**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Department of Computer Science Engineering**   **Amrita School of Computing**

**Name***: v.yaswanth*

**Roll No:** 24344

**Verified By :**

***FIRST WEEK:***

***LAB 1:***

*PROGRAMM 1:* How to install java program in our devices

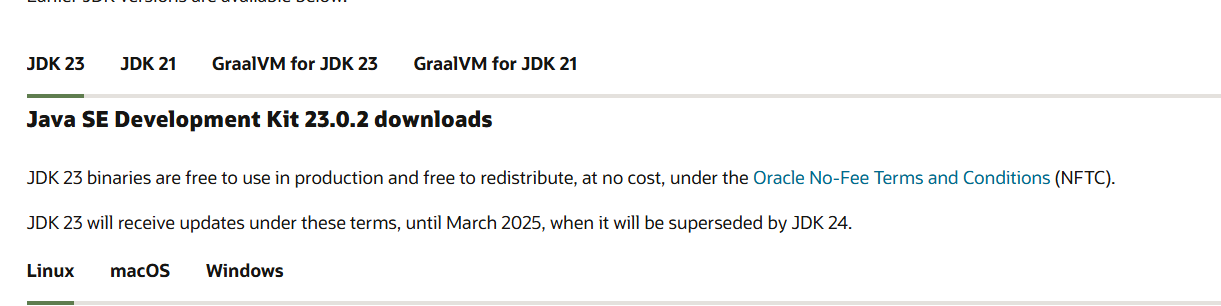
Procedure:

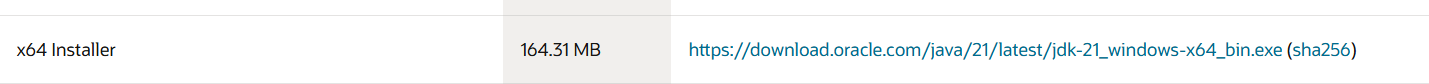
***Step 1:*** Go to official website of oracle java downloads



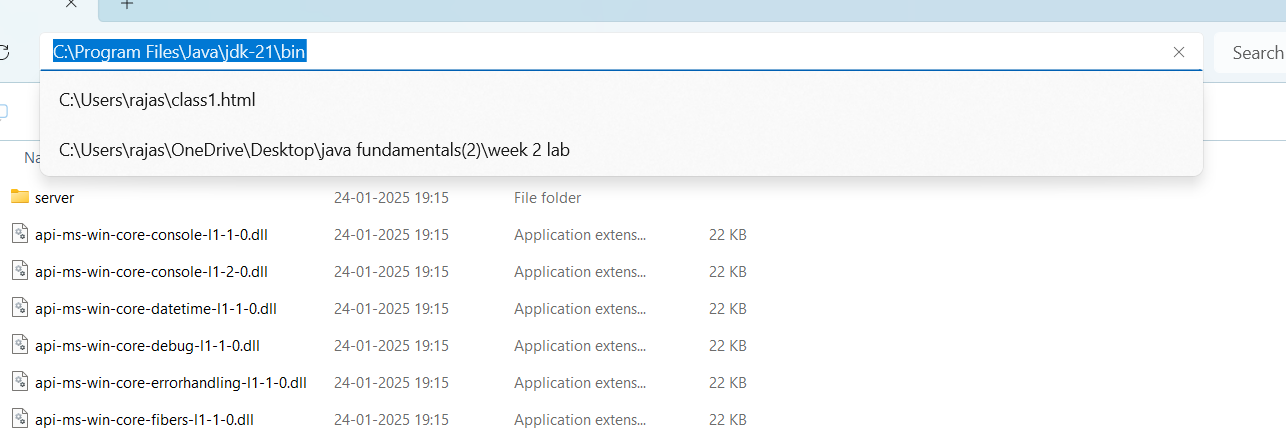
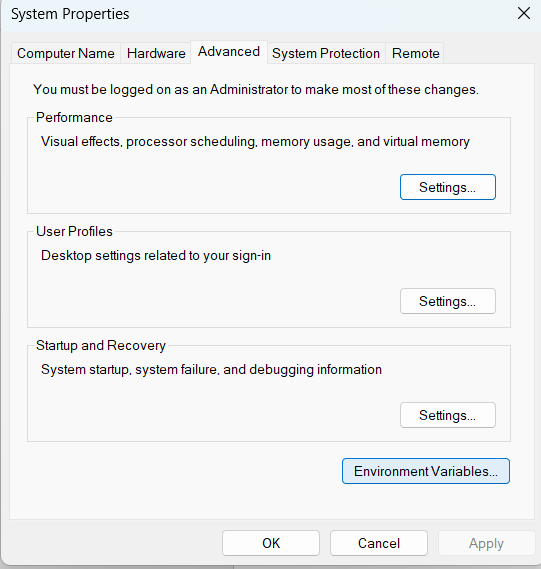
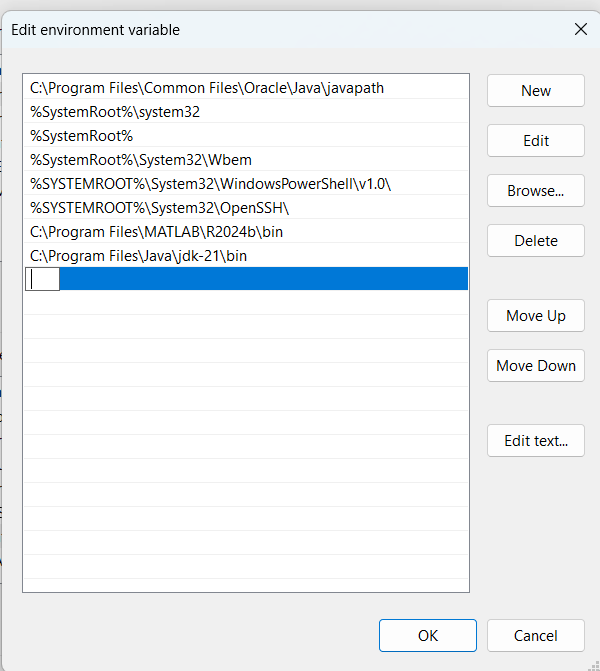
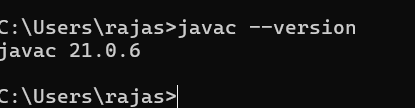
Click on the link as shown above

***Step 2:***



* Select JDK 21 as shown above
* Go to windows and as shown bottom in the figure
* Go to and click select x64 installer
* As shown in the down click on it
* 
* As it was not fully downloaded in the system we need set environment variables

***Step 3:***

* Setting environment variables on our system.
* Open c drive on your system
* In that open program files
* In that double click on java
* There you can see jdk 21
* And select it and then select the Bin from that.
* Now select the path or copy path
* 
* In windows bar search edit the system enivornment variables.
* Open it and select the enivornment variables in the down as shown in the below diagram.
* 
* In that select the system variables and select the on the path which is appearing and click on the new.
* And copy the path and click ok and apply as shown in the diagram.
* 
* Now your java compilation is completely downloaded.
* For verification go on command prompt and type javac –version
* It will show the version as it shown below
* 

*PROGRAMME 2 :* Simple code of entering the student details.

Public class Main{

public static void main(String[] args){

System.out.println(“ Student name:YASWANTH”);

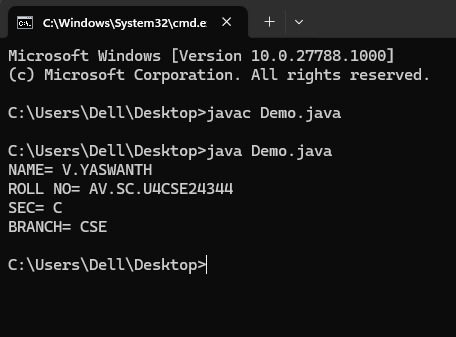
System.out.println(“Roll:no:24344”);

System.out.println(“Section:CSE-C”);

}

}

Output:



***WEEK 2 :***

*LAB 2:*

1.SIMPLE INTREST PROGRAM OF JAVA USING SCANNER CLASS.

