23CSE111

OBJECT ORIENTED PROGRAMMING

LAB MANUAL

****

**Department of Computer Science and Engineering Amrita School of computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

# Name:T. Sri Harshith

**Verified By: Roll No: AV.SC.U4CSE24335**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Experiment Name** | **Page No.** | **Remarks** | **Signature** |
| **1** | Installation Process of JDK | **3** |  |  |
| **2** | a)Write a java program for SI?  b)Write a program in java for area of rectangle.  c) Write a program in java for area of triangle using heron’s formula.  d) (a):Write a program in java for converting temperature from celsius to fahrenheit.  (b):Write a program in java for converting temperature from fahrenheit to celsius.  e) Write a program in java for factorial of a number.  f) Write a program in java for fibonacci series | **13** |  |  |
| 3 | a.)Write a java program with the following instructions:  1. Create a class with name Car  2. Create 4 attributes named car\_color, rand, fuel\_type, mileage  3. Create 3 methods named Start(), Stop(),Service();  4. Create 3 objects named car1, car2, car3 for the class Car  b)Write a java program to create a class BackAccount with two methods deposit( ) and withdraw( )  1. In deposit( ) whenever an amount is deposited it has to be updated with current amount  2. In withdraw( ) whenever an amount is withdrawn it has to be less than current amount else print “Insufficient funds”. |  |  |  |
| **4** | 1)Write a java program with class named “Book”. The class should contain various attributes such as “Title of the book , author , year of publication “. It should also contain a constructor with parameters details of the book.  2)To create a java program with class named Myclass with a static variable “Count” of “int type”, Initialized to 0 and a constant variable “pi” of type double initialized to 3.1415 as attributes of that class Now, define a constructor for “Myclass” that increments the “Count” variable each that an object of Myclass is created. Finally , print the final values of “Count” and “pi” variables |  |  |  |
| **5)** | 1)Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output.  2) Vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each vehicle such as brand and speed.  i. Cars should have an additional property: number of doors, Seating capacity.  ii. Bikes should have a property indicating whether they have gears or not.  iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.  iv. Each class should have a constructor.  Questions:  1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.  2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?  a. Truck should include and additional property capacity (in tons).  b. Create a showTruck() method to display the truck’s capacity.  c. Write a constructor for truck that initializes all properties.  3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Finally display the details.- Write your code in VS CODE and execute |  |  |  |
| **6** | 1)Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and color using the constructor  2) Create a Java program for the scenario.  A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student's percentage in their previous qualification.  i) UG admissions require a minimum of 60%  ii) PG admissions require a minimum of 70%  3) Write a Java Program to create a Calculator class with overloaded methods to perform addition: Take the integer values a and b from the user.  i) Add two integers  ii) Add two doubles  iii) Add three integers  4) Write a Java Program to create a shape class with a method calculateArea() that is overloaded for different shapes(e.g., Square, Rectangle ). Then create a subclass Circle that overrides the calculateArea() method for a circle. |  |  |  |
|  |  |  |  |  |

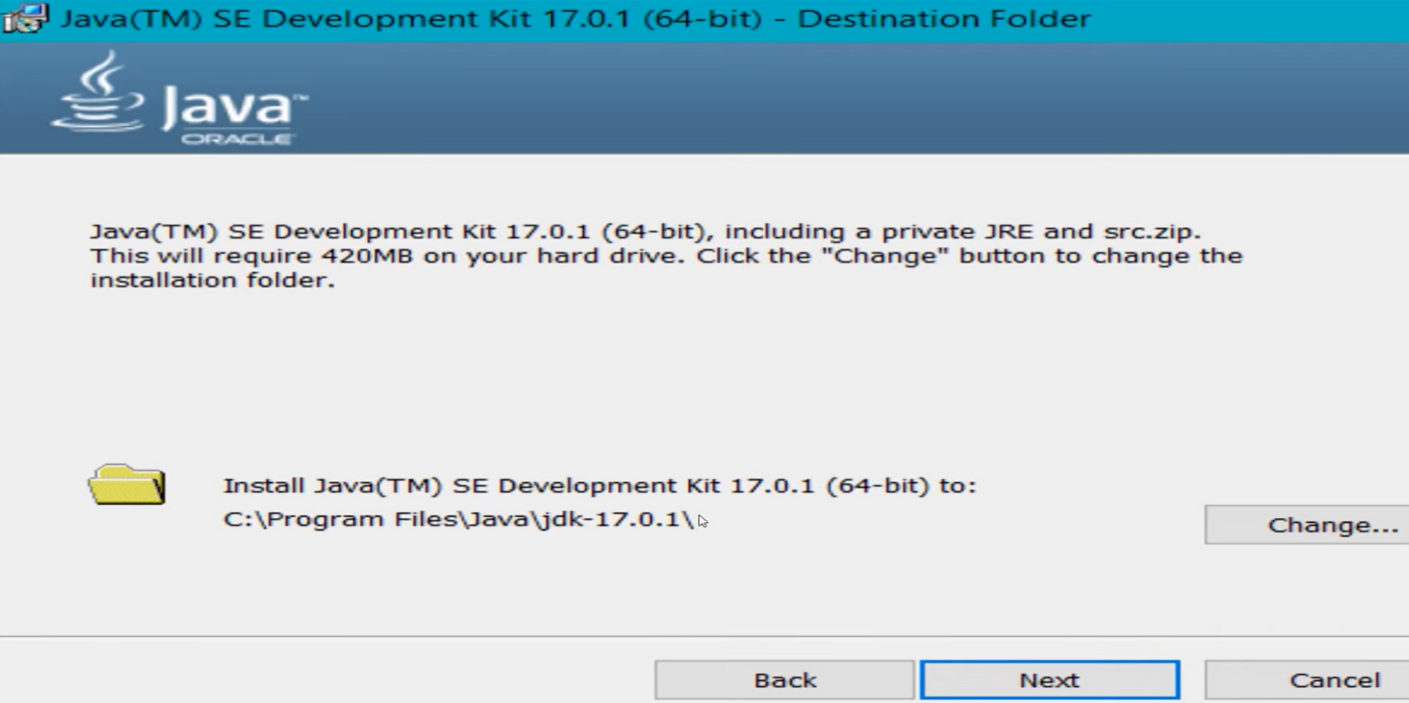
Object Oriented Program

* Open “Oracle” in your web browser.
* Click on “ download Java” in the oracle interface.
* Select the version you want to download (version JDK21 is best).
* Select the Operating System(OS) of your PC and click on the link “x64 installer” to download .
* **Installation:**

