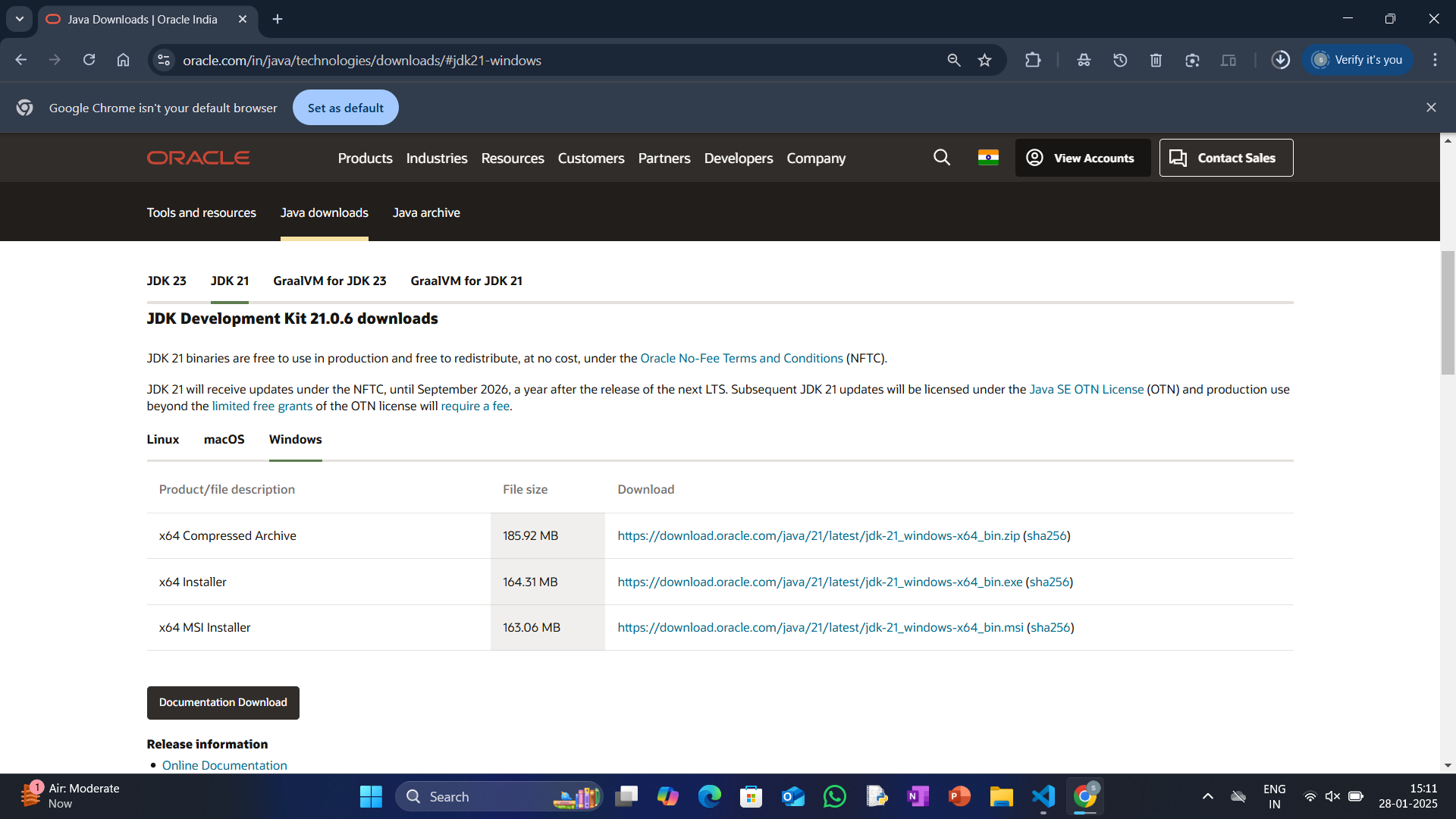
PROGRAME-1

AIM: Download and installation of java

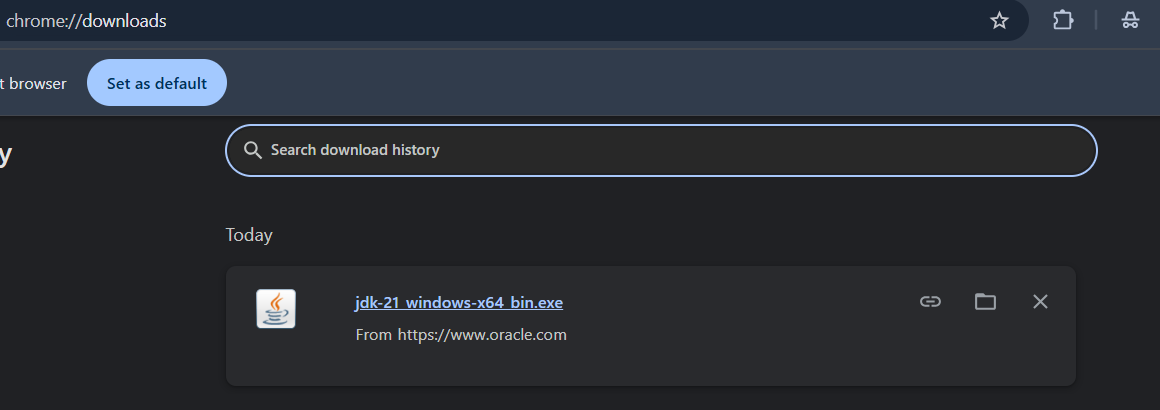
Step 1 : Search for java development kit in chrome to download java



Step 2:Open oracle website. Then select JDK21 and download the type of version for your computer



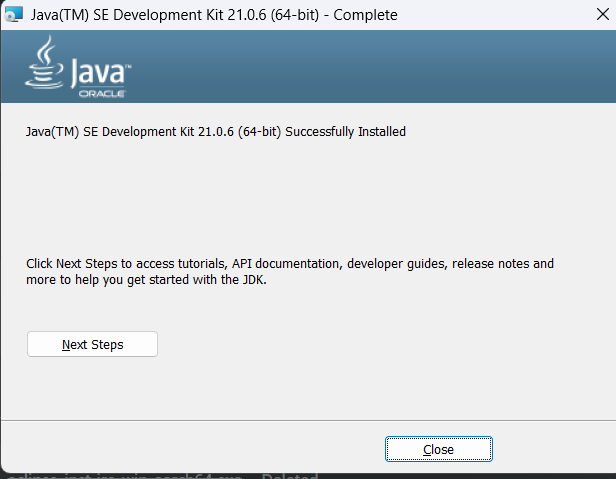
Step 3: after downloading , it will appear like the link below. Click on the link for futher installation of java software.



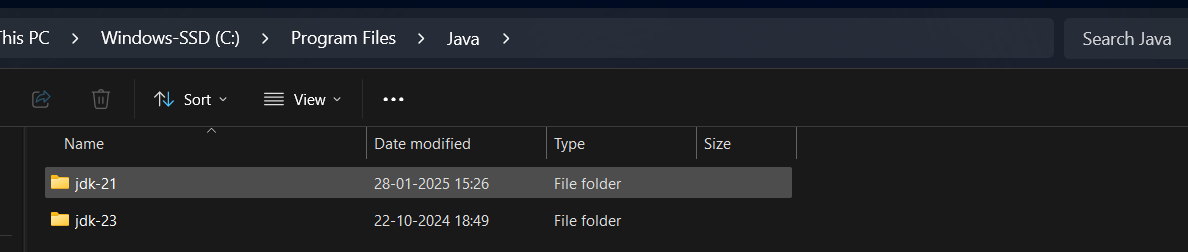
Step 4: click on the next button for futher process of installation of java in computer. At the end section click on next button for final installation.



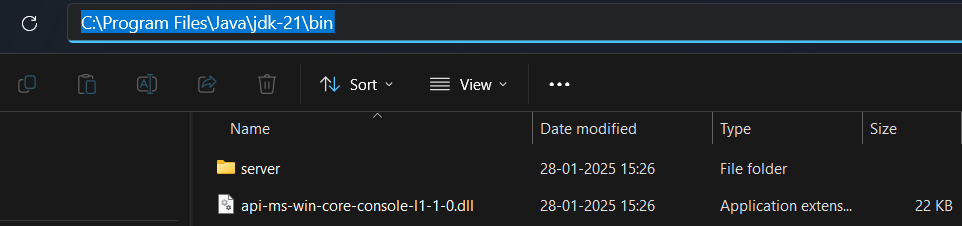
Step 5: at the end section click on the close button to end the installation.



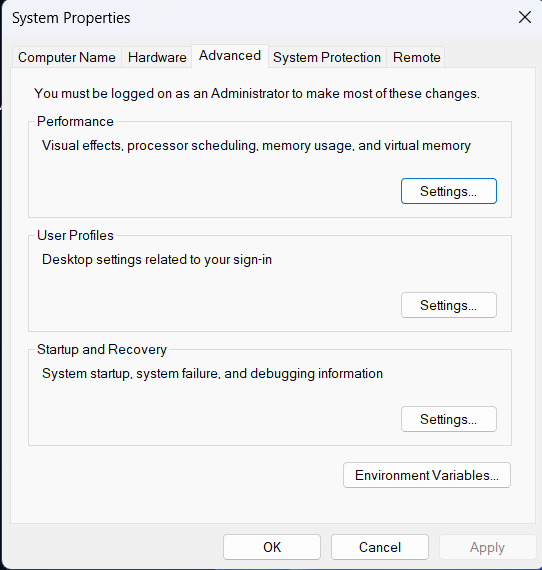
Step6 :to copy the path of the jdk kit in pc go to file manger<< local (c:) <<program files<<java<<jdk 21<<bin .