Aim: Write a java program for simple intrest

CODE:

import java.util.Scanner;

class si{

private String amount;

private String time;

private String intrest;

private String simpleintrest;

public static void main(String[] args){

Scanner si=new Scanner(System.in);

System.out.print("enter the principal amount:");

int amount=si.nextInt();

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

System.out.print("enter the time in terms of months:");

int time=si.nextInt();

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

System.out.print("rate of intrest:");

int intrest=si.nextInt();

System.out.println("rate of intrest:"+intrest);

si.close();

if(amount<=0||time<=0||intrest<=0){

System.out.println("please enter in terms of positive numbers");

return;

}

int simpleintrest=amount\*time\*intrest/100;

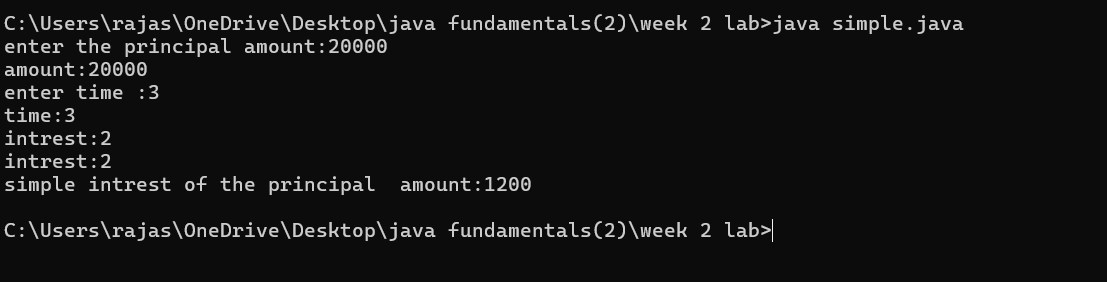
System.out.println("simple intrest:"+simpleintrest);

}

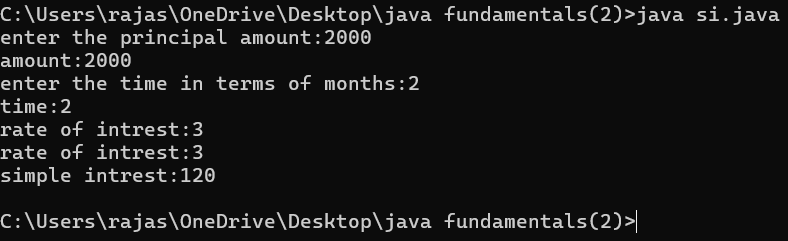
}

OUTPUT:

1.

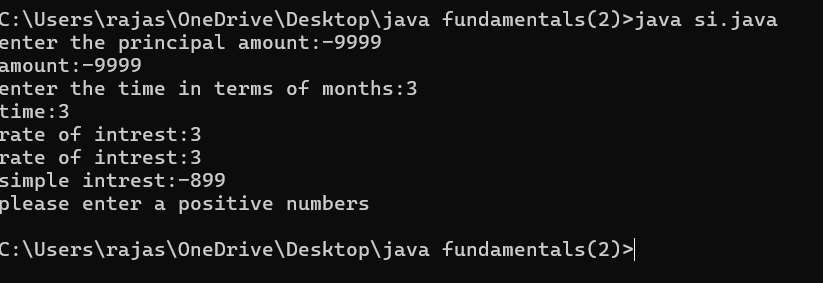


OUTPUT 2:



OUTPUT 3:

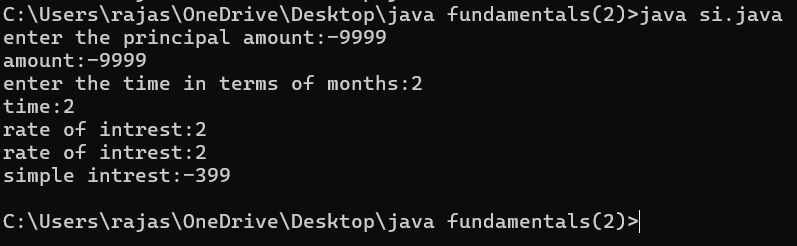
CHECKING FOR NEGATIVE VALUES



ERROR :

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Conflicting class and variable name | The class name si conflicts with the Scanner object named si | Can cause confusion and potential conflicts | Rename the class name |
| No input validation | No check for negative or zero values in amount ,time,or intrest | May allow invalid values leading to incorrect results | Add validation if(amount<=0,t<=0,i<=0) |

ERROR OUTPUT:



2.CONVERTING TEMPERATURE FROM CELSIUS TO FARENHEIT USING SCANNER CLASS

Aim: Writing a java program for temperature calculating

CODE:

import java.util.Scanner;

class Main{

public static void main(String[] args){

Scanner input =new Scanner(System.in);

System.out.println("enter temperature in terms of celsius:");

int c= input.nextInt();

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

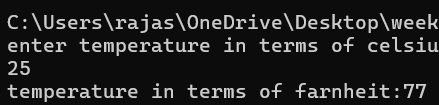
System.out.println("temperature in terms of farnheit:"+f);

Input.close();

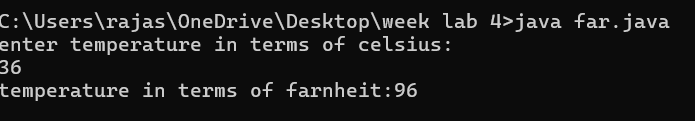
}

}

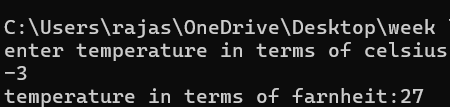
OUTPUT: for c=25



OUTPUT:FOR C=36



OUTPUT: FOR C=-3



|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Resource leak waring | Scanner input not closed | May cause unnecessary memory usage | Add input.close() |

2.ii) JAVA PROGRAM FOR FARNHEIT TO CELSIUS

Aim: writing java program for temperature calculation

CODE:

import java.util.Scanner;

class Main{

public static void main(String[] args){

Scanner input =new Scanner(System.in);

System.out.println("enter temperature in terms of franheit :");

int f= input.nextInt();

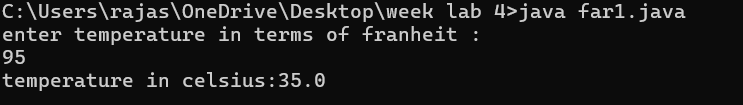
double c=(f-32)\*5/9;

System.out.println("temperature in celsius:"+c);

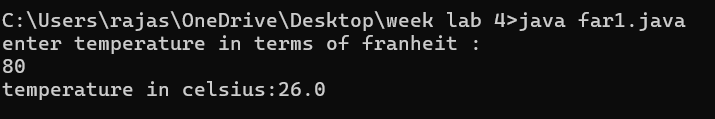
}

}

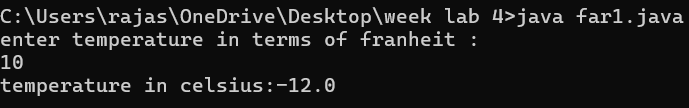
OUTPUT:

FOR F= 95

FOR F=80



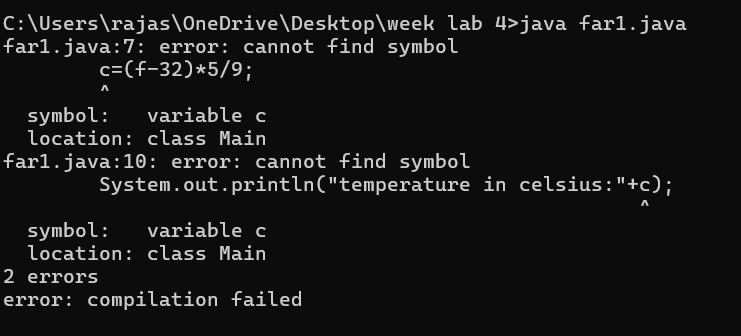
FOR F=10



Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Compilation error | C is used without declaration | The program won’t compile due to an undeclared variable | Declare c as a double or int |
| Logical error | Incorrect formula instead of 9/5 write 5/9 | The conversion will be given wrong output | Use the correct formula (f-32)\*5/9 |

ERROR OUTPUT:



3.JAVA PROGRAM FOR FACTORIAL USING SCANNER CLASS:

Aim: Writing java program for factorial of a number.