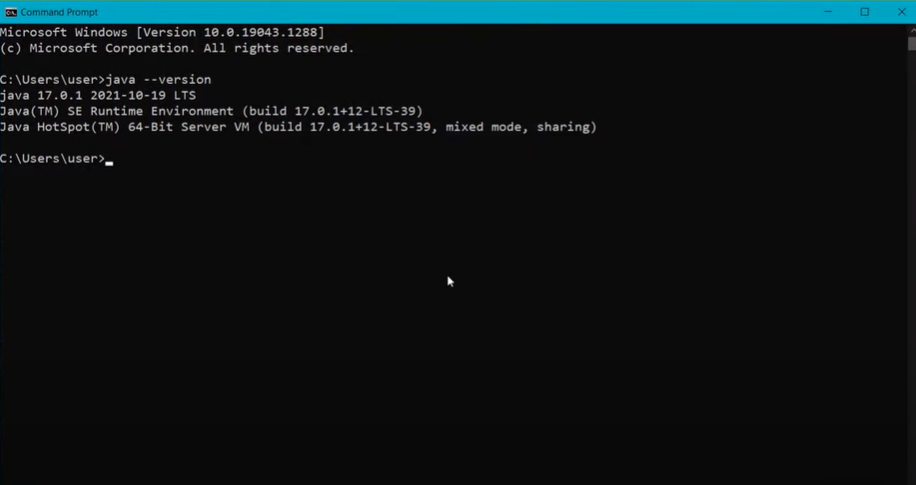
**Click on next**

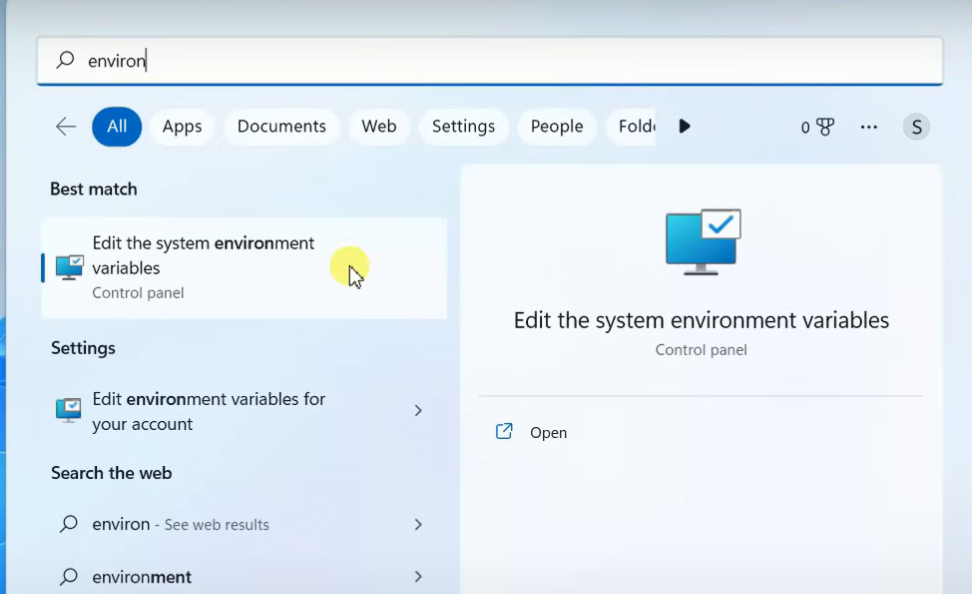
****

**click on next**

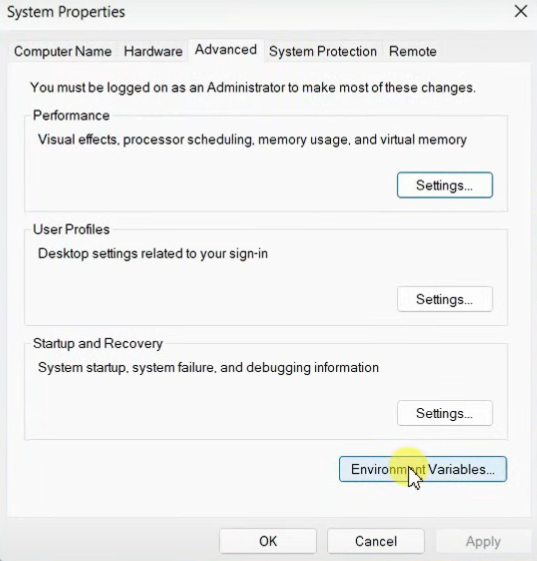
****

**click on close**

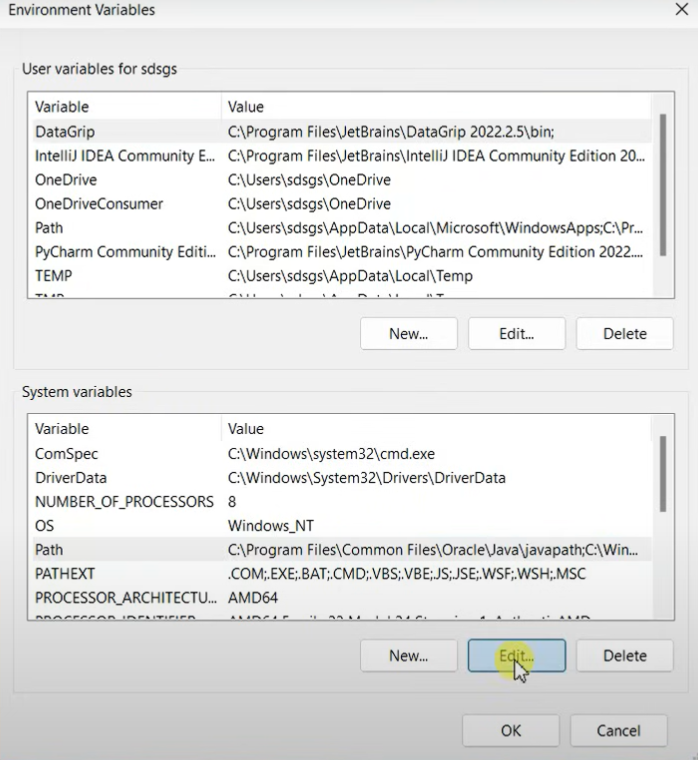
* **to check whether java is downloaded correctly or not**
* **open command window and type “java --version”**
* **to set the path:**
* **Go to “start” and se “Edit the search environment variables”**

****

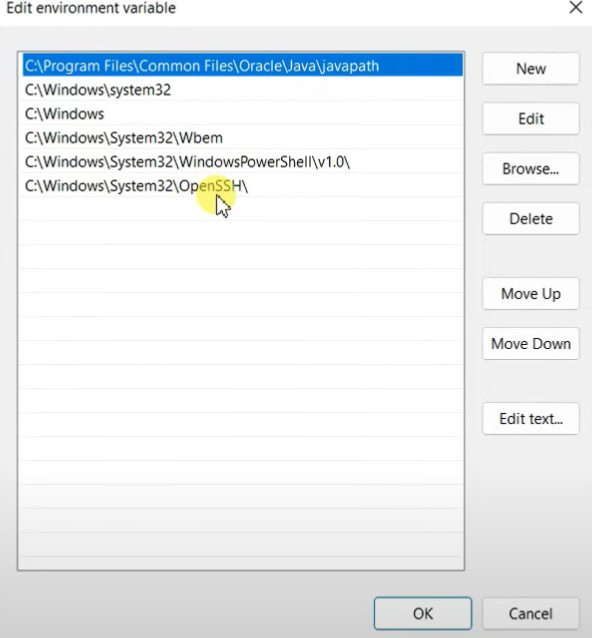
* **Click on open and continue to click on Environment variables**

****

* **Select the “path” option in “system variables”**

****

* **Now click on new paste the path of java**

****

* **To check whether the path is properly set or not**

**Open command window and type “javac --version”**

**If it is showing the exact version of java you are installed then the path is set properly.**

* **Now create a text document with “.java” extension and open in notepad ,**

**To write code.**

**a ) write a simple java program to print your name,roll no,sec etc.**

**Program :**

**class ME**

**{**

**public static void main(String[] args)**

**{**

**System.out.println("NAME= T.Sri Harsith");**

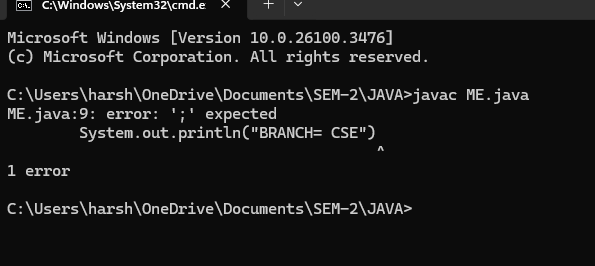
**System.out.println("ROLL NO= AV.SC.U4CSE24335");**

**System.out.println("SEC= B");**

**System.out.println("BRANCH= CSE")**

**}**

**}**

**OUTPUT: **

|  |  |  |
| --- | --- | --- |
| **S.no** | **Error** | **Rectification** |
| **1)** | **My first.java:error:’;’ expected**  **System.out.println("BRANCH= CSE")**  **^** | **In the code, semicolon must be added at the end of the line .** |