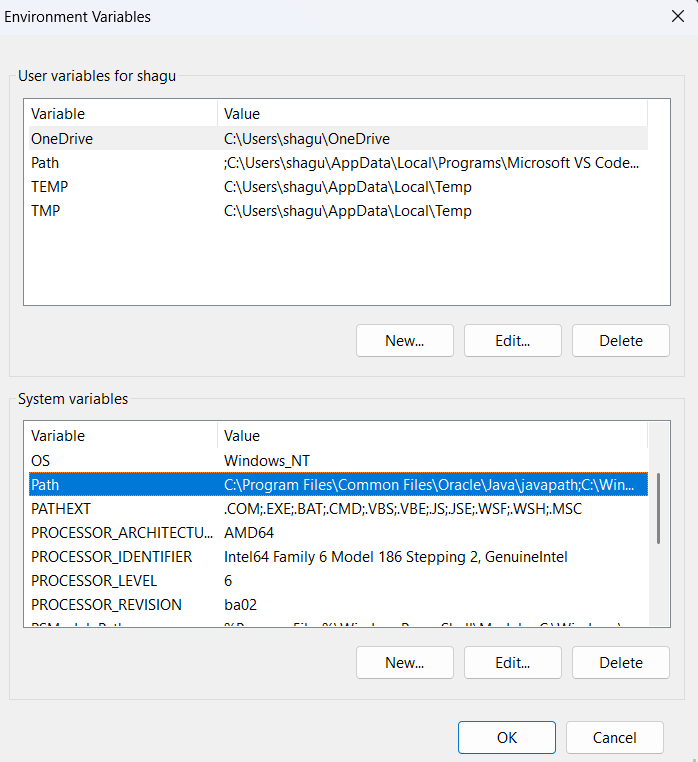
Step 7: copy path on the navigation bar .



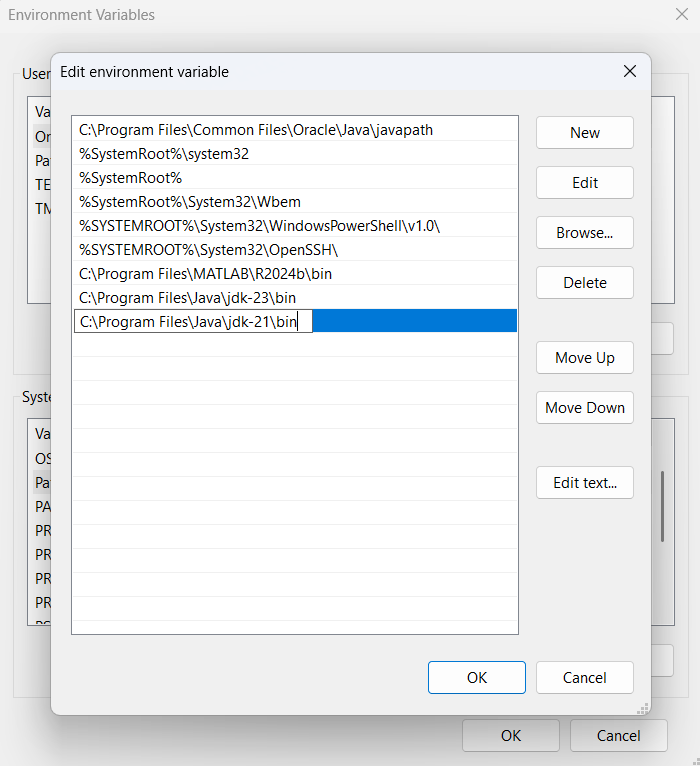
Step 8: now open environmental variables to sset the path in computer.<<click on the environmental variables.



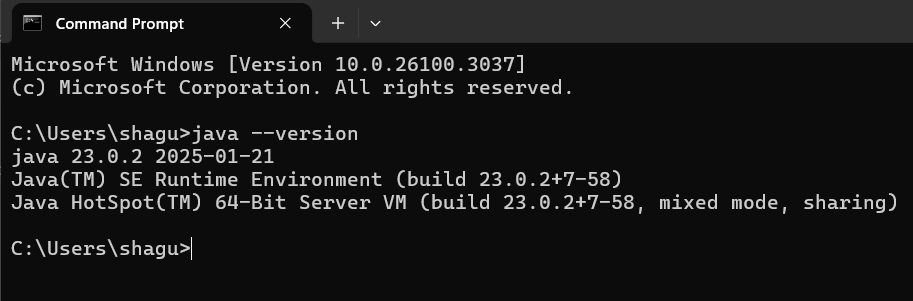
Step 9:after environmental variables another slide will appear of two sections as user variables and system variables<<click on the system variables.<<path<<click on the edit option below .



Step 10:select new << past the path with we have copied on the navigation bar .



Step 11:to check the version installed <<open command prompt<<type java - - version <<enter<<downloaded version will be displayed.



PROGRAM 2:

AIM: To write a java program to print the message

“Welcome programming printing “.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<hello.java

CODE:

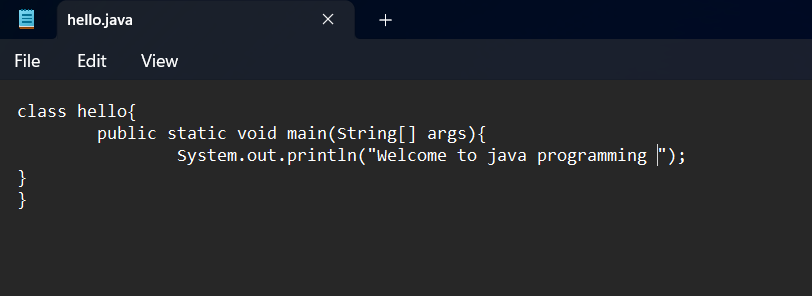
class hello{

public static void main(String [ ] args){

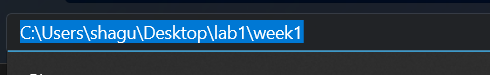
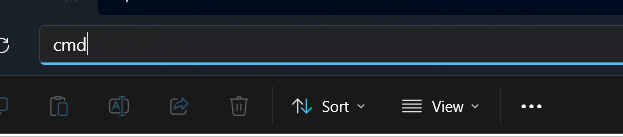
System.out.println(“Welcome to java programming”);

}

}

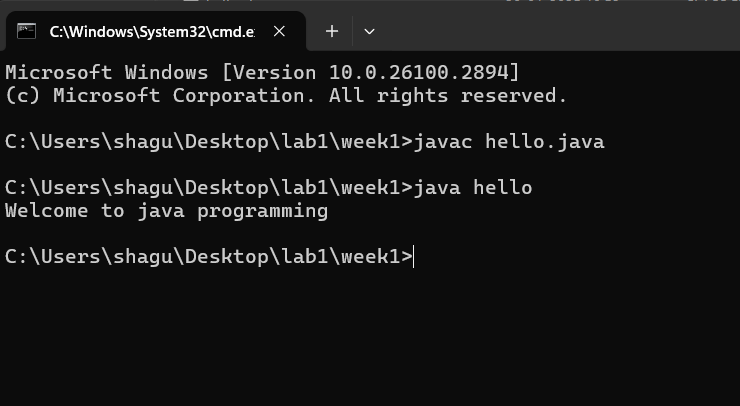


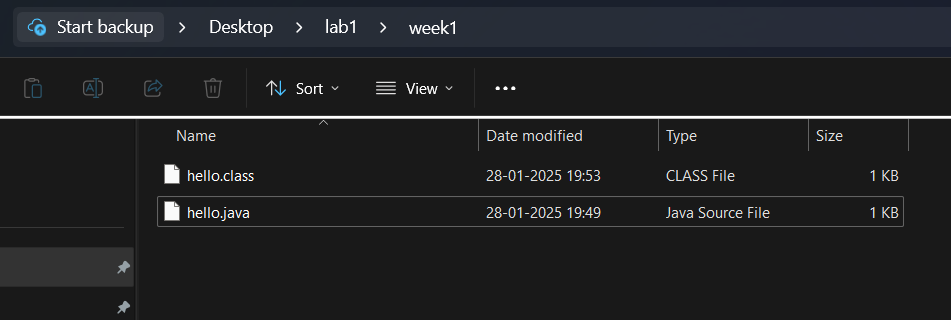
Step 2: to the path clear it and type cmd for running the program.

Step 3:follow commands as: javac hello.java<<enter<<jav hello

The program runs successfully.And creates a java clsss as shown below .





PROGRAME 3:

AIM : To write a java program to print the name,section and roll no .

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<hello.java

CODE:

class hello{

public static void main(String [ ] args){

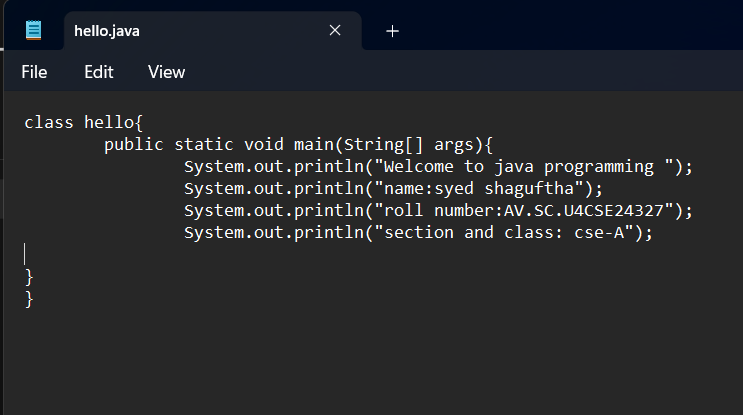
System.out.println(“name : syed ”);

System.out.println(“roll number : AV.SC.U4CSE244444 ”);

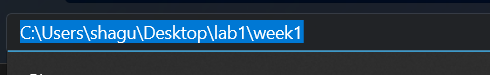
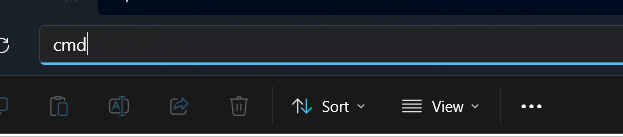
System.out.println(“class and section: CSE -A ”);