CODE:

import java.util.Scanner;

class Main{

public static void main(String[] args){

Scanner ms=new Scanner(System.in);

System.out.println("enter factorial of n:");

int n= ms.nextInt();

int t=1;

if (n<0){

System.out.println(“enter a positive number instead of negative”);

Return;

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

t=n\*t;

}

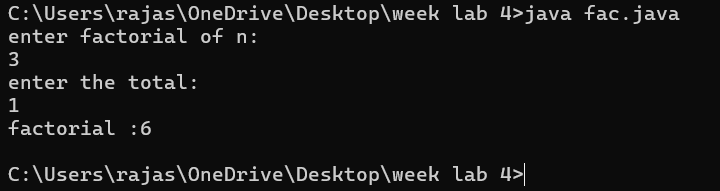
System.out.println("factorial :"+t);

}

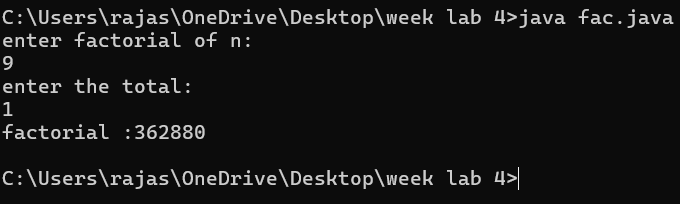
}

OUTPUT:for n=3

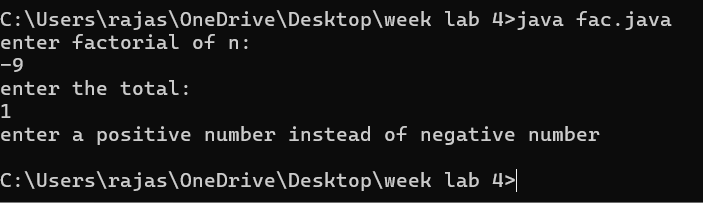
1.



OUTPUT 2: for n=9



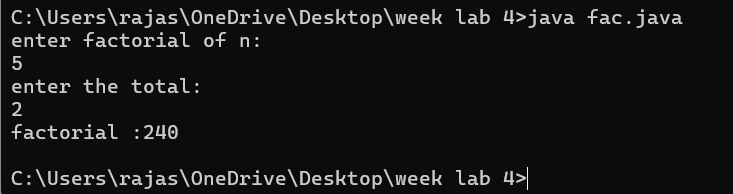
OUTPUT 3: for n=-9



ERROR RECTIFICATION TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Scanner not closed | Scanner object is not closed after use | Minor resource leak | Add ms.close;at the end main() |
| Incorrect factorial calculation | The factorial is misapplied | Produce incorrect results | Initialize t=1 before the loop |

Error:



4.JAVA PROGRAM FOR AREA OF RECTANGLE:

Aim:Writing java program for area of rectangle.

CODE:

import java.util.Scanner;

class Main{

public static void main(String[] args){

Scanner ss=new Scanner(System.in);

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

int l=ss.nextInt();

System.out.println("enter the breadth of rectangle:");

Int b=ss.nextInt();

if(l<0||b<0){

System.out.println(“enter positive numbers”);

return;

}

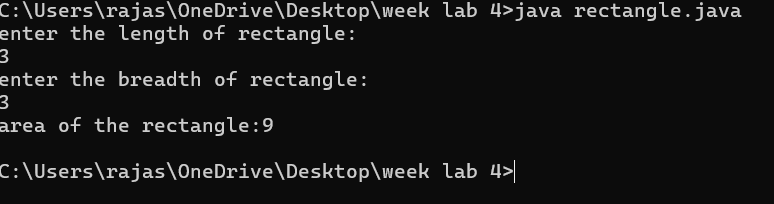
int a=l\*b;

System.out.println("area of the rectangle:"+a);

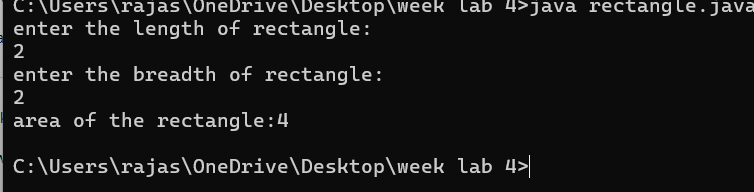
}

}

OUTPUT 1:

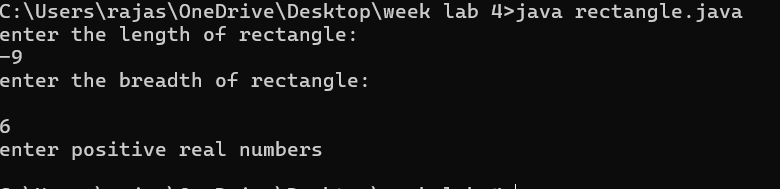


OUTPUT 2:



OUTPUT 3:

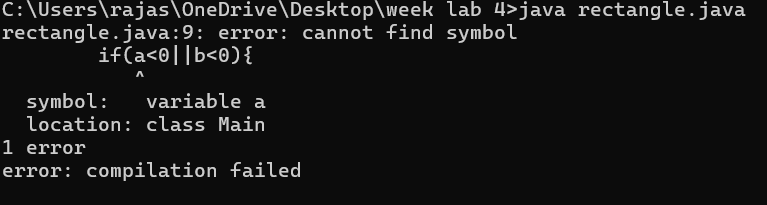
For negative values



ERROR RECTIFICATION TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| No return after error | If sides are invalid  Program still tries to compute area | May result in terms of incorrect values | Add return to the terms |
| No input validation  And unrecognized a | Program does not check for negative or zero | Incorrect output | Add if (l<=0 and b<=0) |

Error:



5.JAVA PROGRAM FOR AREA OF TRIANGLE USING HERON’S FORMULA:

Aim:Writing a programm for area of triangle using heron’s formula using a scanner class.

CODE:

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner ss = new Scanner(System.in);

System.out.println("Enter 1st side length of triangle:");

int a = ss.nextInt();

System.out.println("Enter 2nd side:");

int b = ss.nextInt();

System.out.println("Enter 3rd side:");

int c = ss.nextInt();

if (a <= 0 || b <= 0 || c <= 0) {

System.out.println("Error: Please enter positive numbers.");

return;

}

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

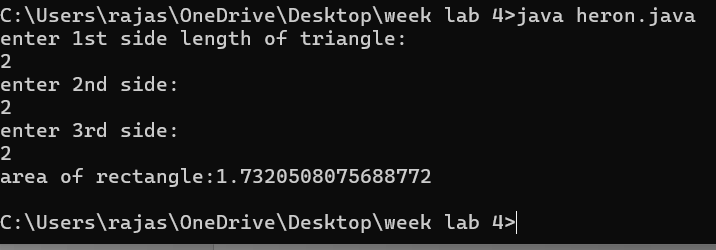
double A = Math.sqrt(s \* (s - a) \* (s - b) \* (s - c));

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

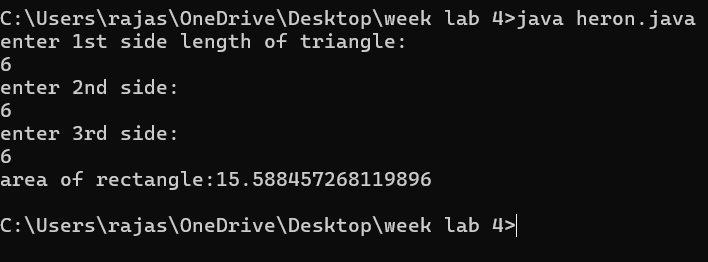
}

}

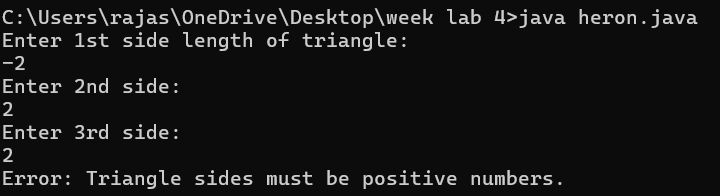
OUTPUT 1:



OUTPUT 2:



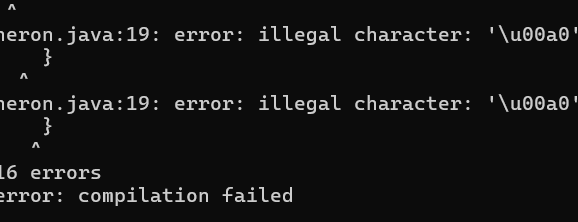
OUTPUT 3:



ERROR RECTIFICATION TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error | Description | Impact | Solution |
| Missing multiplication operators | Between (s-a),(s-b),(s-c) is missing | Compilation error  To invalid syntax | Use multiplication opertors in between those terms |
| No return after error | If sides are invalid  Program still tries to compute area | May result in terms of incorrect values | Add return to the terms |

Error:



6.JAVA PROGRAM FOR FIBONACCI SERIES USING SCANNER CLASS:

Aim:Writing a java programm for Fibonacci series using scanner class.