**IMPORTANT POINTS:**

**1)System.out.println(“ this string will be printed”)- this line of code is used to print any string**

**2) If you want to save your java file as “first” then “class first{“ should be written**

**AIM:**

**class ME**

**{**

**public static void main(String[] args)**

**{**

**System.out.println("NAME= T.Sri Harsith");**

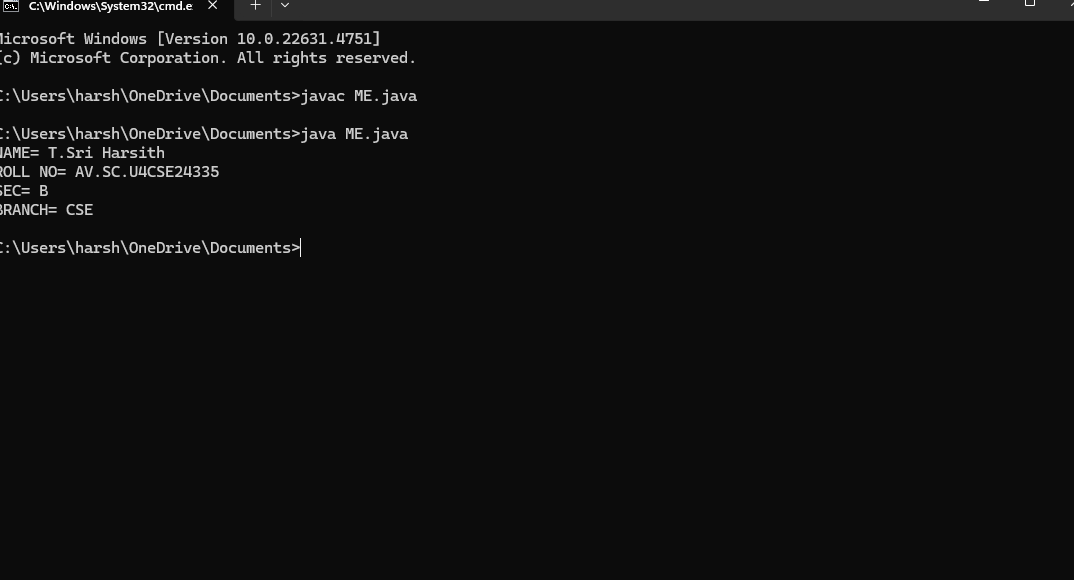
**System.out.println("ROLL NO= AV.SC.U4CSE24335");**

**System.out.println("SEC= B");**

**System.out.println("BRANCH= CSE")**

**}**

**}**

****

**WEEK-2**

**PROGRAM-1: Write a java program for SI?**

import java.util.Scanner;

class simple\_interest{

public static void main(String[] args){

Scanner si=new Scanner(System.in);

System.out.println("enter the principle:");

int p=si.nextint();

System.out.println("enter the rate of interest:");

int r=si.nextint();

System.out.println("enter the time:");

int t=si.nextint();

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

System.out.println("simple interest is: " + simple\_interest);

}

}

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  | int r=si.nextint(); | Int r=si.nextInt(); |

Program-1(rectification):

import java.util.Scanner;

class simple\_interest{

public static void main(String[] args){

Scanner si=new Scanner(System.in);

System.out.println("enter the principle:");

int p=si.nextInt();

System.out.println("enter the rate of interest:");

int r=si.nextInt();

System.out.println("enter the time:");

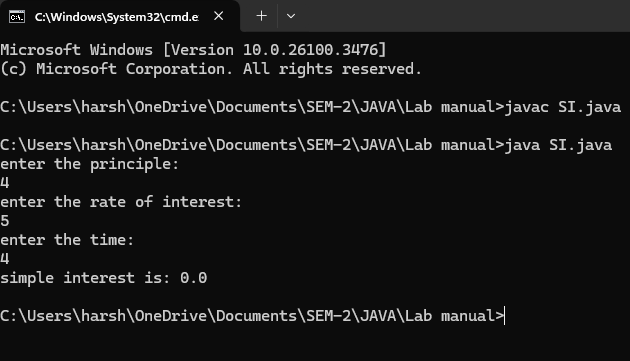
int t=si.nextInt();

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

System.out.println("simple interest is: " + simple\_interest);

}

}

****

**PROGRAM-2: Write a program in java for area of rectangle.**

import java .util. Scanner;

class area\_rectangle{

public static void main(String[] args){

Scanner a=new Scanner(System.in);

System.out.println("enter the length:");

float l=a.nextFloat();

System.out.println("enter the width:");

float w=a.nextFloat();

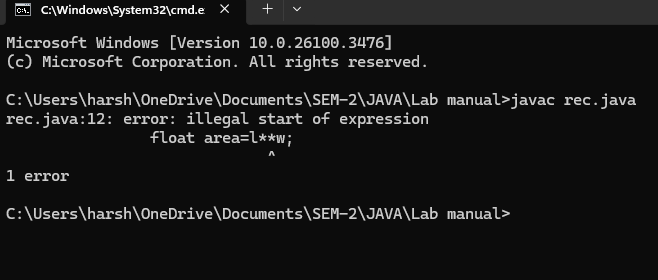
float area=l\*\*w;

System.out.println("area of rectangle is" + area);

}

}

Output:

****

|  |  |  |
| --- | --- | --- |
| code | error | rectification |
|  | Float area=l\*\*w; | For multiplication we should use “ \* ” for one time,so  Float area=l\*w; |

**Program-2 (rectification):**

import java .util. Scanner;

class area\_rectangle{

public static void main(String[] args){

Scanner a=new Scanner(System.in);

System.out.println("enter the length:");

float l=a.nextFloat();

System.out.println("enter the width:");

float w=a.nextFloat();

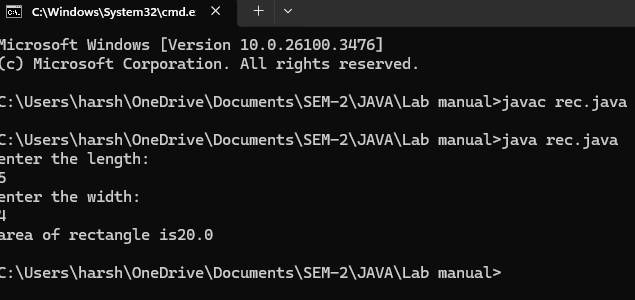
float area=l\*w;

System.out.println("area of rectangle is" + area);

}

}

**OUTPUT:**

****

**PROGRAM-3:Write a program in java for area of triangle using heron’s formula.**

import java.util.Scanner;

class heronsformula{

public static void main (String[] args){

Scanner input=new Scanner(System.in);

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

Double a=input.nextDouble();

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

Double b=input.nextDouble();

System.out.println("enter the value for c :");

Double c=input.nextDouble();

Double s=(a+b+c)/2;

System.out.println("S is the value of semi perimeter"+s);

Double x=s\*(s-a)\*(s-b)\*(s\_c);

System.out.println("the value of x is:"+x);

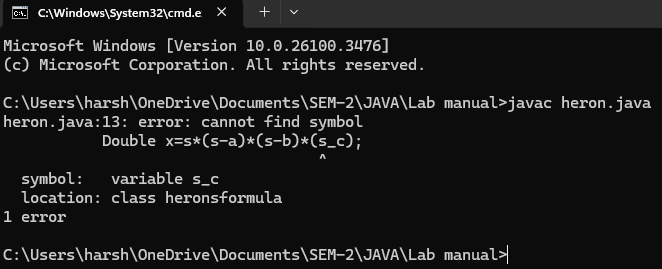
Double area=Math.sqrt(x);