}

}

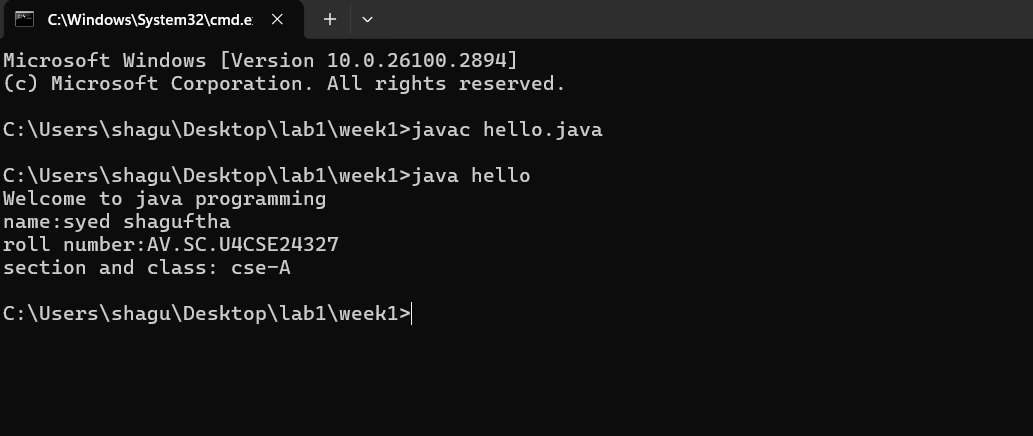


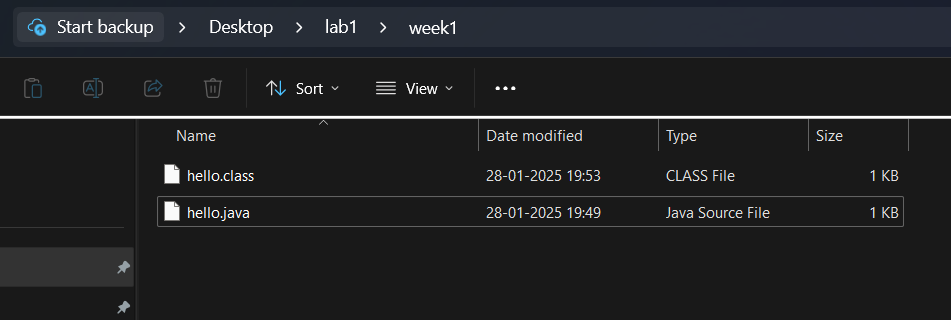
Step 2: to the path clear it and type cmd for running the program.

Step 3:follow commands as: javac hello.java<<enter<<jav hello

The program runs successfully.And creates a java clsss as shown below .





WEEK-2:

PROGRAM 1:

AIM: To write java program to calculate the area of rectangle .

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<exam.java

CODE:

import java.util.Scanner;

class exam{

public static void main(String[] args){

Scanner input= new Scannner(System.in);

System.out.print(“enter the length-l:”);

float l=input.nextFloat();

System.out.print(“enter the length-l:”);

float l=input.nextFloat();

System.out.print(“enter the breadth-b”);

float l=input.nextFloat();

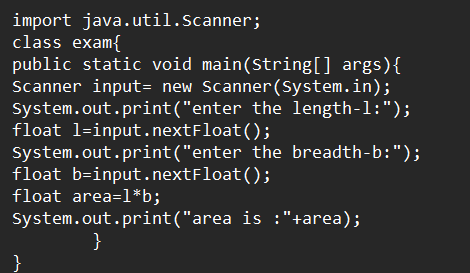
float area=l\*b;

System.out.print(“area is :”+area);

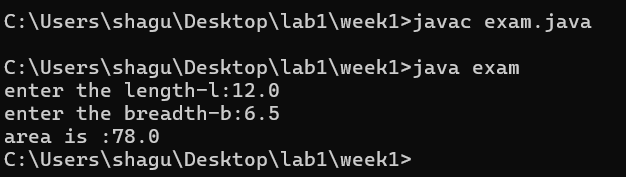
}

}

Step 2:open code in commond prompt and run it.



Step 3:enter the commands as javac exam.java <<java exam.the program is excuted successfully.



Step 4:after runner the program system automatically creates a class for it .

ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MEASSAGE | ERROR RECTIFICATION |
| 1. | Error:”;”expected | Inserted “;”in line7 |
| 2. | Error:”?”unkown sysmbol | Replaced”?”with “:” |

IMPORTANT POINTS:

1. used Scanner library to get input from user in run time .

2. ”import java.util.Scanner;”-step to import library.

3. “Scanner input=new Scanner(System.in);”-step to use the scanner .

**PROGRAM-2:**

1. AIM: To write java program to convert temperature from celcius to farenheit and vice via.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<exam.java

CODE:

//code for celciius to farenheit

import java.util.Scanner;

class exam{

public static void main(String[ ] args){

Scanner input=new Scanner(System.in);

System.out.print(“Enter the celcius :”);

float c:input.nextFloat();

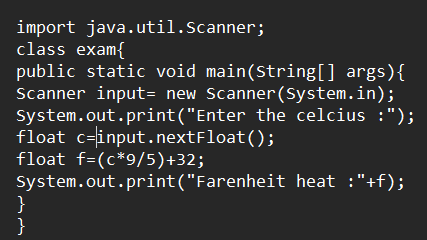
float f=(c\*9/5)+32;

System.out.print(“Farenheit heat :”+f);

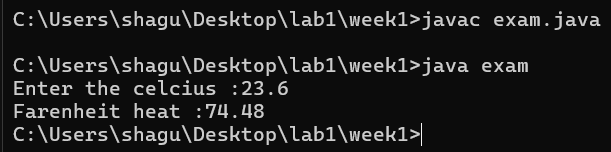
}

}

Step 2:open in commond prompt and run it.



Step 3: enter the commands as javac exam <<java exam the program is excuted successfully.



Step 4:After runner the program system automatically creates a class of it.

ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error :”:” unknow symbol | Replace”:” with”=” |
| 2. | Error: “scanner”small letter case censitive | “Scanner” |

IMPORTANT POINTS:

1.used Scanner library to get input from user in run time.

2.”import java.util.Scanner;”-step to import library.

3.”Scanner input= new Scanner(System.in);”-step to use the scanner.

**PROGRAM**

1. AIM:To write java program to convert temperature from farenheit to celceius.

CODE:

//code for farenheit to celciius

import java.util.Scanner;

class exam{

public static void main(String[ ] args){

Scanner input=new Scanner(System.in);

System.out.print(“Enter the farenheit :”);

float f:input.nextFloat();

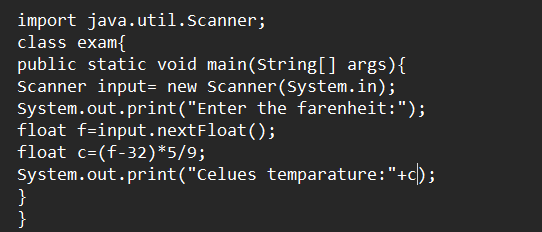
float c=(f-32)\*5/9;

System.out.print(“celcius temparature :”+c);

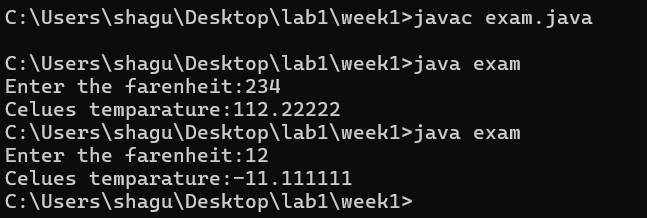
}

}

Step 2:open in commond prompt and run it.



Step 3: enter the commands as javac exam <<java exam the program is excuted successfully.



Step 4:After runner the program system automatically creates a class of it.

ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error :”oout” unknow keywoard | Replace”oout” with”out” |
| 2. | Error: “scanner”small letter case censitive | “Scanner” |

IMPORTANT POINTS:

1.used Scanner library to get input from user in run time.

2.”import java.util.Scanner;”-step to import library.

3.”Scanner input= new Scanner(System.in);”-step to use the scanner.

**PROGRAM**

1. AIM: To write java program to calculate the simple

CODE:

Import java.util.Scanner;

class exam{

public static void main(String[ ] args){

Scanner input=new Scanner(System.in);

System.out.print(“enter the principle value(p):”);

float p=input.nextFloat();

System.out.print(“enter the rate of interest value(r):”);

float r=input.nextFloat();

System.out.print(“enter the time period value(t):”);

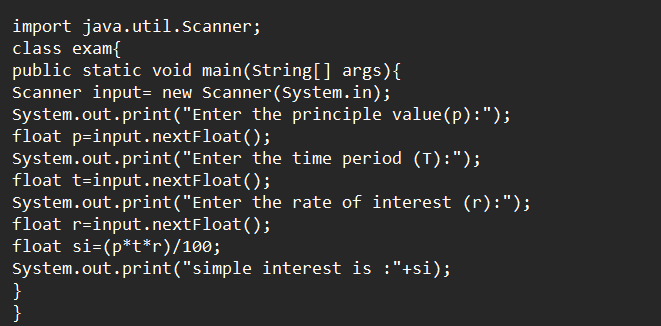
float t=input.nextFloat();

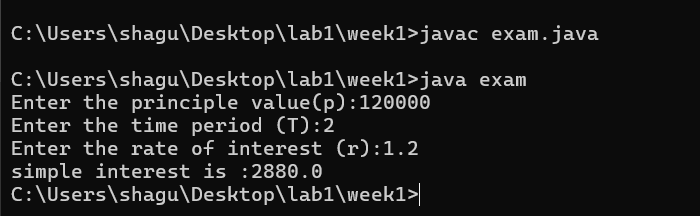
float si=(p\*t\*r)/100;

System.out.println(“simple interest is:”+si);

}

}





ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error :”T” is not declred | Replace:”T” with”t” |
| 2. | Error : expected’;’ in line 8 | Insert ‘;’ in line 8 end |

IMPORTANT POINTS:

1.java is a case sensitive language so “apple” is different from “APPLE”,so clear declaration of variables is important..