CODE:

import java.util.Scanner;

class Main{

public static void main(String[] args){

Scanner ss=new Scanner(System.in);

System.out.println("enter Fibonacci length you want:");

int n=ss.nextInt();

if(n>0){

int f1=0;

int f2=1;

System.out.println("fibnnocai series of length:"+n);

System.out.println(f1);

System.out.println(f2);

for (int i=3;i<=n;++i){

int f3=f1+f2;

System.out.println(f3);

f1=f2;

f2=f3;

}

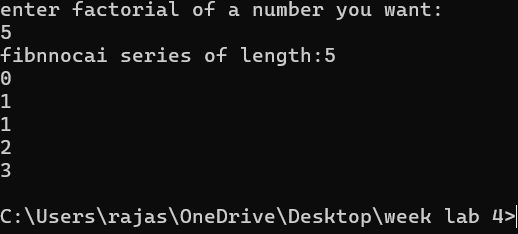
}

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

}

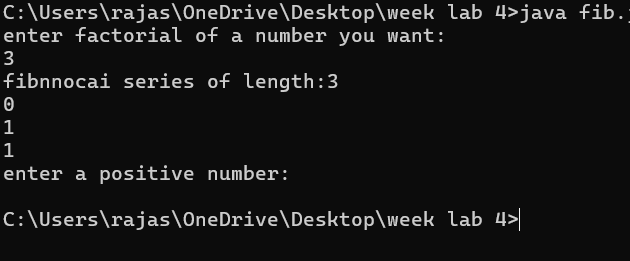
}

OUTPUT:for n=5



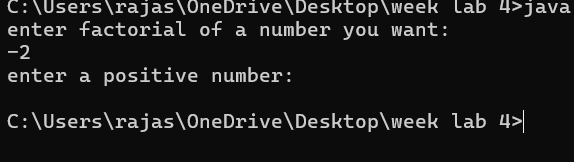
FOR n=3

Output:



FOR n=-2;

OUTPUT:



ERROR RECTIFICATION TABLE:

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | Impact | Solution |
| Resource leak | Scanneer input is not closed | May cause unnecessary memory usage | Add input.close();  After reading the input |

***Week 3***

*Lab 3:*

1.Write a code for method attributes of a car for three objects.

Aim: writing java code using method attributes.

Code:

class car

{

public String car\_color;

public String car\_brand;

public String fuel\_type;

public float mileage;

public void start()

{

System.out.println("Car starts");

}

public void stop()

{

System.out.println("Car stops");

}

public void service()

{

System.out.println("Car service");

}

public static void main(String [] args){

// object one creation

car car1= new car();

car1.car\_color="Red";

car1.car\_brand="BMW";

car1.fuel\_type="Petrol";

car1.mileage=62.5F;

//calling methods for object 1

car1.start();

car1.stop();

car1.service();

System.out.println("color of the car1 is "+car1.car\_color);

System.out.println("brand of the car1 is "+car1.car\_brand);

System.out.println("fuel type of the car1 is "+car1.fuel\_type);

System.out.println("mileage of the car1 is"+car1.mileage);

// object two creation

car car2= new car();

car2.car\_color="Blue";

car2.car\_brand="Audi";

car2.fuel\_type="Petrol";

car2.mileage=64.5F;

// calling methods for object 2

car2.start();

car2.stop();

car2.service();

System.out.println("color of the car2 is "+car2.car\_color);

System.out.println("brand of the car2 is "+car2.car\_brand);

System.out.println("fuel type of the car2 is "+car2.fuel\_type);

System.out.println("mileage of the car2 is"+car2.mileage);

//object three creation

car car3= new car();

car3.car\_color="Yellow";

car3.car\_brand="Benz";

car3.fuel\_type="Diesel";

car3.mileage=66.5F;

// calling methods for object 3

car3.start();

car3.stop();

car3.service();

System.out.println("color of the car3 is "+car3.car\_color);

System.out.println("brand of the car3 is "+car3.car\_brand);

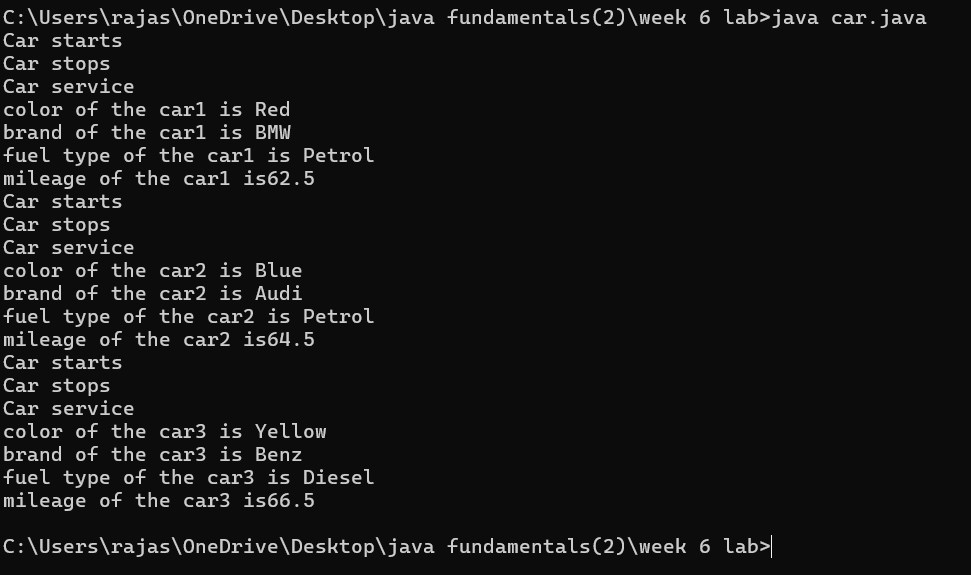
System.out.println("fuel type of the car3 is "+car3.fuel\_type);

System.out.println("mileage of the car3 is"+car3.mileage);

}

}

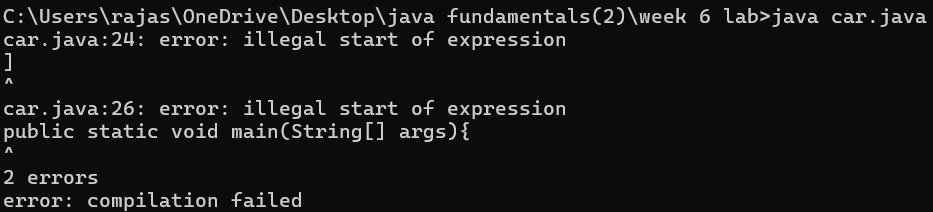
OUTPUT:



ERROR :

|  |  |  |  |
| --- | --- | --- | --- |
| Error type | Description | impact | Solution |
| Syntax error | Typo in the variable name mileage in the constructor | Compilation error ;  Constructor won’t work | Corrected to mileage |
| Mthod signature | Getter methods color,brand,fuel,mile had parameters but were expected to return instance values | Causes incorrect behaviour when  Calling methods | Removed parameters from method definition |

Error:



CLASS DIAGRAM :

|  |
| --- |
| Main |
| -car\_color:string  -car\_brand: string  - fuel\_type:string  -mileage:int |
| +color():String  +brand():String  +fuel():String  +mile():int |

IMPORTANT POINTS:

1. The car class has four attributes: Car\_color, Car\_brand, fuel\_type, and mileage.

2. It also has three methods: start(), service(), and stop().

3. The start(), service(), and stop() methods all print the same details about the car.

4. Each method prints the car's color, brand, fuel type, and mileage to the console

5. The main method creates three instances of the car class: car1, car2, and car3.

6. Each car object is assigned specific values for Car\_color, Car\_brand, fuel\_type, and mileage.

2.WRITING JAVA CODE FOR BANK ACCOUNT

Aim:writing a java code for the bank account details

Code:

class Bank\_Account {

private String accname;

private int acno;

private float balance;

public Bank\_Account(String accname, int acno, float balance) {

this.accname = accname;

this.acno = acno;

this.balance = balance;

}

public void withdraw(int amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawal of " + amount + " successful. Remaining balance: " + balance);

} else {

System.out.println("Insufficient balance for " + accname);

}

}

public void deposit(int amount) {

balance += amount;

System.out.println("Deposit of " + amount + " successful. Updated balance: " + balance);

}

public void displayDetails() {

System.out.println("Account Name: " + accname);

System.out.println("Account Number: " + acno);

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

}

public static void main(String[] args) {

Bank\_Account s = new Bank\_Account("Anil", 34571, 25000);

s.withdraw(20000);

s.deposit(2000);

s.displayDetails();

System.out.println();

Bank\_Account s1 = new Bank\_Account("Jeevan", 333226, 25000);

s1.withdraw(2500);

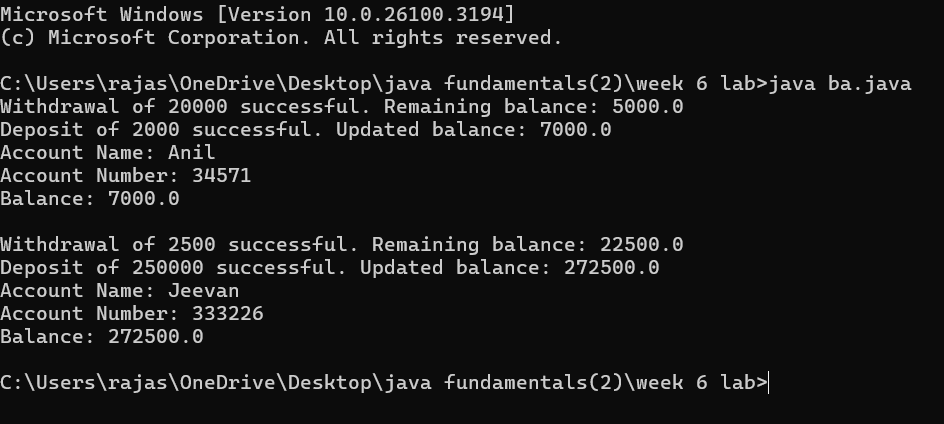
s1.deposit(250000);

s1.displayDetails();

}

}

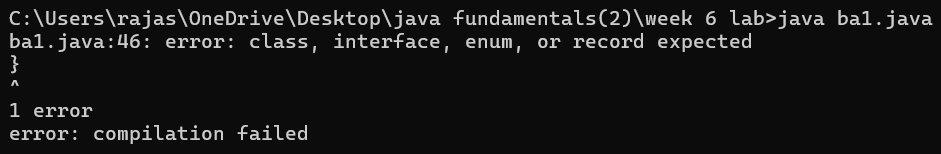
Output:



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. |

Error:



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:***

*LAB 4:*

1.*AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOUKD 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:

class Book {

private static String title;

private static String author;

private static int year;

public static void setDetails(String title, String author, int year) {

Book.title = title;

Book.author = author;

Book.year = year;

}

public static void displayDetails() {

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

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

System.out.println("Year: " + year);

System.out.println();

}

public static void main(String[] args) {

Book.setDetails("1984", "George Orwell", 1949);

System.out.println("Details of Book 1:");

Book.displayDetails();

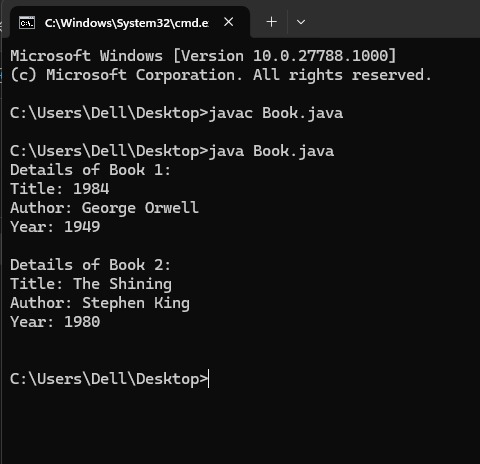
Book.setDetails("The Shining", "Stephen King", 1980);

System.out.println("Details of Book 2:");

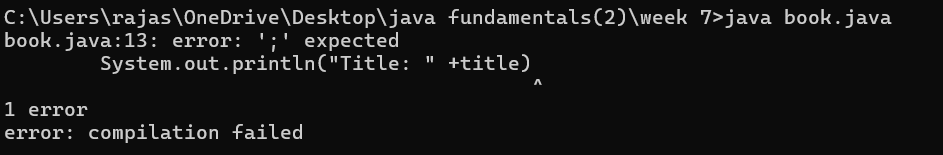
Book.displayDetails();

    }

}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();

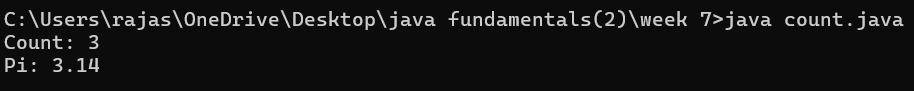
System.out.println("Count: " +count);

System.out.println("Pi: " +pi);

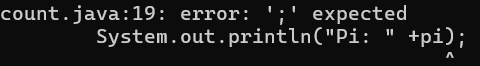
}

}

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 contains two static member variables: count and pi.

2. count is initialized to 0 and increments each time a new MyClass object is instantiated.

3. pi is a constant and immutable.

4. The constructor MyClass() increases the count variable by 1 every time a new MyClass object is created.

***WEEK 5:***

*LAB 5:*

***AIM:*** *Create a calculator using the operations including addition, subtraction Multiplication and division using multilevel inheritance and display the desired Output*

CODE:

class addition

{

public int add(int a, int b)

{

int addition = a+b;

return addition;

}

}

class subtraction extends addition

{

public int sub(int a, int b)

{

int subtraction = a-b;

return subtraction;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

int multiplication = a\*b;

return multiplication;

}

}

class division extends multiplication

{

public int div(int a,int b)

{

int division = a/b;

return division;

}

}

class calculator

{

public static void main(String args[])

{

division obj = new division();

System.out.println("Addition is:"+ obj.add(10,2));

System.out.println ("Subtraction is:"+obj.sub(8,4));

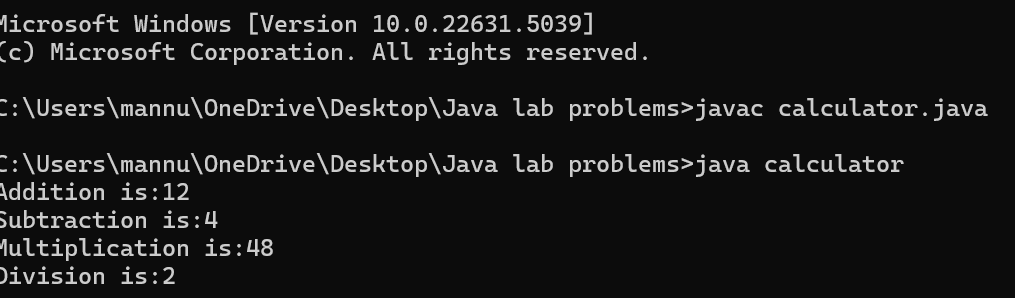
System.out.println("Multiplication is:"+obj.mult(12,4));

System.out.println("Division is:"+obj.div(8,4));