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

}

}

**Output:**

****

|  |  |  |
| --- | --- | --- |
| **code** | **error** | **rectification** |
|  | **In formula ,**  **X=s\*(s-a)\*(s-b)\*(s\_c)** | **Instead of “ – “ we used “\_” there fore we got an error**  **X=s\*(s-a)\*(s-b)\*(s-c)** |

Rectification:

import java.util.Scanner;

class heronsformula{

public static void main (String[] args){

Scanner input=new Scanner(System.in);

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

Double a=input.nextDouble();

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

Double b=input.nextDouble();

System.out.println("enter the value for c :");

Double c=input.nextDouble();

Double s=(a+b+c)/2;

System.out.println("S is the value of semi perimeter"+s);

Double x=s\*(s-a)\*(s-b)\*(s-c);

System.out.println("the value of x is:"+x);

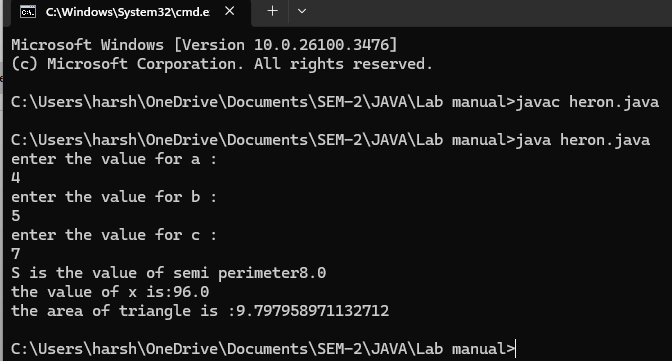
Double area=Math.sqrt(x);

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

}

}

**OUTPUT:**

****

**PROGRAM-4(a):Write a program in java for converting temperature from celsius to fahrenheit.**

import java.util.Scanner;

class temperature\_conversion{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

System.out.println("enter the Celcius :");

float ctemp=input.nextFloat();

float ftemp;

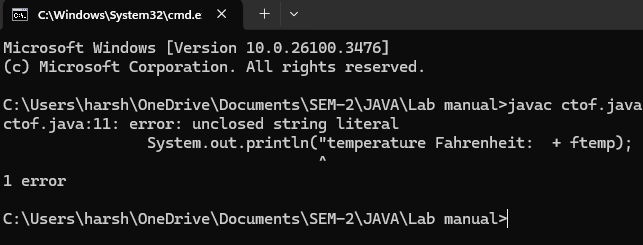
ftemp=(ctemp\*9/5)+32;

System.out.println("temperature Fahrenheit: + ftemp);

}

}

**OUTPUT:**

****

|  |  |  |
| --- | --- | --- |
| **code** | **error** | **rectification** |
|  | **System.out.println(“temperature Fahrenhiet: +ftemp)** | **System.out.println(“temperature**  **Fahrenhiet: “+ftemp)** |

**Rectification:**

**import java.util.Scanner;**

**class temperature\_conversion{**

**public static void main(String[] args){**

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

**System.out.println("enter the Celcius :");**

**float ctemp=input.nextFloat();**

**float ftemp;**

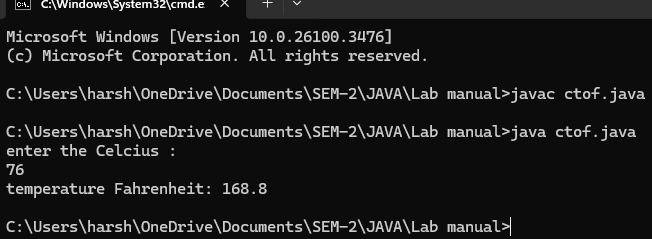
**ftemp=(ctemp\*9/5)+32;**

**System.out.println("temperature Fahrenheit: " + ftemp);**

**}**

**}**

**OUTPUT:**

****

**PROGRAM-4(b):Write a program in java for converting temperature from fahrenheit to celsius**.

import java.util.Scanner;

class temperature\_conver{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

System.out.println("enter the temperature in Fahrenheit:");

float ftemp=input.nextFloat;

float ctemp;

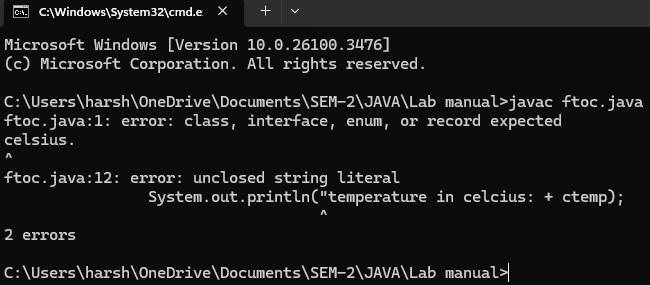
ctemp=(ftemp-32)\*5/9;

System.out.println("temperature in celcius: + ctemp);

}

}

Output:



|  |  |  |
| --- | --- | --- |
|  | **error** | **Rectification** |
|  | **Strings**  **System.out.println("temperature in celcius: + ctemp);** | **System.out.println("temperature in celcius:” + ctemp);** |

Rectification:

import java.util.Scanner;

class temperature\_conver{

public static void main(String[] args){

Scanner input=new Scanner(System.in);

System.out.println("enter the temperature in Fahrenheit:");

float ftemp=input.nextFloat();

float ctemp;

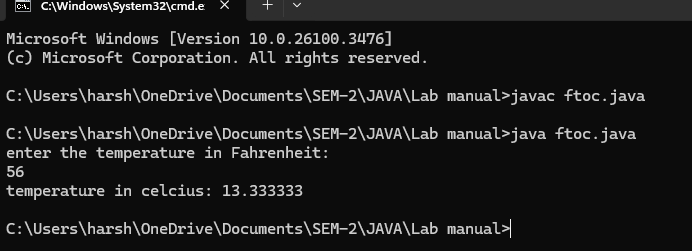
ctemp=(ftemp-32)\*5/9;

System.out.println("temperature in celcius: " +ctemp);

}

}

output:



**PROGRAM-5:Write a program in java for factorial of a number**.

import java.util.Scanner;

class factorial {

public static void main(String[] args) {

Scanner f=new Scanner(System.in);

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

int number=f.nextInt();

long factorial = 1;

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

factorial \*= i;

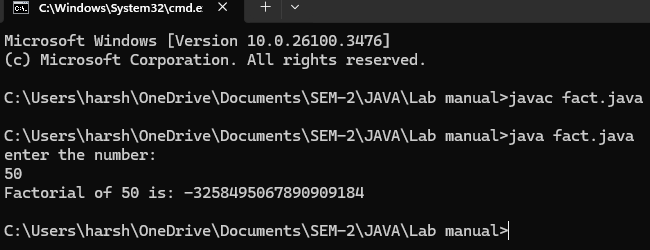
}

System.out.println("Factorial of " + number + " is: " + factorial);

}

}

Output:



|  |  |  |
| --- | --- | --- |
| **code** | **error** | **rectification** |
|  | **For root in java we**  **don’t use \*\*.** | **Use Math.sqrt**  **statement.**  **Double**  **area=Math.sqrt(x);** |

**PROGRAM-6:Write a program in java for fibonacci series**.

Output:

import java.util.Scanner;

class Fibonacci {

public static void main(String[] args) {

Scanner f=new Scanner(System.in);

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

int n=f.nextInt();

int firstTerm = 0, secondTerm = 1;

System.out.println("Fibonacci Series up to " + n + " numbers:");

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

System.out.print(firstTerm + " ");

int nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

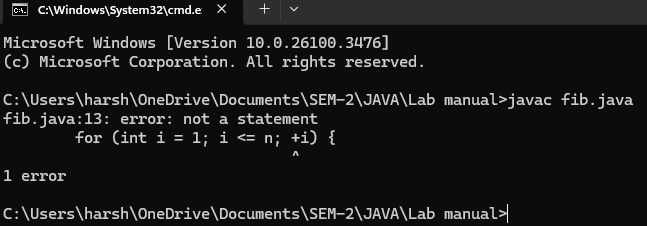
secondTerm = nextTerm;