1. AIM: To write a program to find the largest of three numbers using ternary operators.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<exam.java

CODE:

import java.util.Scanner;

class exam{

public static void main(String [ ] args){

Scanner input=new Scanner(System.in);

System.out.print(“enter n1:”);

int n1=input.nextInt();

System.out.print(“enter n2:”);

int n2=input.nextInt();

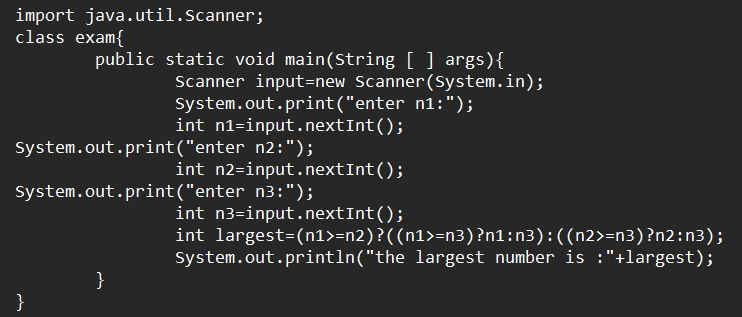
System.out.print(“enter n3:”);

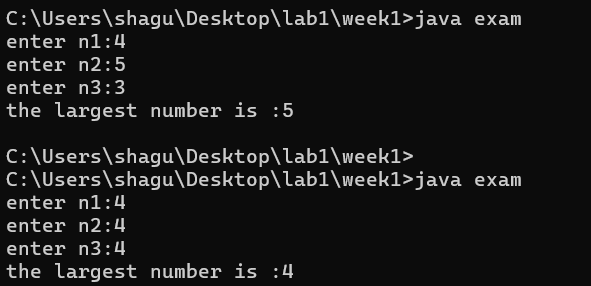
int n3=input.nextInt();

int largest=(n1>=n2)?((n1>=n3)?n1:n3):((n2>=n3)?n2:n3);

System.out.println(“the lsrgest number is :”+lsrgest);

}

}



ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error :”;” expected in line 6 | Insert”;” in end of line 6 |
|  | Error :”nextint();” non identified | Replace”next.Int();” |

IMPORTANT POINTS:

1. Ternary operators: a shorthand for the if-else statement, used to execute condition-based operations in a single line.
2. It evaluates a Boolean condition and returns trueValue if the condition is true, otherwise it returns falseValue.
3. AIM: To write a program for the factorial of the numbers.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<exam.java

CODE:

import java.util.Scanner;

class exam{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

System.out.println("fibinocci series");

System.out.println("enter a number:");

int n =input.nextInt();

int f1=0,f2=1;<br>

System.out.println(" "+f1);

System.out.println(" "+f2);

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

int f3=f1+f2;

System.out.println(" "+f3);

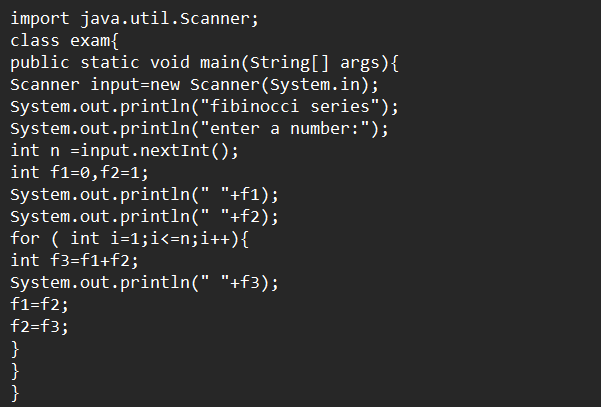
f1=f2;

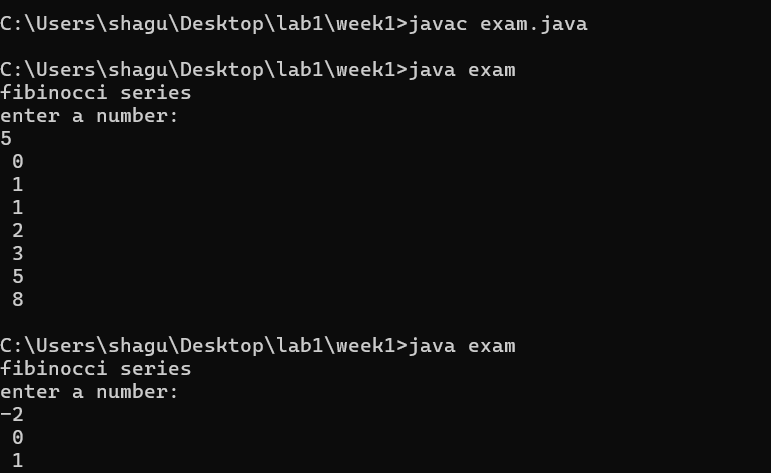
f2=f3;

}

}

}





ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error: line-9 illegal start of expression | Rebuilt of the line -9 |
|  | Error :iteration error | Correct iteration inserted |

IMPORTANT POINTS:

1.Java for loop is a control flow statement that allows code to be executed repeatedly based on a given condition.

2.The for loop in java provides an efficient way to iterate over a range of values ,execute code multiple times,or traverse arrays and collections.

WEEK-3

PROGRAME-1

AIM: To write a program for car color and all respective complextions using constructor and method.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<car.java

CODE:

class car{

//creating the attributes requires for the classs

String car\_name,car\_color,car\_brand,fule\_type;

int maleage;

//constructor

car(String car\_name,String car\_color,String car\_brand,String fule\_type,int maleage){

this.car\_name=car\_name;

this.car\_color=car\_color;

this.car\_brand=car\_brand;

this.fule\_type=fule\_type;

this.maleage=maleage;

}

//creating the methods forte class

public void start(){

System.out.println("this is start statement: "+car\_name+" "+car\_color);

}

public void stop(){

System.out.println("this is start statement: "+car\_brand+" "+fule\_type);

}

public void services(){

System.out.println("this is start statement: "+maleage);

}

public static void main(String[] args){

//creating the object for the class

car car1=new car("maruthi","navy blue","KIA","petrol", 1234);

car1.start();

car car2=new car("maruthi","navy blue","KIA","petrol", 1234);

car2.stop();

car car3=new car("maruthi","navy blue","KIA","petrol", 1234);

car3.services();

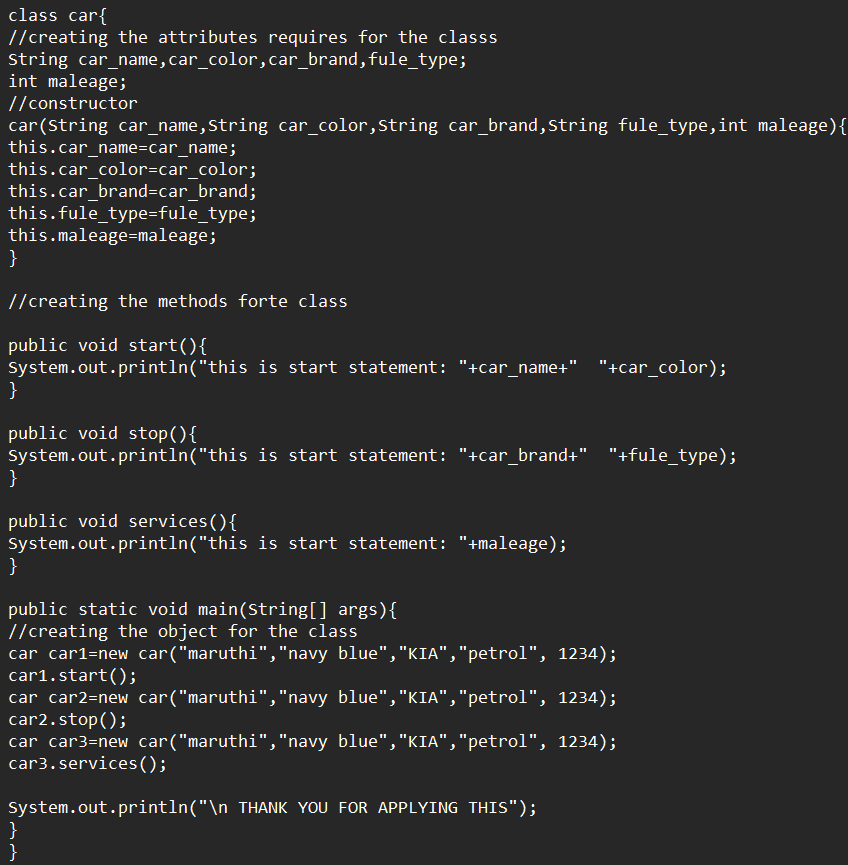
System.out.println("\n THANK YOU FOR APPLYING THIS");

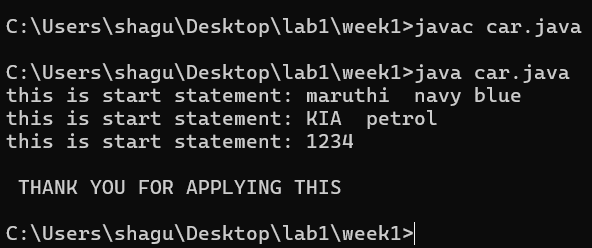
}

}

CLASS DIAGRAM:

|  |
| --- |
| Car() |
| +car\_name:string  +car\_color:string  +car\_brand:string  +fule\_type: int  +maleage:int |
| +start:void()  +stop:void()  +static:void() |





ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error: line7 expected ‘; | Inserted ‘;’ |
|  | Error :line 7 unknow’\_\_’ | Removed ‘\_’ |
|  | Error : correct data type declararion in constructor | Rectified by declaring the data type as String and int. |

IMPORTANT POINTS:

1. Java constructor is used to save the variables present in different or same class or methods.

2. In Java, the this keyword refers to the current instance of a class. It is commonly used to distinguish between instance variables and parameters with the same name, or to refer to the current object from within a method or constructor.

3. In Java, a method is a block of code that performs a specific task and can be invoked to execute that task. It typically consists of a method signature (name, return type, and parameters) and the body of the method, which contains the logic.