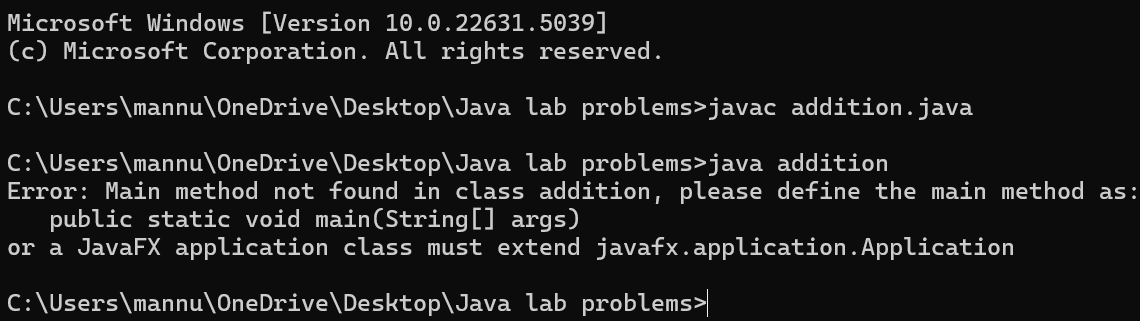
  }

}

**OUTPUT:**



**Negative Case:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error Type | Cause | Rectification |
| 1 | Constructor error | Invalid name to method | Defined class name |
| 2 | Syntax error | Expected ‘(‘ | Added parenthesis |
| 3 | Logical error | Incorrect arithmetic  operation | Correct operation  rectified |

**Important Points:**

**Inheritence:**

The concept of OOP where a class inherits the properties and behaviours from

Another class (parent class) which promotes code reusability and hieratchical relationships

**Multilevel Inheritence:**

This is a type of inheritance in which a class inherited from another class, and

That superclass, in turn, inherits from yet another class, creating a chain of

inheritence

**extends:**

The extends keyword defines the relation of child class with the parent class



**2.Aim:** A vehicle rental company wants to develop a system that maintains Information about different types of vehicles available for rent

The Company rents out cars, bikes and truck and they need a program to

Store details about each vehicle, such as brand and speed

Cars should have an additional property: number of doors

Bikes should have a property indicating whether they have gears or not

The system should also include a function to display details about each vehicle

And indicate when a vehicle is starting

Class diagram

**Code:**

class vehicle{

String brand;

int speed;

public vehicle(String brand,int speed){

this.brand=brand;

this.speed=speed;

}

public static void main(String[] args) {

car obj1=new car("ford",34,4);

bike obj2=new bike("hero",100,true);

truck obj3=new truck("tata",60,40);

}

}

class car extends vehicle{

int noofdoors;

public car(String brand, int speed,int noofdoors) {

super(brand, speed);

this.noofdoors=noofdoors;

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

System.out.println("Speed of car is:"+speed);

System.out.println("no of doors of car:"+noofdoors);

}

}

class bike extends vehicle{

boolean gears;

public bike(String brand,int speed,boolean gears){

super(brand, speed);

this.gears=gears;

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

System.out.println("Speed of bike is:"+speed);

System.out.println("Gears of bike:"+gears);

}

}

class truck extends vehicle{

int weight;

public truck(String brand,int speed,int weight){

super(brand,speed);

this.weight=weight;

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

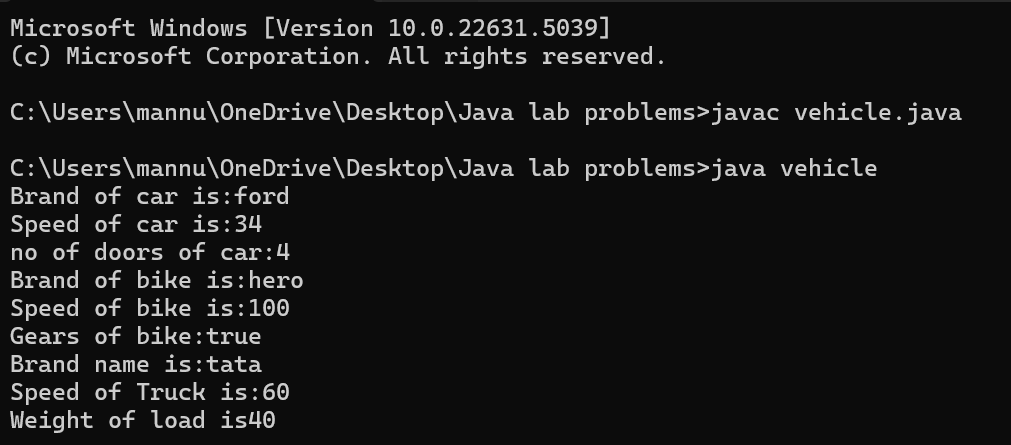
System.out.println("Speed of Truck is:"+speed);

System.out.println("Weight of load is"+weight);

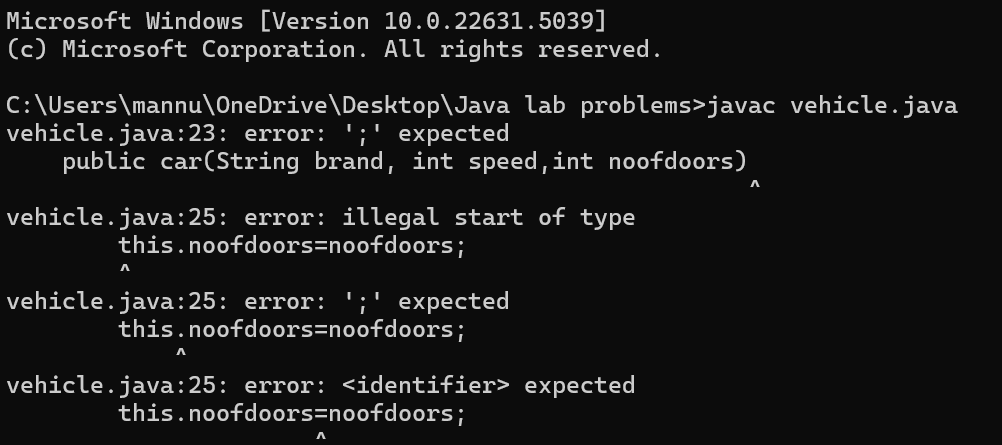
}

}

**OUTPUT:**



**NEGATIVE CASE:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 |  |  |  |

**Important Points:**

**Hierarchical Inheritence:**

This is a type of inheritance occurs when multiple subclasses inherit from a

Single parent class

***WEEK-6***

**1.Aim**: Write a java program to create a vehicle class with a method displayInfo().

Override this method in the car subclass to provide specific information a About car.

**CODE:**

class vehicle

{

public String car\_model;

public String car\_company;

public int car\_price;

void displayinfo()

{

System.out.println("yaswanth Car Agency");

}

}

class car extends vehicle

{

void displayinfo()

{

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

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

System.out.println("Car price is:"+car\_price);

}

}

class Overriding

{

public static void main(String args[])

{

car obj1 = new car();

obj1.car\_model="Swift";

obj1.car\_company="Suzuki";

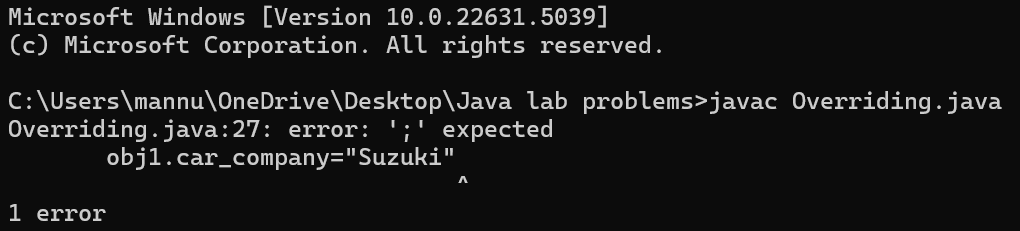
obj1.car\_price=1000000;

obj1.displayinfo();

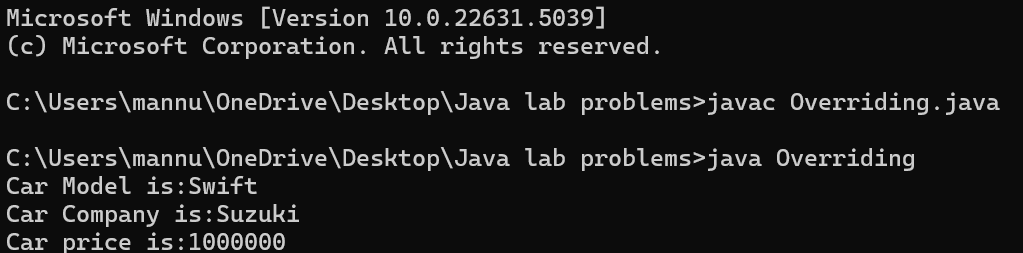
}

}

**NEGATIVE CASE:**

****

**OUTPUT:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

**2.**A college is developing an automated admission system that verifies Student eligibility for UG and PG programs. Each program has different Eligibility criteria based on the student’s percentage in their previous