}

}

}

Output:



|  |  |  |
| --- | --- | --- |
| **code** | **error** | **rectification** |
|  | **For increment we have used +i** | **We must use ++i for increment** |

Rectification:

class Fibonacci {

public static void main(String[] args) {

Scanner f=new Scanner(System.in);

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

int n=f.nextInt();

int firstTerm = 0, secondTerm = 1;

System.out.println("Fibonacci Series up to " + n + " numbers:");

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

System.out.print(firstTerm + " ")

int nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

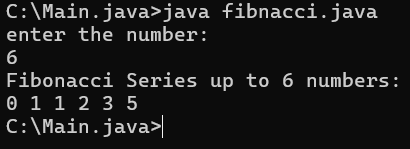
secondTerm = nextTerm;

}

}

}

**Output:**



**Week-3**

1. Aim:Create the java program with the following instructions
2. Create a class with name Car
3. Create 4 attributes named Car\_Color , Car\_brand, fuel\_type, mileage
4. Create 3 method named Start( ) , Stop( ),  Service( )
5. Create 3 objects Car1 ,  Car2 , Car3
6. Create a constructor which should print “Welcome to Car Garage”

code: class Car{

public String carColor;

private String carBrand;

private String fuelType;

public int mileage;

Car(String carColor , String carBrand , String fuelType , int mileage){

this.carColor = carColor;

this.carBrand = carBrand;

this.fuelType = fuelType;

this.mileage = mileage;

System.out.println(carColor + " " + carBrand + " " + fuelType + " " + mileage);

}

public void Start(){

System.out.println("The car has just started");

}

public void Stop(){

System.out.println("The car has just stopped");

}

public void Service(){

System.out.println("The car is in good condition");

}

public static void main(String[] args){

System.out.println(“Welcome to car Garage”);

Car Car1 = new Car("Black","Hyundai","Petrol",100);

Car1.Start();

Car Car2 = new Car("White","Suzuki","Diesel",150);

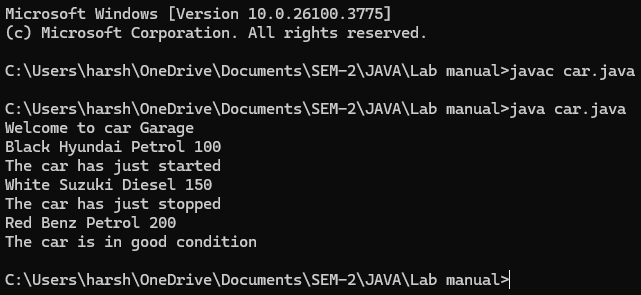
Car2.Stop();

Car Car3 = new Car("Red","Benz","Petrol",200);

Car3.Service();

}}

**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| s.no | Expected Error | Reason |
| 1 | } | } is expected at end of the calass |
| 2 | Setting the parameters inside  the constructer | Without setting the constructor we cannot pass the values |

Class Daigram:

|  |
| --- |
| Car |
| + carColor : String  - carBrand : String  - fuelType : String  + mileage : int |
| + Car( ) : void  + Start( ) : void  + Stop( ) : void  + Service( ) : void |

Important points:

* Private is an access specifier , It means attributes can be used in particular class only.
* Method is a part of code which only runs when it is called.

**2)**Aim:  Write a java program to create a class BackAccount with two methods deposit( ) and withdraw( )

1. In deposit( ) whenever an amount is deposited it has to be updated with current amount
2. In withdraw( ) whenever an amount is withdrawn it has to be less than current amount else print “Insufficient funds”.

**CODE:**

public class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

balance = initialBalance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: $" + amount);

System.out.println("Current Balance: $" + balance);

} else {

System.out.println("Deposit amount must be greater than zero.");

}

}

public void withdraw(double amount) {

if (amount > 0) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: $" + amount);

System.out.println("Current Balance: $" + balance);

} else {

System.out.println("Insufficient funds.");

}

} else {

System.out.println("Withdraw amount must be greater than zero.");

}

}

public double getBalance() {

return balance;

}

public static void main(String[] args) {

BankAccount account = new BankAccount(1000);

account.deposit(500);

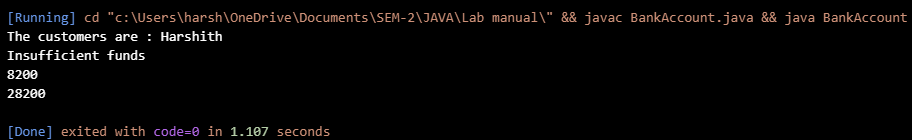
account.withdraw(200);

account.withdraw(2000);

}

}

OUTPUT:



**Errors :**

Error rectification table:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | subscription | Impact | Solution |
| Missing balance dispaly | Withdraw and deposit methods update the balance but not dispaly | User cannot see the current balance after transactions | Add system.out.println on both methods |
| No getter methods | No method to retrieve account details | Cannot account details outside of the class | Add getter methods like getBalance(), getAccName(), etc. |

CLASS DIAGRAM:

|  |
| --- |
| Bank\_Account |
| -accname:String  -acno:int  -balance:float |
| +withdraw(amount:int):void  +deposit(amount:int):void  +displaydetails():void |

IMPORTANT POINTS:

1. The Bank Account class has a private attribute balance to store the account balance.

2.The class has a constructor, BankAccount(double initialBalance), which initializes the balance. If the initial balance is not positive, it sets the balance to 0.

3.The deposit(double amount) method adds a positive amount to the balance and prints a message. If the deposit amount is not positive, it prints an error message.

4.The getBalance() method returns the current balance of the account.

5.The Main1 class contains the main method, which serves as the entry point of the program

**WEEK-4**

1.AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOULD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.

CODE:

public class Book {

public String title;

public String author;

public int year;

Book(String title, String author, int year) {

this.title = title;

this.author = author;

this.year = year;

}

public void displayDetails() {

System.out.println("Title: " +title);

System.out.println("Author: " +author);

System.out.println("Year of Publication" +year);

}

public static void main(String[] args) {

Book b1 = new Book("Mathematics", "Ramanujan", 1950);

Book b2 = new Book("Physics", "CV Raman", 1960);

b1.displayDetails();

b2.displayDetails();

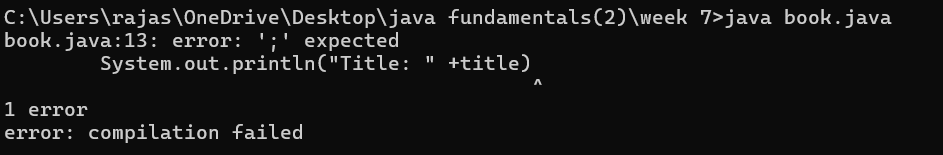
}

}

OUTPUT:



ERROR:



ERROR TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| ERROR TYPE | DESCRIPTION | IMPACT | SOLUTION |
| Compilation error | Missing semicolon | Code will not compile | Add semicolons at the end of the lines |
| Logical error | Missing spacing in the print statements | Output may be concatenated improperly | Add a space after year of publication in the display details () method |

CLASS DIAGRAM:

|  |
| --- |
| Book |
| - title: String  - author: String  - year: int |
| Book(String, String, int)  + displayDetails(): void |

IMPORTANT POINTS:

1. The Book class has three member variables: title, author, and year.
2. The constructor Book(String title, String author, int year) initializes the member variables with the values provided when a new Book object is created
3. The method displayDetails() prints the details of the book (title, author, and year of publication) to the console

2. AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH A STATIC VARIABLE COUNT OF INT TYPE. INTIALIZE IT TO ZERO AND A CONSTANT VARIABLE “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS ATTRIBUTES OF THAT CLASS. NOW DEFINE A CONSTRUCTOR FOR “MyClass”, THAT INCREMENTS THE COUNT VARIABLE EACH TIME AN OBJECT OF “MyClass” IS CREATED. FINALLY, PRINT THE FINAL VALUES OF ‘COUNT’ AND ‘PI’ VARIABLES AND CREATE 3 OBJECTS.