**PROGRAM**

AIM: To write a program for car color and all respective complextions using constructor and method.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<BANK.java

CODE:

import java.util.Scanner;

class BankAccount {

 // Class-level variable to store balance

    private float existing;

    private Scanner input; // Single Scanner instance for input

    public  String name;

    // Constructor

    public BankAccount() {

        input = new Scanner(System.in);

        System.out.println("Enter the account holder name :");

        this.name=input.next();

        System.out.print("Enter existing amount in bank account: ");

        this.existing = input.nextFloat();

    }

    // Deposit method

    public void deposit() {

        System.out.print("Enter amount to be deposited: ");

        float deposit = input.nextFloat();

        existing += deposit;

        System.out.println("Existing amount now is: " + existing);

    }

    // Withdrawal method

    public void withdrawal() {

        System.out.print("Enter amount to be withdrawn: ");

        float withdrawal = input.nextFloat();

        if (existing < withdrawal) {

            System.out.println("Not sufficient balance.");

        } else {

            existing -= withdrawal;

            System.out.println("Remaining balance: " + existing);

        }

    }

    // Main method

    public static void main(String[] args) {

        BankAccount customer1 = new BankAccount();

        customer1.deposit();

        customer1.withdrawal();

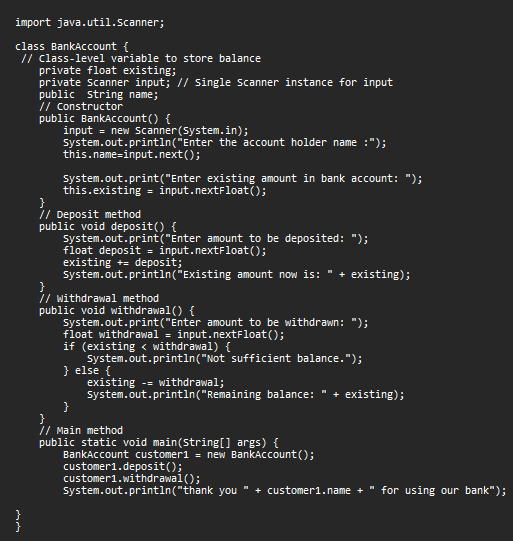
        System.out.println("thank you " + customer1.name + " for using our bank");

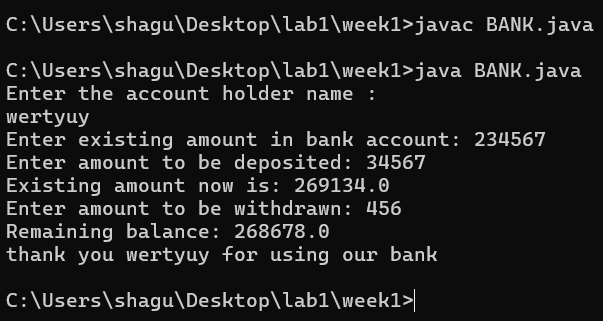
}

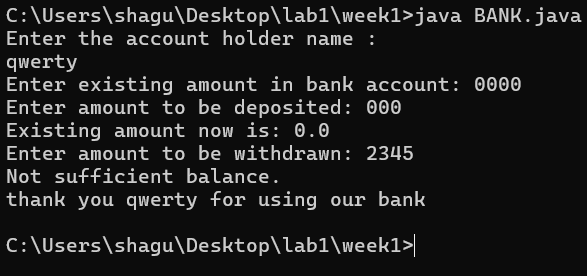
}

CLASS DIAGRAM:

|  |
| --- |
| BankAccount |
| -existing:float  +name:String |
| +BankAccount()  +deposit:void()  +withdraw:void() |







ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error: nextString(); wrong identifier | Rectification: next(); |
|  | Error :line 7 unknow’\_\_’ | Removed ‘\_’ |
|  | Error : if statement ‘{}’ expected | Inserted ‘{}’ |

IMPORTANT POINTS:

1. Java constructor is used to save the variables present in different or same class or methods.

2. In Java, the this keyword refers to the current instance of a class. It is commonly used to distinguish between instance variables and parameters with the same name, or to refer to the current object from within a method or constructor.

3. In Java, a method is a block of code that performs a specific task and can be invoked to execute that task. It typically consists of a method signature (name, return type, and parameters) and the body of the method, which contains the logic.

WEEK-4

PROGRAME-1

AIM: To write a program for printing the title of the book and the author and year of publication using the constructors

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<person.java

CODE:

class book{

//creating the variable

public String title\_of\_book;

public String author;

public int year\_publication;

//creating a constructor

book(String title\_of\_book,String author,int year\_publication){

this.title\_of\_book=title\_of\_book;

this.author=author;

this.year\_publication=year\_publication;

}

//creating the method to print DETAILS

public void details(){

System.out.println("the title of the book is: "+title\_of\_book+"\nThe author of te book is: "+author+"\nthe year of publication is:"+year\_publication+"\n");

}

//creating the main class and objects for the method

public static void main(String[] args){

book one=new book("THE GREAT INDIAN RIVERS","DR.SHIVARAM",1989);

one.details();

book two=new book("ANGLES IN TIBET","S.SLUMP",2001);

two.details();

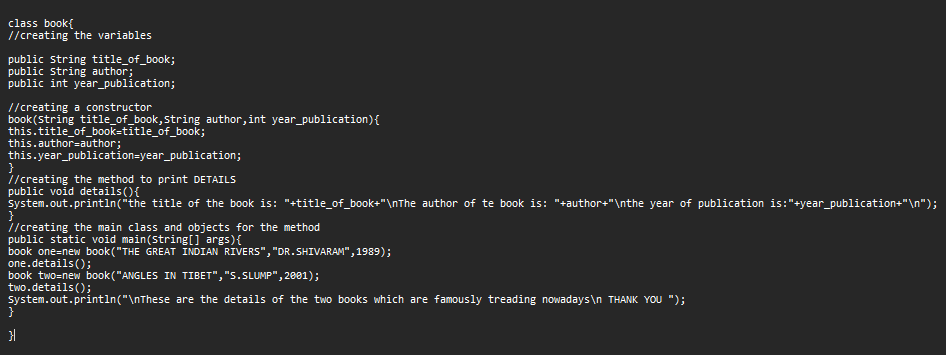
System.out.println("\nThese are the details of the two books which are famously treading nowadays\n THANK YOU ");

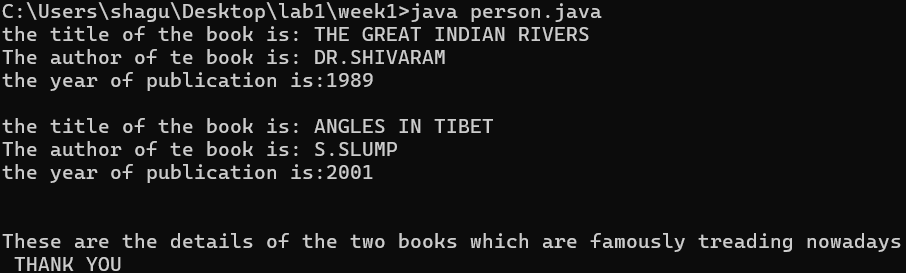
}

}

CLASS DIAGRAM:

|  |
| --- |
| Book |
| +title\_of\_book:string  +author:string  +year\_publication:int |
| +book()  +detailes:void() |





ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error: “ this.year\_public;=year\_public;” | Rectification: removed the ‘;’ |
| 2. | Error :”missing ‘;’-“System.out.println(“..”); | Inserted the ‘;’ in the line. |

IMPORTANT POINTS:

1. Java constructor is used to save the variables present in different or same class or methods.
2. In Java, the this keyword refers to the current instance of a class. It is commonly used to distinguish between instance variables and parameters with the same name, or to refer to the current object from within a method or constructor.

PROGRAME-2s

AIM: To write a program for printing the title of the book and the author and year of publication using the constructors

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<exam.java

CODE:

class myclass{

//creating the variables

static int count=0;

final double pi=3.1415;

//creating a constructor

myclass(){

count++;// creatinfg the condition for the increment of the static count variable

}

//method to print the values

public void values(){

System.out.println(+count);

System.out.println(+pi);

}

//object and the main function

public static void main(String[] args){

//creating the four objects to check the code for the condition of constructor

myclass one=new myclass();

one.values();

myclass two=new myclass();

two.values();

myclass three=new myclass();

three.values();

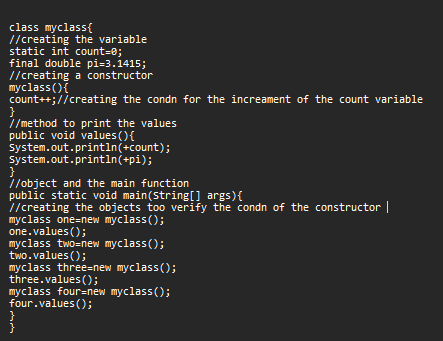
myclass four=new myclass();

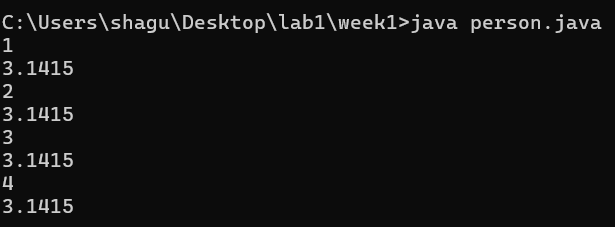
four.values();

}

} CLASS DUAGRAM:

|  |
| --- |
| Myclass |
| -count:0  -pi:3.1415 |
| +myclass()  +values:void() |





ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error: argument required of type int | Rectification: rectified the argument issue. |
|  | Error :line 7 unknow’\_\_’ | Removed ‘\_’ |
|  | Error : if statement ‘;’ expected | Inserted:count++; |

IMPORTANT POINTS:

1. Java constructor is used to save the variables present in different or same class or methods.
2. In Java, the ++ operator increments a variable by 1, either as **pre-increment** (++x) or **post-increment** (x++).
3. In Java:
4. **static**: A static variable belongs to the class, not instances, meaning all objects share the same value.
5. **final**: A final variable cannot be modified once assigned, making it constant.

WEEK-5

PROGRAME-1

AIM: create a calculator using the operations including add,sub,multi and div using multilevel inheritanceand display the desipred output

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<inheritance.java

CODE:

import java.util.Scanner;

class Calculator {

// Base class for the calculator

Calculator(){

System.out.println("\nthis is the calculator program\n");

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

} }

class Simple extends Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

} }

class Super extends Simple {

public int square(int a) {

return a \* a;

}

public int cube(int a) {

return a \* a \* a;

}

public double squareRoot(int a) {

return Math.sqrt(a);

}

}

class Advanced extends Super {

public double divide(int a, int b) {

if (b != 0) {

return (double) a / b;

} else {

return 0; // Division by zero is not allowed.