Qualification:

UG admissions require a minimum of 60%

PG admissions require a minimum of 70%

**CODE:**

class Student

{

public String name;

public double percentage;

Student(String name, double percentage)

{

this.name = name;

this.percentage = percentage;

}

void eligibility()

{

System.out.println("Amrita Vishwa Vidyapeetham");

}

}

class UGStudent extends Student

{

UGStudent(String name, double percentage)

{

super(name,percentage);

}

void eligibility()

{

if (percentage>60)

{

System.out.println(name+" is selected to the college");

}

else

{

System.out.println(name+" is not matching the required criteria");

}

}

}

class PGStudent extends Student

{

PGStudent(String name, double percentage)

{

super(name,percentage);

}

void eligibility()

{

if(percentage>70)

{

System.out.println(name+" is selected to college");

}

else

{

System.out.println(name+"is not matching the required criteria");

}

}

}

class School

{

public static void main(String args[])

{

PGStudent obj1 = new PGStudent("Bhanu",90.0);

UGStudent obj2 = new UGStudent("Teja",80.0);

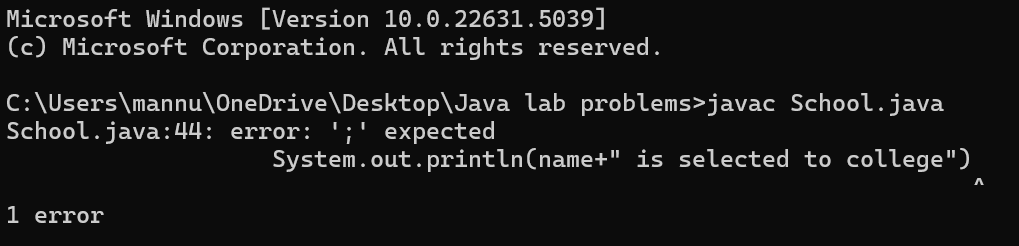
obj1.eligibility();

obj2.eligibility();

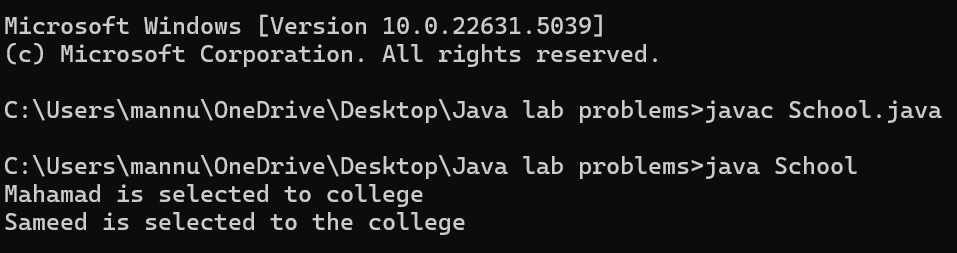
}

}

**NEGATIVE CASE:**



**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

**3**.Create a calculator class with overloaded methods 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 Overloading

{

public static void main(String args[]){

calculator c = new calculator();

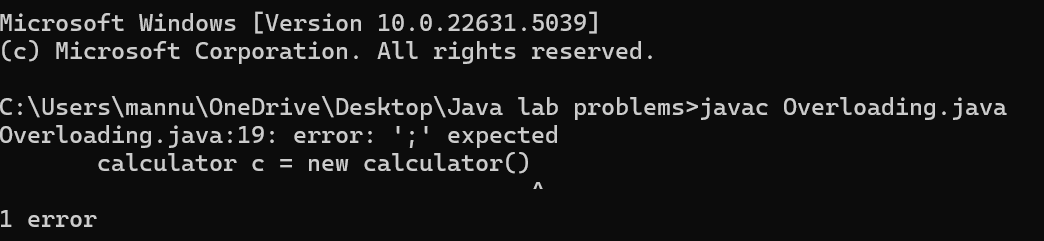
System.out.println("Addition of 3 and 4 is:"+c.add(3,4));

System.out.println("Addition of 2.2 and 4.4 is:"+c.add(2.2,4.4));

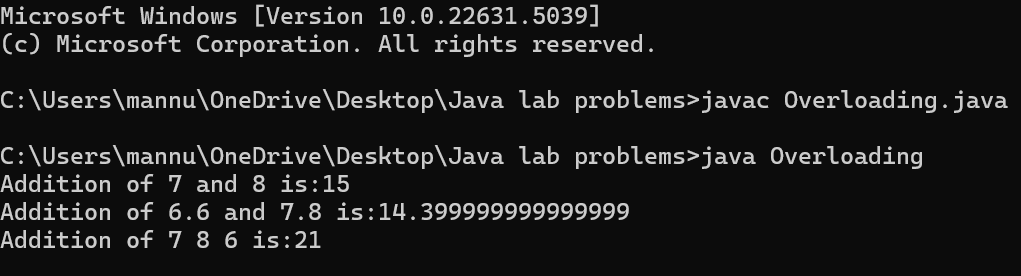
System.out.println("Addition of 2 4 6 is:"+c.add(2,4,6));

}}

**NEGATIVE CASE:**



**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

***WEEK-7***

**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() to make a specific sound for each animal.

**PROGRAM:**

 abstract class Animal {

    abstract void sound();

}

class Lion extends Animal {

    public void sound() {

        System.out.println("Lion Roars");

    }

}

class Tiger extends Animal {

    public void sound() {

        System.out.println("Tiger growls");

    }

}

public class AbstractDemo {

    public static void main(String[] args) {

        System.out.println("v.yaswanth, 24344, cse-c");

        Lion L = new Lion();

        Tiger T = new Tiger();

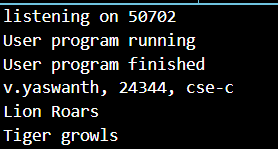
        L.sound();

        T.sound();

    }

}

**OUTPUT:**



**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | rectification |
| 2. | Runtime errror | Incorrect path | Using correct path |
| 3. | Syntax error | No semicoln | Using semicolon |

**IMPORTANT POINTS:**

1. Abstract Class (Animal)
   * Defines an abstract method sound() with no implementation.
   * Cannot be instantiated directly (cannot create Animal objects).
2. Concrete Subclasses (Lion and Tiger)
   * Inherit from Animal and must implement the sound() method.
   * Lion → Prints "Lion Roars".
   * Tiger → Prints "Tiger growls".

**2)AIM:**

Write a java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea() method .Create subclasses Sphere and Cube that extend the Shape3D class and Implements the respective methods to calculate the Volume and Surface area of each Shape.

**PROGRAM:**

abstract class Shape3D{

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D{

    int radius;

    Sphere(int radius){

    this.radius=radius;

    }

public double calculateVolume(){

    return (4.0/3.0)\*Math.PI\*Math.pow(radius,3);

}

public double calculateSurfaceArea(){

    return 4\*Math.PI\*Math.pow(radius,2);

}

}

class Cube extends Shape3D{

    int edge;

    Cube(int edge){

        this.edge=edge;

    }

public double calculateVolume(){

    return Math.pow(edge,3);

}

public double calculateSurfaceArea(){

    return 6\*Math.pow(edge,2);

}

}

public class Abstract{

    public static void main(String[] args) {

        System.out.println("v.yaswanth, 24344, cse-c");

        Sphere sphere=new Sphere(5);

        System.out.println("Volume of sphere is "+sphere.calculateVolume());

        System.out.println("Surface area of sphere is "+sphere.calculateSurfaceArea());

        Cube cube=new Cube(5);

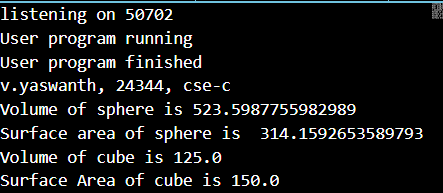
        System.out.println("Volume of cube is "+cube.calculateVolume());

        System.out.println("Surface Area of cube is "+cube.calculateSurfaceArea());

}

}

**OUTPUT:**



**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason for error** | **Rectification** |
| **1.** | No class | No class name declared | Created class named ‘Absract’ |
| **2.** | Syntax error | Not added keyword | Added keyword named ‘new’ |

**IMPORTANT POINTS:**

1. Abstract Class (Shape3D)

* Defines two abstract methods:
  + calculateVolume() → Computes the 3D shape's volume.
  + calculateSurfaceArea() → Computes the surface area.
* Cannot be instantiated directly (must be extended by subclasses).