CODE:

public class MyClass {

static int count = 0;

static final double pi = 3.14;

MyClass() {

count++;

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

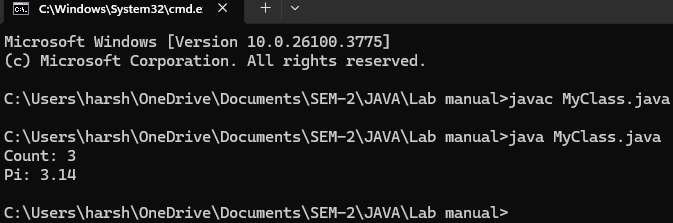
System.out.println("Count: " +count);

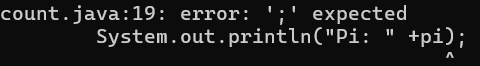
System.out.println("Pi: " +pi);

}

}

OUTPUT:

 Error:



ERROR TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Logical error | Missing space in the println statement for count and pi | Output may be concatenated improperly. | Add a space before +count and+pi in the println statement. |

CLASS DIAGRAM:

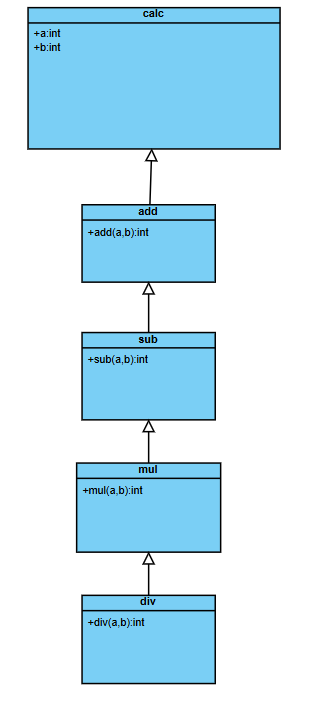
|  |
| --- |
| MyClass |
| - count: int (static)  - pi: double (static final) |
| + MyClass()  + main(String[] args): void |

IMPORTANT POINTS:

1. The MyClass class has two static member variables: count and pi
2. Count is initialized to 0 and is incremented each time a new MyClass object is created.
3. pi is a constant and cannot be changed.
4. he constructor MyClass() increments the count variable by 1 each time a new MyClass object is created.

**WEEK-5**

Aim: To Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output.



CODE:

import java.util.Scanner;

class Calc {

Scanner obj = new Scanner(System.in);

int a, b;

public Calc() {

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

a = obj.nextInt();

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

b = obj.nextInt();

}

}

class add extends Calculator {

public int addition() {

return a + b;

}

}

class sub extends add {

public int subtraction() {

return a - b;

}

}

class mul extends sub {

public int multiplication() {

return a \* b;

}

}

class div extends mul {

public double division() {

if (b == 0) {

System.out.println( "Please enter a number > 0 for b.");

return 0.0;

} else {

return (double) a / b;

}

}

}

class multilevel{

public static void main(String[] args) {

div obj = new div();

System.out.println("Addition of a and b: " + obj.add());

System.out.println("Subtraction of a and b: " + obj.sub());

System.out.println("Multiplication of a and b: " + obj.mul());

System.out.println("Division of a and b: " + obj.div());

Scanner.close();

}

}

OUTPUT:

ERROR TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | solution |
| Resource leak | Scanner input is not closed | Memory/resource leak, inefficient resource management | Adding scanner input close  Scanner.close |

Important points:

1.inheritence concept is used in this problem.

2.multilevel inheritance concept is used for this problem.

3.scanner input must be closed after the code unless it may cause for resource leak.

1. **Aim:** A vehicle rental company wants to devlop a system that maintains information about different types of vehicles available for rent. The company rents out cars , bikes and trucks and they need a program to store details about each vehicle, such as brand and speed.

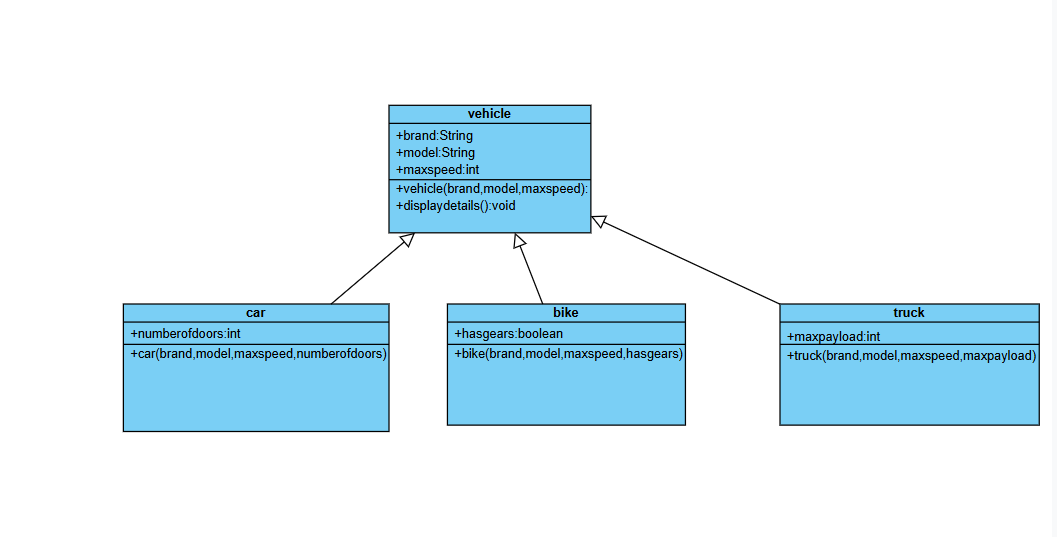
\*Cars should have an additional property such as number\_of\_doors .

\*Bikes should have a property indicating whether they have gears or not.

\*Truck should have a property of their capacity(in tons).

\*Every class should have a constructor.

Class diagram:



CODE:

class vehicle{

    String brand ;

    String model;

    int maxspeed;

    public vehicle(String brand,String model,int maxspeed){

        this.brand=brand;

        this.model=model;

        this.maxspeed=maxspeed;

    }

    public void start(){

        System.out.println(brand+" "+ model+" is starting");

    }

     public void displaydetails(){

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

        System.out.println("model:"+model);

        System.out.println("maxspeed:"+maxspeed);

     }

}

class car extends vehicle{

    public  int numberofdoors;

    public car(String brand,String model,int maxspeed,int numberofdoors){

        super(brand,model,maxspeed);

        this.numberofdoors=numberofdoors;

    }

    public void displaydetails(){

        super.displaydetails();

        System.out.println("number of doors:"+numberofdoors);

    }

}

class bike extends vehicle{

    public boolean hasgears;

    public bike(String brand,String model,int maxspeed,boolean hasgears){

        super(brand,model,maxspeed);

        this.hasgears=hasgears;

    }

    public void displaydetails(){

        super.displaydetails();

        System.out.println("has gears:"+(hasgears? "yes":"no"));

    }

}

class truck extends vehicle{

    public int maxpayload;

    public truck(String brand,String model,int maxspeed,int maxpayload){

        super(brand, model, maxspeed);

        this.maxpayload=maxpayload;

    }

        public void displaydetails(){

            super.displaydetails();

            System.out.println("maxpayload:"+maxpayload);

        }

}

    public class Main{

        public static void main(String[] args){

            car c=new car("Toyota", "Camry", 80, 4);

            c.displaydetails();

            c.start();

            bike b=new bike("Honda", "CBR500R",120,true);

            b.displaydetails();

            b.start();

            truck t=new truck("tata","intra",60,10000);

            t.displaydetails();

            t.start();

        }

    }

Output:



Important points:

\*The car, bike, and truck classes inherit from the vehicle. Hierarchical Inheritance is being used here.