}

}

public int modulus(int a, int b) {

return a % b;

}

}

public class inherit {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

System.out.println("enter a value:");

int a=input.nextInt();

System.out.println("enter b value: ");

int b=input.nextInt();

Simple simpleCalc = new Simple();

System.out.println("Addition: " + simpleCalc.add(a, b));

System.out.println("Subtraction: " + simpleCalc.subtract(a, b));

System.out.println("Multiplication: " + simpleCalc.multiply(a, b));

Advanced advancedCalc = new Advanced();

System.out.println("Division: " + advancedCalc.divide(a, b));

System.out.println("Modulus: " + advancedCalc.modulus(a, b));

Super superCalc = new Super();

System.out.println("Square: " + superCalc.square(a));

System.out.println("Cube: " + superCalc.cube(b))

System.out.println("Square Root: " + superCalc.squareRoot(b));

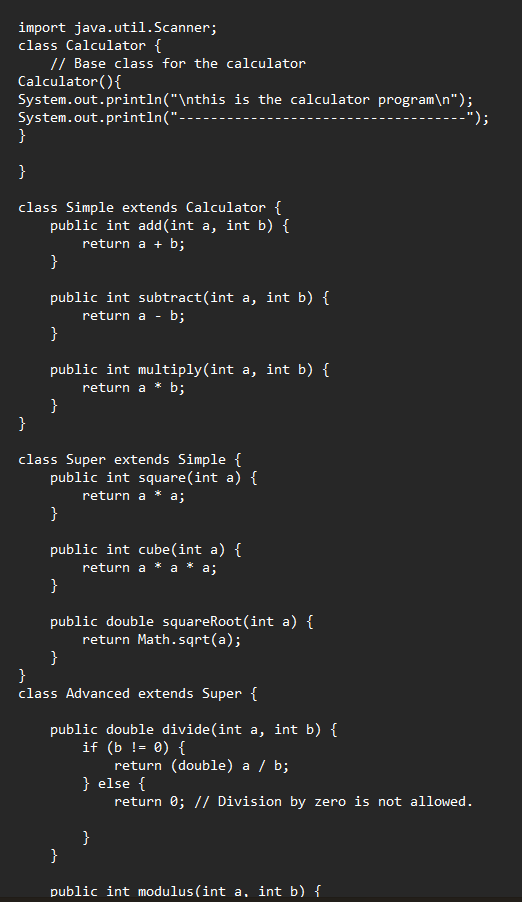
} }

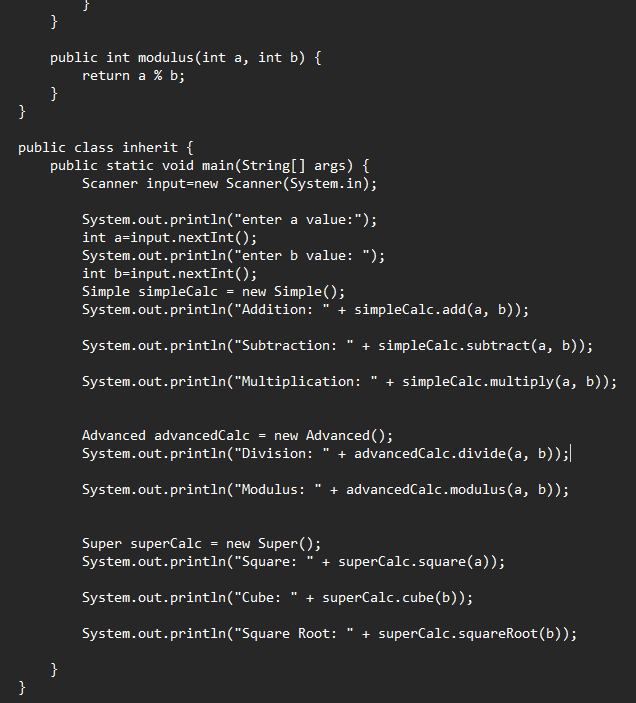
|  |
| --- |
| Calculator |
| +Calculator() |

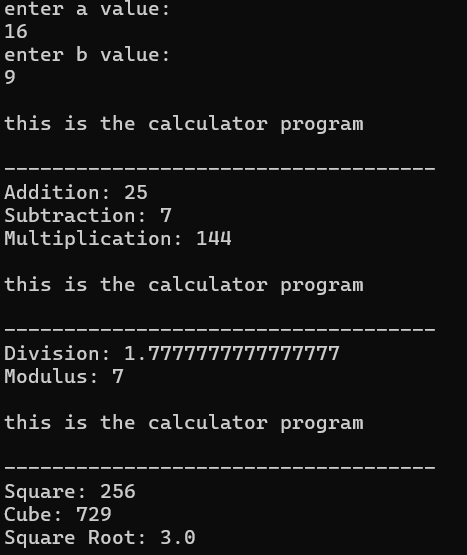
|  |
| --- |
| Siimple |
| +add(int):return int  +substract(int):return int  +multiply(int):return int |

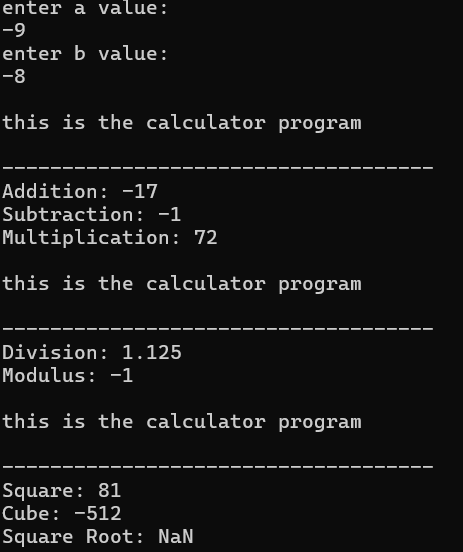
|  |
| --- |
| Super |
| +square(double):return double  +cube(int):return int  +squareRoot(double):double |

|  |
| --- |
| Advanced |
| +divide(double):return double  +module(int):return int |









ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
|  | Error: mutipile inheritance in the Advanved class | Implemented Advanced class from Super class. |
|  | Error : Scanner; | Scanner(System.in); |

IMPORTANT POINTS:

1. Multiple inheritance lets a class inherit from multiple parents, combining their features, but can cause issues like the diamond problem, resolved by MRO.
2. Math.sqrt() in Java calculates the square root of a non-negative double value and returns a double result, or NaN if the input is negative.
3. The import java.util.Scanner; statement in Java allows you to use the Scanner class from the java.util package, which is commonly used to read user input from the console.

PROGRAME-2

AIM: create a java program of a vehicle entry company hireachical wants to develop his system that maintains information about different types of cars and bikes and they need a program to store details about each vehicle auch as brand and speed

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<inheritance.java

CODE:

class Vehicle{

String brand;

int speed;

Vehicle(String brand,int speed){

this.brand=brand;

this.speed=speed;

}

void Details(){

System.out.println("Brand:"+brand);

System.out.println("\nSpeed:"+speed);

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

}

}//End of super class

class CARS extends Vehicle{

int doors;

int capacity;

public CARS(String brand,int speed,int doors,int capacity){

super(brand, speed);

this.doors=doors;

this.capacity=capacity;

}

void cardetails(){

System.out.println("\nNumber of doors:"+doors);

System.out.println("\nCapacity:"+capacity);

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

}

}//End of car sub-class

class Bikes extends Vehicle{

Boolean gears;

Bikes(String brand,int speed,Boolean gears){

super(brand, speed);

this.gears=gears;

}

void bikedetails(){

if (gears==true) {

System.out.println("This bike has gears.");

}

else{

System.out.println("This bike does not have gear system.");

}

}

}//End of bike sub-class

class Trucks extends Vehicle{

int tons;

Trucks(String brand,int speed,int tons){

super(brand, speed);

this.tons=tons;

}

void truckdetails(){

System.out.println("The capacity of truck is: "+tons);

}

}//End of truck sub-class

class inherit{

public static void main(String[] args){

CARS c=new CARS("Tayota",120,5,2);

c.cardetails();

c.Details();

Bikes b=new Bikes("KTM",80,true);

b.bikedetails();

b.Details();

Trucks t=new Trucks("TATA",150,1);

t.truckdetails();

t.Details();

System.out.println("THANK YOU FOR COMING TO OUR COMPANY :) ~ ^ !");

}

}

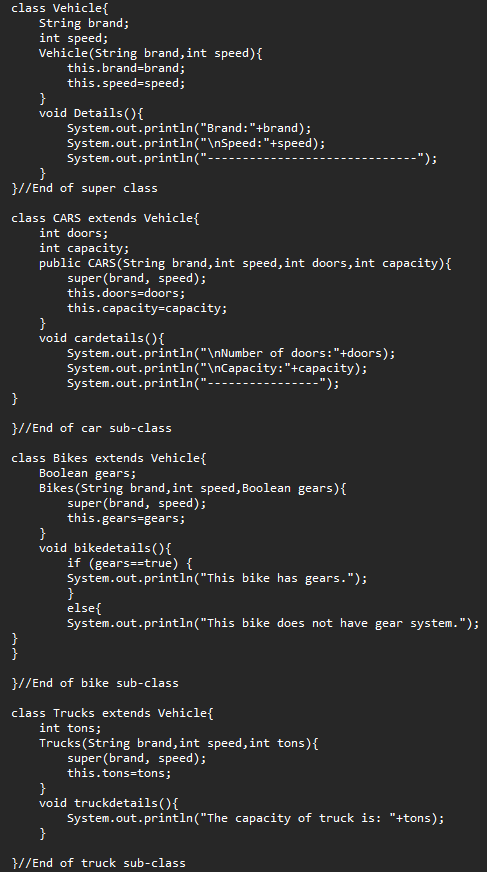
CLASS DIAGRAMS :

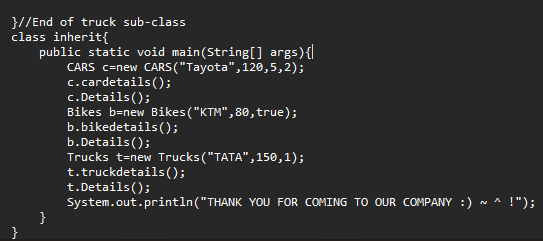
|  |
| --- |
| Vehicle |
| +brand:String  +speed:int |
| +Vehicle();  +Details():void |

|  |
| --- |
| CARS |
| +doors:int  +capacity:int |
| +CARS()  +cardetails():void |

|  |
| --- |
| Bikes |
| +gears:Boolean |
| +Bikes()  +bikedetails():void |

|  |
| --- |
| Trucks |
| +tons:int |
| +Trucks()  +truckdetails():void |







ERRORS:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error: Incorrect Constructor Arguments. | the arguments passed when creating an object match the constructor's parameter list in both **number** and **type**. |
| 2. | Error : Scanner; | Scanner(System.in); |

IMPORTANT POINTS:

1. Hierarchical inheritance is a type of inheritance where multiple subclasses inherit from a single parent class, allowing code reuse and reducing redundancy.
2. A **constructor** is a special method in a class used to initialize new objects with default or provided values. It is automatically called when an object is created and sets up the object's initial state.