2. Subclasses (Sphere and Cube)

* Extend Shape3D and must implement both abstract methods.
* Sphere Class:
  + Takes radius as input in the constructor.
  + Volume Formula: (4/3)πr³
  + Surface Area Formula: 4πr²
* Cube Class:
  + Takes edge (side length) as input in the constructor.
  + Volume Formula: edge³
  + Surface Area Formula: 6 × edge²

**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) 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 triangle of increasing numbers

In the main() method ,create objects of both subclasses and print the patterns for a given number of rows

**PROGRAM:**

abstract class PatternPrinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println("\n" + title + " Pattern:");

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

    }

}

class StarPattern extends PatternPrinter {

    @Override

    void printPattern(int n) {

        displayTitle("Star");

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

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

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

            }

            System.out.println();

        }

    }

}

class NumberPattern extends PatternPrinter {

    @Override

    void printPattern(int n) {

        displayTitle("Number");

        int num = 1;

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

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

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

            }

            System.out.println();

        }

    }

}

public class Pattern{

    public static void main(String[] args) {

       System.out.println("v.yaswanth, 24344, cse-c");

        int rows = 5;

        PatternPrinter starPrinter = new StarPattern();

        PatternPrinter numberPrinter = new NumberPattern();

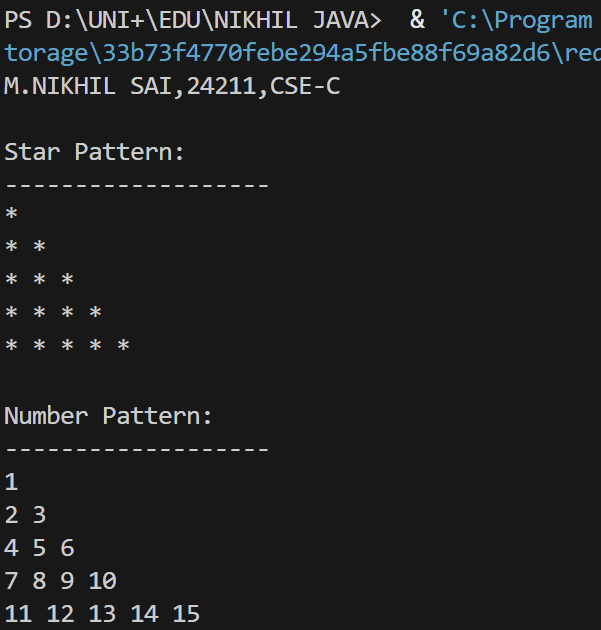
        starPrinter.printPattern(rows);

        numberPrinter.printPattern(rows);

    }

}

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error type** | **Error Rectification** |
| **1** | Logical error in wrong formula | Formula Rectified |
| **2** | Runtime error in incorrect path | Copied correct path |

**IMPORTANT POINTS:**

1. Abstract Class (PatternPrinter)

* Defines:
  + Abstract Method → printPattern(int n) (must be implemented by subclasses).
  + Concrete Method → displayTitle(String title) (prints a formatted pattern title).
* Cannot be instantiated directly (must be extended).

2. Subclasses (StarPattern and NumberPattern)

* Extend PatternPrinter and override printPattern(int n).
* StarPattern:
  + Prints a right-angled triangle of stars (\*)

NumberPattern:

* Prints a right-angled triangle of increasing numbers.

***WEEK-8***

**1)AIM:**

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

**PROGRAM:**

interface Shape{

    double getPerimeter();

}

class Rectangle implements Shape{

    double length;

    double width;

    Rectangle(double length,double width){

        this.length=length;

        this.width=width;

    }

    public double getPerimeter(){

        return 2\*(length+width);

    }

}

class Circle implements Shape{

    double radius;

    Circle(double radius){

        this.radius=radius;

    }

    public double getPerimeter(){

        return 2\*3.14\*radius;

    }

}

class Triangle implements Shape{

    double s1;

    double s2;

    double s3;

    Triangle(double s1,double s2,double s3){

        this.s1=s1;

        this.s2=s2;

        this.s3=s3;

    }

    public double getPerimeter(){

        return s1+s2+s3;

    }

}

public class Result{

    public static void main(String[] args){

        System.out.println("v.yaswanth, 24344, cse-c");

        Rectangle r=new Rectangle(10.0,6.0);

        Circle c=new Circle(5.0);

        Triangle t=new Triangle (5.0,6.0,8.0);

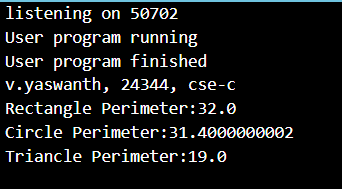
       System.out.println("Rectangle Perimeter:"+r.getPerimeter());

       System.out.println("Circle Perimeter:"+c.getPerimeter());

       System.out.println("Triancle Perimeter:"+t.getPerimeter());

    }

}

**OUTPUT:**

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| 1 | Syntax error in closing brackets | Need to close brackets |
| 2 | Wrong syntax in string | Rectified by changing the string |

**IMPORTANT POINTS:**

1. Interface (Shape)

* Defines one abstract method:
  + getPerimeter() → Must be implemented by all classes that implement Shape.
* Promotes polymorphism → Allows different shapes to be treated uniformly.

2. Implementing Classes (Rectangle, Circle, Triangle)

* Each implements Shape and provides its own getPerimeter() logic:
  + Rectangle → 2 × (length + width)
  + Circle → 2 × π × radius (approximated as 2 × 3.14 × radius)
  + Triangle → side1 + side2 + side3

3. Polymorphism & Method Overriding

* Same method name (getPerimeter()) behaves differently for each shape.
* Runtime binding ensures the correct method is called based on the object type.

**2)AIM:**

Write a java program to create an interface playable with a method play() that takes no arguments and return void. Create three classes Football, Volleyball, and Basketball that implement the playable interface and override the play() method to play the respective sports.

**PROGRAM:**

interface Playable {

    void play();

}

class Football implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Football: Kick the ball and score goals!");

    }

}

class Volleyball implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Volleyball: Serve, bump, set, and spike the ball over the net!");

    }

}

class Basketball implements Playable {

    @Override

    public void play() {

        System.out.println("Playing Basketball: Dribble, pass, and shoot the ball into the hoop!");

    }

}

public class Sports {

    public static void main(String[] args) {

        System.out.println("v.yaswanth, 24344, cse-c");

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

        football.play();

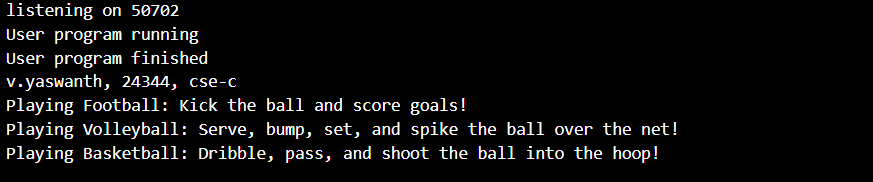
        volleyball.play();

        basketball.play();

    }

}

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ERROR** | **RECTIFICATION** |
| 1 | Java.lang.error | Execute with “extends” instead of “implements” |

**IMPORTANT POINTS:**

**1**. Interface Definition (Playable)

* Declares a single abstract method: void play()
* Serves as a contract that implementing classes must follow
* Enables polymorphism by allowing different implementations

2. Implementing Classes

* Football, Volleyball, Basketball all implement Playable
* Each provides its own specific implementation of play()
* Demonstrates how the same method can have different behaviors

3. Polymorphism in Action

* Objects are created using interface reference type (Playable)
* Method calls are resolved at runtime based on actual object type
* Example: football.play() calls Football's implementation