\*Each subclass calls the parent constructor using super(brand, speed);, ensuring proper initialization.

ERROR TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | description | Impact | solution |
| ("tata","intra",”60”,10000 | In this case the int value has defined as a string in object creation | Leads to undefined constructor | ("tata","intra",60,10000)  Removing double codes for this 60 |

**WEEK-6**

1)

AIM:

To Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and color using the constructor

CODE:

class vehicle{

public void displayinfo(){

System.out.println("this is information about cars");

}

}

class car extends vehicle{

public void displayinfo(String brand,String model){

System.out.println("this is information about "+brand+" "+model);

}

}

class Main{

public static void main(String[] args){

vehicle v=new vehicle();

v.displayinfo();

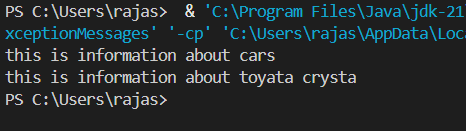
car c=new car();

c.displayinfo("toyata","crysta");

}

}

OUTPUT:



IMPORTANT POINTS:

1. The class car **inherits** from vehicle using the extends keyword
2. The method displayinfo(String brand, String model) in car is an **overloaded** method, not an **overridden** one
3. vehicle v = new vehicle(); creates a vehicle object
4. v.displayinfo(); calls the superclass method → prints info about cars.
5. All methods are declared as public, so they can be accessed from outside the class

2.AIM:A college is developing an automated admission system that verifies student eligibility for undergraduate and postgraduate each program has different criteria based on student percentage in their previous qualification

1.UG graduate requires minimum 60%.

2.PG graduate requires minimum 70%.

CODE:

class student{

String name;

double percentage;

public student(String name,double percentage){

this.name=name;

this.percentage=percentage;

}

void eligibility(){

System.out.println(name+"this student must have the foolowing criteria");

}

}

class UG extends student{

UG(String name,double percentage){

super(name,percentage);

}

void eligibility(){

if(percentage>=60){

System.out.println(name+"this student is eligible for ug");

}

else{

System.out.println(name+"this student is not eligible for ug");

}

}

}

class PG extends student{

PG(String name,double percentage){

super(name,percentage);

}

void eligibility(){

if(percentage>=70.0){

System.out.println(name+"this student is not eligible for pg");

}

else{

System.out.println(name+"this student is eligible for pg");

}

}

}

class collegeadmission{

public static void main(String[] args){

student s1=new student("Rahul",80.0);

s1.eligibility();

student s2=new student("Rahul",50.0);

s2.eligibility();

PG s=new PG("rahul",70.0);

s.eligibility();

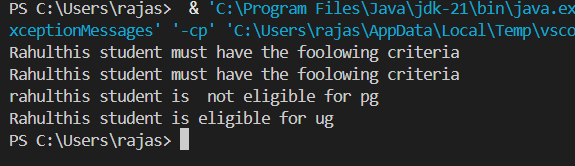
UG u=new UG("Rahul",60.0);

u.eligibility();

}

}

OUTPUT:



IMPORTANT POINTS:

1.Class Hierarchy & Inheritance

2. Subclasses (UG, PG) call the parent constructor using super(name, percentage) to initialize name and percentage.

3. UG and PG **override** the eligibility() method from student

4. The eligibility() methods in UG and PG check percentage thresholds:

5. Multiple objects (student, UG, PG) are created and their eligibility() methods are called.

3 AIM:Create a calculator class with overloaded method to perform addition

1.add two integers

2.add two doubles.

3.add three integers.

CODE:

class calculator{

public int add(int a,int b){

return a+b;

}

public double add(double a,double b){

return a+b;

}

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

return a+b+c;

}

}

class Addition{

public static void main(String[] args){

calculator calc=new calculator();

System.out.println("sum of two integers 5,10:"+calc.add(5,10));

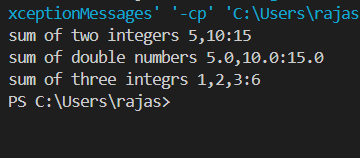
System.out.println("sum of double numbers 5.0,10.0:"+calc.add(5.0,10.0));

System.out.println("sum of three integrs 1,2,3:"+calc.add(1,2,3));

}

}

OUTPUT:



IMPORTANT POINTS:

**1.Method Overloading** in Java – same method name (add) with different parameter lists.

2. calculator class contains multiple overloaded add() methods:

3. Addition class contains the main() method:

4. Java resolves which method to call based on **argument types and number of arguments**.

4 AIM:Create a shape class with a method calculateArea() that is overloaded for different shapes(eg:squares,rectangle) then create a subclass circle that overrides the calculateArea() method for a circle

CODE:

class shape{

public double calculateArea(double a){

return a\*a;

}

public double calculateArea(double a,double b){

return a\*b;

}

}

class circle extends shape{

public double calculateArea(double radius){

return 3.14\*radius\*radius;

}

}

class Area{

public static void main(String[] args){

shape s=new shape();

System.out.println("area of the square"+ s.calculateArea(5.0));

System.out.println("area of rectangle:"+s.calculateArea(5.0,2.0));

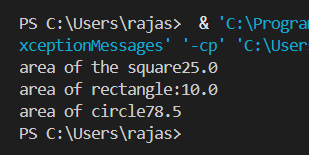
circle c=new circle();

System.out.println("area of circle"+c.calculateArea(5.0));

}

}

OUTPUT:



IMPORTANT POINTS:

1. **Method Overloading** in class shape

2. **Method Overriding** in class circle

3. **Inheritance** – circle extends shape

**WEEK-7**

**Task-1**

**AIM:** Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal.

- Important Points:

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**INPUT:**

public abstract class Animal {

     public abstract void sound();

}

 class Lion extends Animal{

    public  void sound(){

        System.out.println("Roars");

    }

}

class Tiger extends Animal{

    public void sound(){

        System.out.println("Growls");

    }

}

class Bag{

    public static void main(String[] args){

       System.out.println("Name: T Sri Harshith, Class: CSE-B, Roll no: AV.SC.U4CSE24335");

        Lion l=new Lion();

        l.sound();

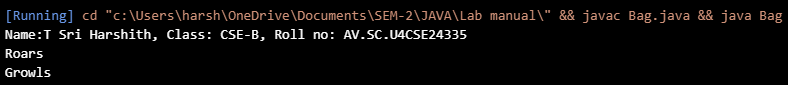
        Tiger t=new Tiger();

        t.sound();

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** | |
| 1 | Syntax error | Abstract key word is missed before method | Abstract keyword is added |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**Task-2**

**AIM:** Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

**INPUT:**

public abstract class Shape3D {

    public abstract void CalculateVolume();

    public abstract void CalculateSurfaceArea();

}

class Sphere extends Shape3D {

    int radius;

    Sphere(int radius) {

        this.radius = radius;

    }

    public void CalculateVolume() {

        System.out.println("Volume of sphere is: " + (4.0 / 3.0) \* Math.PI \* radius \* radius \* radius);

    }

    public void CalculateSurfaceArea() {

        System.out.println("Surface area of sphere is: " + 4 \* Math.PI \* radius \* radius);

    }

}

class Cube extends Shape3D {

    int side;

    Cube(int side) {

        this.side = side;

    }

    public void CalculateVolume() {

        System.out.println("Volume of cube is: " + side \* side \* side);

    }

    public void CalculateSurfaceArea() {

        System.out.println("Surface area of cube is: " + 6 \* side \* side);

    }

}

class Normal{

    public static void main(String[] args){

        System.out.println("Name: T Sri Harshith,class: CSE-B,Rollno:AV.SC.U4CSE24335");

        Sphere sp = new Sphere(5);

        Cube c = new Cube(4);

        System.out.println("    ");

        sp.CalculateSurfaceArea();

        System.out.println("    ");

        sp.CalculateVolume();

        System.out.println("    ");

        c.CalculateSurfaceArea();

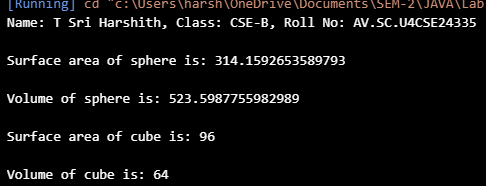
        System.out.println("    ");

        c.CalculateVolume();

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | package is missed before abstract class | Package is imported |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**Task-3**

**AIM:** Write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method printpattern(int n) and a concrete method to display the pattern title.