WEEK-6

PROGRAME-1

AIM: To write a program for creating a shape with method calculatearea, that is overloaded for different shapes ,create a subclass circle that over riding the calculatearea method for a circle.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:

import java.util.Scanner;

class Shape{

    float s=3;

    int l=5,b=6;

    double r=2.3;

    public void calculatarea(float s){

        System.out.print("area of a square is:");

        double area=s\*s;

        System.out.println(area);

    }

    public void calculatarea(int l,int b){

        System.out.print("area of a rectangle is:");

        double area=l\*b;

        System.out.println(area);

    }

    public void calculatarea(double r){

        System.out.println("over riding method");

        System.out.println("radius is:"+r);

}

}

class Circle extends Shape{

    public void calculatarea(double r){

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

        System.out.print("area of a circle is:");

        double area=3.14\*r\*r;

        System.out.println(area);

}

}

class week{

    public static void main(String[] args) {

        Shape s1=new Shape();

        s1.calculatarea(3);

        s1.calculatarea(5,6);

        s1.calculatarea(2.3);

        Circle c1=new Circle();

        c1.calculatarea(3.4);

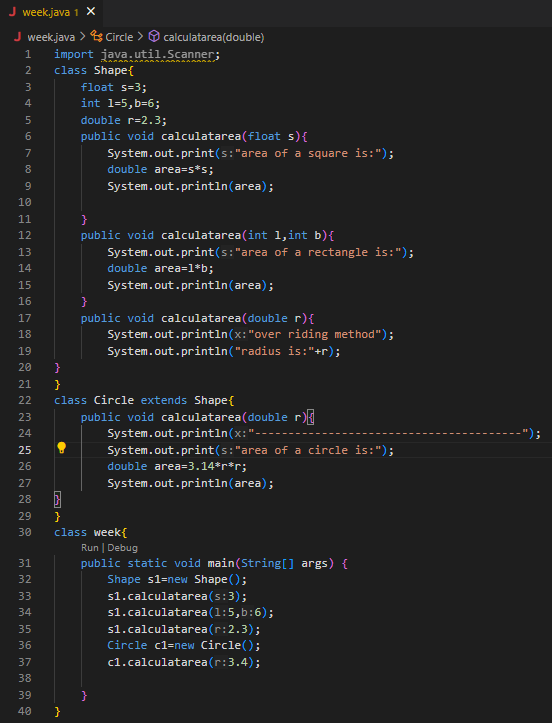
    }

}

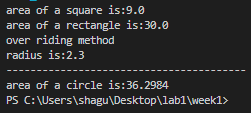
**CLASS DIAGRAMS :**

|  |
| --- |
| **Shape** |
| **+a:float;**  **+l,b:int;**  **+r:double;** |
| **+calculatearea(float s);**  **+calculatearea(int l, int b);**  **+calculatearea(double r);** |

|  |
| --- |
| **Circle** |
| **+calculatearea(double r);** |



**OUTPUT:**



**ERROR:**

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1. | Error:’;’ is missing in print statemnt | Rectification:inserted the’;’ in print statement |

IMPORTANT POINTS:

 Method overriding allows a subclass to provide a new version of a method already defined in its parent class.

 The method name, parameters, and return type must be the same.

 It is used to change or extend the behavior of inherited methods.

 Overriding supports runtime polymorphism (decides which method to run at runtime).

 It helps in writing flexible, reusable, and organized code

PROGRAME-2

AIM: To create calculator with over load method to perform overload i)add two integers ii)add two double iii)add three int

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:

class Calculator {

// Method 1: Add two integers

int add(int a, int b) {

return a + b;

}

// Method 2: Add two doubles

double add(double a, double b) {

return a + b;

}

// Method 3: Add three integers

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

return a + b + c;

}

}

public class q3 {

public static void main(String[] args) {

Calculator calc = new Calculator();

// Test the overloaded methods

int sum1 = calc.add(10, 20);

double sum2 = calc.add(5.5, 6.7);

int sum3 = calc.add(1, 2, 3);

// Display results

System.out.println("Sum of two integers: " + sum1);

System.out.println("Sum of two doubles: " + sum2);

System.out.println("Sum of three integers: " + sum3);

}

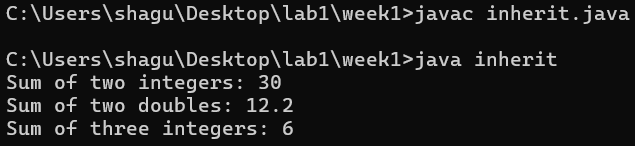
}

CLASS DIAGRAMS:

|  |
| --- |
| Calculator |
|  |
| + add(a: int, b: int): int  + add(a: double, b: double): double  + add(a: int, b: int, c: int): int |

|  |
| --- |
| Inherit |
|  |
| +main:void |

OUTPUT:;



ERROR:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1. | Initially file name stored in another name | Rectified by changing name to the stored one. |

IMPOERTANT POINTS:

1. Method overloading means defining **multiple methods with the same name** but **different parameters**.
2. It happens **within the same class**.
3. The methods must differ in **number or type of parameters**.
4. It increases **code readability** and **flexibility**.
5. Overloading is decided at **compile-time** (compile-time polymorphism).

PROGRAME-3

AIM: To create a collage program for developing automated admission that verifies students eligibility for UG,PG programs.each program has different eligibility criteria based on the students percentage[%] in their perious qualification.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE: class AdmissionSystem {

void checkEligibility(String name, double percentage, String programType) {

if (programType.equals("UG")) {

if (percentage >= 60) {

System.out.println(name + " is eligible for Undergraduate program.");

} else {

System.out.println(name + " is NOT eligible for Undergraduate program.");

}

} else if (programType.equals("PG")) {

if (percentage >= 70) {

System.out.println(name + " is eligible for Postgraduate program.");

} else {

System.out.println(name + " is NOT eligible for Postgraduate program.");

}

} else {

System.out.println("Invalid program type: " + programType);

}

}

}

public class inherit {

public static void main(String[] args) {

AdmissionSystem admission = new AdmissionSystem();

// Test cases with hardcoded values

admission.checkEligibility("Ram", 68.5, "UG");

admission.checkEligibility("raj", 68.5, "PG");

System.out.println("thank you for coming to our collage");

}

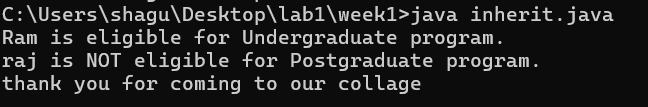
}

CLASS DIAGRAMS :

|  |
| --- |
| AdmisssionSystem |
|  |
| + checkEligibility(name: String,  percentage: double,  programType: String): void |

|  |
| --- |
| Inherit |
|  |
| +main:void |

OUTPUT:



ERROR:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1. | Error in constructor declaration | Rectified by giving correct data type name for the taken variables |

IMPORTANT POINTS:

1. **if statement** is used to check a condition. If it's true, the code inside runs.
2. **else if** checks another condition if the first if is false.
3. You can have **multiple else if** blocks to check different conditions.
4. **else** runs when **none** of the above conditions are true.
5. Used to perform **different actions** based on **different conditions**.

PROGRAME-4

AIM: To write a program for creating vehicle class with amethod displayInfo().Overide this method in the car subclass to provide specific information about cars[car company,car model, car prize,seating capacity, petrol or not{Boolean}]

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:

class Vehicle {

String car\_company;

String car\_model;

long car\_prize;

int seating\_capacity;

boolean petrol;

Vehicle(String car\_company, String car\_model, long car\_prize, int seating\_capacity, boolean petrol) {

this.car\_company = car\_company;

this.car\_model = car\_model;

this.car\_prize = car\_prize;

this.seating\_capacity = seating\_capacity;

this.petrol = petrol;

}

void displayInfo() {

System.out.println("Car company is: " + car\_company);

System.out.println("Model of the car is: " + car\_model);

System.out.println("Prize of the car is: " + car\_prize);

System.out.println("Seating capacity of the car: " + seating\_capacity);

System.out.println("Fuel type (petrol?): " + petrol);

}

}

class Car extends Vehicle {

Car(String car\_company, String car\_model, long car\_prize, int seating\_capacity, boolean petrol) {

super(car\_company, car\_model, car\_prize, seating\_capacity, petrol);

}

@Override

void displayInfo() {

System.out.println("----- Car Details (Overridden Method) -----");

System.out.println("Car company is: " + car\_company);

System.out.println("Model of the car is: " + car\_model);

System.out.println("Prize of the car is: " + car\_prize);

System.out.println("Seating capacity of the car: " + seating\_capacity);

System.out.println("Fuel type (petrol?): " + petrol);

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

}

}

public class inherit {

public static void main(String[] args) {

Car car1 = new Car("Ford", "X-Series", 2000000, 4, true);

car1.displayInfo();

Car car2=new Car("mersidies","BMW",23009900,6,true);

car2.displayInfo();

}

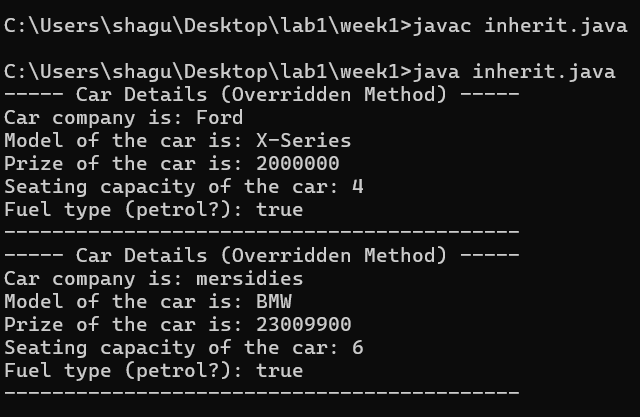
}

CLASS DIAGRAMS :

|  |
| --- |
| Vehicle |
| +car\_company:String  +car\_model:String  +car\_prize:long  +seating\_capacity:int  +petrol: Boolean |
| +Vehicle(String,String,long,int,boolean)  +displayInfo():void |

|  |
| --- |
| Car |
| (inherit all from Vehicle) |
| +car(String,String,long,int,boolean)  +displayInfo():void() @Overiden |

OUTPUT:



ERROR:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1. | Error :’;’ missing in printing statement | Rectified by inserting the ‘;’ in the printing statement. |

IMPORTANT POINTS:

1. **Method overriding** allows a subclass to provide a new version of a method already defined in its parent class.
2. It enables **runtime polymorphism**, where the method call is decided at **runtime** based on the object type.
3. Methods are defined **inside a class** to perform specific tasks or actions.
4. They usually have a **name**, **return type**, and **optional parameters**, and can be called using an object of the class.