Implement two subclasses:

1) Star pattern - Prints a right-angled triangle of stars(\*).

2) Number pattern - Prints a right- angled triangles of increasing numbers.

In the main() method, create Objects

Star Pattern Number pattern

\* 1

\*\* 1 2

\*\*\* 1 2 3

\*\*\*\* 1 2 3 4

\*\*\*\*\* 1 2 3 4 5

**Importantpoints:**

1.understanding the loops

2.understanding the nested for

3.learning the pattern printing

4.understanding abstract classes and methods.

**INPUT:**

public abstract class pattrenprinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println(title);

    }

}

class StarPattern extends pattrenprinter {

    void printPattern(int n) {

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

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

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

            }

            System.out.println();

        }

    }

}

class NumberPattern extends pattrenprinter {

    void printPattern(int n) {

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

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

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

            }

            System.out.println();

        }

    }

}

 class PatternProgram {

    public static void main(String[] args) {

        System.out.println("Name: T Sri Harshith,class: CSE-B,Rollno:AV.SC.U4CSE24335");

        StarPattern sp = new StarPattern();

        NumberPattern np = new NumberPattern();

        System.out.println("    ");

        sp.displayTitle("Star Pattern");

        sp.printPattern(5);

        System.out.println("    ");

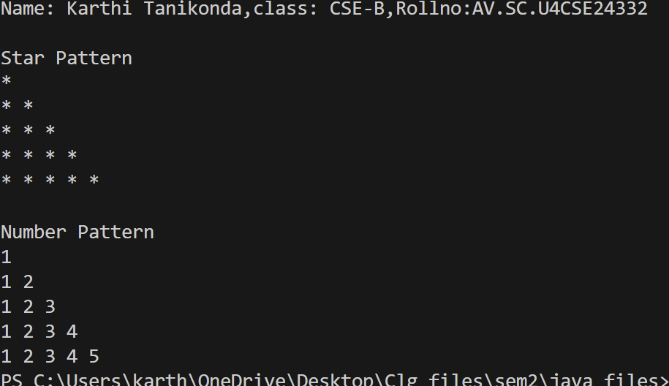
        np.displayTitle("Number Pattern");

        np.printPattern(5);

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Rectification** |
| 1 | Syntax error | Increment condition is added in subclass method |
| 2 | Logical error | Corrected logic in subclass method |
| 3 | Incorrect condition | Rectified the condition |

# **WEEK-8**

**Task-1:**

**Aim:** Write a Java program to create an interface Shape with the getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.

**Class Diagram:**

+ getPerimeter(): void

<<interface>> Shape 4

|  |
| --- |
| Rectangle |
| * length: double * breadth: double |
| +Rectangle()  +getPerimeter():void |

|  |
| --- |
| Circle |
| * radius: double * pi: double |
| +Circle()  +getPerimeter():void |

|  |
| --- |
| Triangle |
| * a: double * b: double * c: double |

|  |
| --- |
| +Triangle()  +getPerimeter():void |

**PROGRAM:**

interface Shape4{ void getPerimeter();

}

class Rectangle implements Shape4{ double length;

double breadth;

public Rectangle(double length, double breadth) { this.length = length;

this.breadth = breadth;} @Override

public void getPerimeter(){ double p = 2 \* (length + breadth);

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

}

}

class Circle implements Shape4{ double radius;

double pi;

public Circle(double radius, double pi){

this.radius=radius; this.pi=pi;} @Override

public void getPerimeter(){ double p1=2\*pi\*radius;

System.out.println("Perimeter of circle is:"+p1);

}

}

class Triangle implements Shape4{ double a;

double b; double c;

public Triangle(double a, double b, double c){ this.a=a;

this.b=b; this.c=c;

}

@Override

public void getPerimeter(){ double p3=a+b+c;

System.out.println("Perimeter of Triangle is:"+p3);

}

}

public class Perimeters{

public static void main(String[] args){ Rectangle r=new Rectangle(3,5); Circle c=new Circle(4,3.1415); Triangle t=new Triangle(3,4,5);

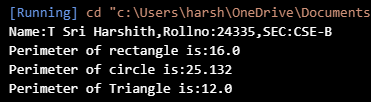
System.out.println("Name:T Sri Harshith,Rollno:24335,SEC:CSE-B"); r.getPerimeter();

c.getPerimeter(); t.getPerimeter();

}

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **Error**  **name** | **Error Rectification** |
| 1. | **Method Signature Mismatch** | Different return type or parameters than interface declaration. |
| 2. | Calculation  Logic Error | incorrect formula for perimeter. |

+ play(): void

Football

**IMPORTANT POINTS:**

When performing a mathematical operations with double variables We should not write (1/2) instead write (1.0/2.0) or (1.0/2).

**TASK-2:**

**AIM:**Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.

**CLASS DIAGRAM:**

+ play(): void

<<interface>> Playable

+ play(): void

Basketball

Volleyball

+ play(): void

**PROGRAM:**

interface Playable{ void play();

}

class Football implements Playable{ public void play(){ System.out.println("Player kicks the ball");

}

}

class Volleyball implements Playable{ public void play(){

System.out.println("Hits the ball with hand");

}

}

class Basketball implements Playable{ public void play(){

System.out.println("Throws the ball in the basket");

}

}

public class Playables{

public static void main(String[] args){ Football f=new Football();

Volleyball v=new Volleyball(); Basketball b=new Basketball();

System.out.println("name:T Sri Harshith,roll.no:AV.SC.U4CSE24335mSection:CSE-B"); f.play();

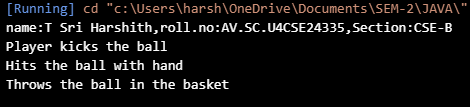
v.play();

b.play();

}

}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.No | Error | Error rectification |
| 1 | play() not overridden in  sports classes. | Implement play() |
| 2 | Error: ”;” expected | Giving the ‘;’ in wanted places |

**Important points:**

Use implements keyword to get the methods of and variables from interface to a class.

**Task-3:**

**Aim:**Write a Java program to implement a login system using interfaces.

**CLASS DIAGRAM:**

+ play(): void

<<interface>> Login\_System

+ login(id: String,Pass:String):boolean

University

**PROGRAM:**

interface Login\_System{

boolean login(String id,String Pass);

}

class University implements Login\_System{

public boolean login(String id,String Pass){

if (id.equals("Student123") && Pass.equals("Pass123")){ System.out.println("Login Successful");

return true;

}

else {

System.out.println("Invalid Credentials"); return false;

}

}

}

class Login1{

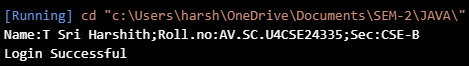
public static void main(String[] args){ University u=new University();

System.out.println("Name:T Sri Harshith;Roll.no:AV.SC.U4CSE24335;Sec:CSE-B"); u.login("Student123","Pass123");

}

}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.No | Error | Error rectification |
| 1 | Login\_System() not  overridden in University classes. | Implement Login\_System() in  university class using @override annotation |
| 2 | Error: ”;” expected | Giving the ‘;’ in wanted places |

**Important points:**

Boolean type is used in many applications when we need to check the input is true or false.