WEEK-7

PROGRAME-1

AIM: To write a program for creating a Animal abstract class having an abstract method sound();overridden in its subclass Lion and Tiger with thir each sound.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:abstract class Animal{

    public abstract void sound();

}

//Over riden

class Lion extends Animal{

    public void sound(){

        System.out.println("Roar......Roar....!");

    }

}

//Overriden

class Tiger extends Animal{

    public void sound(){

        System.out.println("Growl.........Growl.....!");

    }

}

class Week{

    public static void main(String[] args) {

        Lion l = new Lion();

        Tiger t = new Tiger();

        l.sound();

        t.sound();

    }

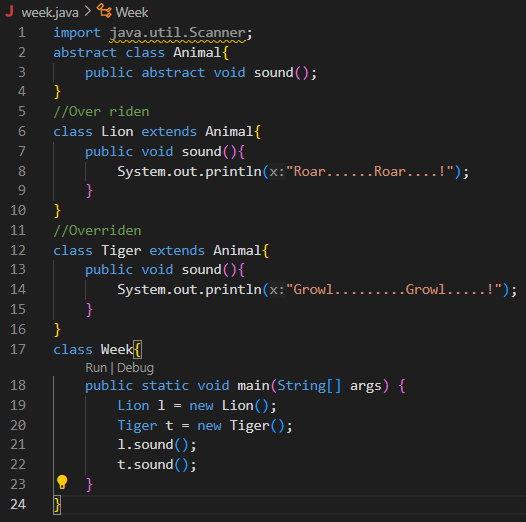
}

CLASS DIAGRAMS :

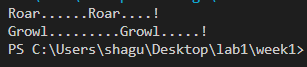
|  |
| --- |
| Animal |
| (abstract)sound():void |

|  |
| --- |
| Tiger |
| +Sound();void; |

|  |
| --- |
| Lion |
| +Sound():void; |



OUTPUT:



ERRROR:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1. | Error:java:6line error under “implementation” | Replaced it with “extends ” |

IMPORTANT POINTS;

1. **Cannot be instantiated** – Abstract classes cannot create objects directly; they are meant to be inherited by other classes.
2. **Contain abstract methods** – They can include abstract methods (without implementation) that must be defined in the subclass.
3. **Redefining a parent class method** – Overriding allows a subclass to provide a specific implementation of a method already defined in its parent class.

PROGRAME-2

AIM: To write a program to create an abstract class Shape abstract method calculatevolume() and surface area().create subclass Sphere and cube that extends the shape3d class implement the respective methods to calculate the volume and perimeter of each shape.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:

abstract class Shape3D{

abstract void calculateVolume();

abstract void calculateSurfaceArea();

}

class Sphere extends Shape3D{

    double r=6;

    @Override

    void calculateVolume(){

        System.out.println("Volume of sphere is "+(4.0/3.0)\*Math.PI\*Math.pow(r,2));

    }

    void calculateSurfaceArea(){

        System.out.println("Surface area of a sphere is "+4.0\*Math.PI\*Math.pow(r,2));

    }

}

class Cube extends Shape3D{

    double l=10;

    @Override

    void calculateVolume(){

        System.out.println("Volume of cube is "+(Math.pow(l,3)));

    }

    void calculateSurfaceArea(){

        System.out.println("Surface area of a cube is "+6.0\*Math.pow(l,2));

    }

}

class week{

    public static void main(String[] args){

        Sphere s=new Sphere();

        s.calculateVolume();

        s.calculateSurfaceArea();

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

        Cube c=new Cube();

        c.calculateVolume();

        c.calculateSurfaceArea();

    }

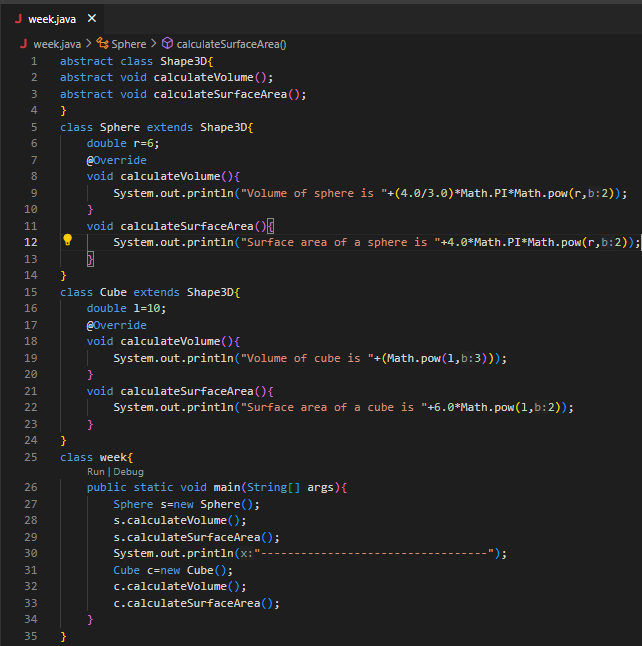
}

CLASS DIAGRAM:

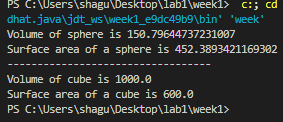
|  |
| --- |
| Shape3D |
|  |
| (abstract)calculatevolume:void  (abstract)calculatesurfacearea:void |

|  |
| --- |
| Sphere |
|  |
| +calcualtevolume:void  +calculatesurfacearea:void |

|  |
| --- |
| Cube |
|  |
| +calculatevolume:void  +calculatesurfacearea:void |



OUTPUT:



ERROR:

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
|  | Error: Math.pow(2,r)= output:2^r {wrong} | Replaced by :Math.pow(r,2) output: r^2{correct} |

IMPORTANT POINTS:

1. **Cannot be instantiated** – Abstract classes cannot create objects directly; they are meant to be inherited by other classes.
2. **Contain abstract methods** – They can include abstract methods (without implementation) that must be defined in the subclass.
3. **Redefining a parent class method** – Overriding allows a subclass to provide a specific implementation of a method already defined in its parent class

PROGRAME-3

AIM: Write a Java program using an abstract class to define a method for pattern printing.Create an abstract class named PatternPrinter with an abstract method printPattern(int n) & a concrete method to display the pattern title.

Implement 2 subclasses:

1. StarPattern – prints a right-angled triangle of class (\*)

2. NumberPattern – prints a right-angled triangle of increasing numbers

In the main() method, create objects of both subclasses & print patterns for a given no. of rows.

Step 1:open notepad<<save the note pad in the path[desktop<<oops<<week 1<<week.java

CODE:

import java.util.Scanner;

abstract class Pattran{

    abstract void printpattran();

    abstract void concrete();

}

class Pattran1 extends Pattran{

     void concrete(){

        System.out.println("PRINTING THE RIGHT ANGLED \* PATTEREN");

    }

    void printpattran(){

        Scanner inpuut=new Scanner(System.in);

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

        int n=inpuut.nextInt();

        for(int i=0;i<5;i++){

            for(int j=0;j<i;j++){

                System.out.print("\* ");

    }

    System.out.println("");

}

    }

}

class Pattran2 extends Pattran{

    void concrete(){

        System.out.println("PRINTING THE RIGHT ANGLED OF INCREASING NUMBERS PATTEREN");

    }

    void printpattran(){

        Scanner inpuut=new Scanner(System.in);

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

        int n=inpuut.nextInt();

        for(int i=1;i<n;i++){

            for(int j=1;j<=i;j++){

                System.out.print(+j);

                System.out.print("  ");

    }

    System.out.println("");

}

    }

}

class week{

    public static void main(String[] args){

        Pattran1 p1=new Pattran1();

        p1.concrete();

        p1.printpattran();

        Pattran2 p2=new Pattran2();

        p2.concrete();

        p2.printpattran();

    }

}

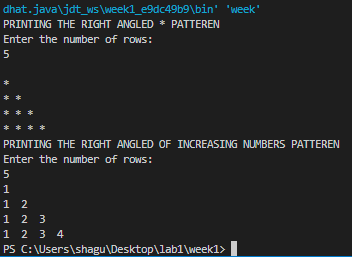
CLASS DIAGRAM:

|  |
| --- |
| <<abstract>>  Pattran |
| +printpattran():void  +concrete(): void |

|  |
| --- |
| Pattran1 |
| +printpattran():void  +concrete():void |

|  |
| --- |
| Pattran2 |
| +printpattran():void  +concrete():void |

OUTPUT:



ERROR:

 Incorrect pattern implemted the loop code correctely

Rectification:

  for(int i=1;i<n;i++){

            for(int j=1;j<=i;j++){

                System.out.print(+j);

                System.out.print("  ");

    }

    System.out.println("");

}

    }

IMPORTANT POINTS:

1. A **for loop** in Java is commonly used for printing patterns by repeating a block of code a specific number of times. In pattern printing, we often use **nested for loops**:
2. The **outer loop** controls the number of rows.
3. The **inner loop(s)** control the content of each row (like stars, numbers, spaces, etc.).
4. In Java, nextInt() is a method from the **Scanner** class. It is used to **read the next integer value** entered by the user from the input (usually from the keyboard).