**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: v.karthik sai tej reddy**

**Verified By Roll No: av.sc.u4cse24343**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNo** | **Title** | **Date** | **Page No** | **Signature** |
| **WEEK 1** |  |  |  |  |
| 1 | Download and Install Java Software |  |  |  |
| 2 | Write a Java program to print the message “Welcome to Java Programming.” |  |  |  |
| **WEEK 2** |  |  |  |  |
| 1 | Write a Java program to calculate fibonacci of a number. |  |  |  |
| 2 | Write a Java program to calculate Simple interest. |  |  |  |
| 3 | Write a Java program to calculate factorial of a number. |  |  |  |
| 4 | Write a Java program to convert temperature from Celsius to Fahrenheit. |  |  |  |
| 5 | Write a java program to calculate the area of the triangle. |  |  |  |
| **WEEK 3** |  |  |  |  |
| 1 | Write a java program with  1. A class with name Car.  2.Create 4 attributes named car\_color , car\_brand ,fuel\_type ,mileage. 3.Create 3 methods named start() ,stop() ,service(). 4.Create 3 objects named car1 ,car2 ,car3. |  |  |  |
| 2 | Write a java program with 1.Create a class named Bankaccount. 2.Create a constructor. 3.Create 2 methods which are withdrawl() and deposit(). |  |  |  |
| **WEEK 4** |  |  |  |  |
| 1 | Write a java program with class named book. This class should contain various attributes such as title, author ,year of publication. It should also contain a constructor with parameters which initializers title ,author ,year of publication. Create a method which displays the details of the book .Display the details of 2 books. |  |  |  |
| 2 | Create a java program with class named myclass with a static variable count of int type ,initial value to zero and a constant variable "pi" of type double initialize 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 values of count and pi variables . Create 3 objects. |  |  |  |
| **WEEK 5** |  |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output. |  |  |  |
| 2 | A vehicle rental company wants to develop a system that maintains information about different types of vechicles 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( should be in super class)  1.cars should have an additional property: no.of doors 2.Bikes should have a property indicating whether they have gears or not. 3.The system should also include a function to display details about each vehicle and indicate when a vehicle is starting. 4.Every class should have a constructor. Question:  1.Which oops concept is used in the above program  2.If the company decides to add a new type of vehicle, Truck, how would you modify the program?  a.Truck should include an additional property capacity (in tons)  b.Create a showTruckdetails() method to display the truck’s capacity.  c.Write a constructor for Truck that initializes all properties Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details. |  |  |  |
| **WEEK 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. |  |  |  |
| 2 | A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.  1. UG admission require min of 60%  2. PG admission require min of 70% |  |  |  |
| 3 | 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. |  |  |  |
| 4 | Create a calculator class with overloaded methods to perform addition.  A. Add two integers  B. Add two double  C. Add three integer |  |  |  |
|  |  |  |  |  |
| **WEEK 7** |  |  |  |  |
| 1 | Write a java program to create an abstract class animal with an abstract method called sound( ).create subclasses lion and tiger that extends the animal class and implement the sound( ) method to make a specific sound for each animal. |  |  |  |
| 2 | Write a java program to create an abstract class shape 3D with an abstract method calculate volume( ) and calculate surface area( ). create subclasses sphere and cube that extends the shape 3D class and implement the respective methods to calculate the volume and surface area of each shape. |  |  |  |
| 3 | 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 a) and a concrete method to display the pattern title. * Implement two subclasses  1. Star Pattern – To print a right angled triangle with stars.   Number Pattern – To print a right angled triangle of increasing numbers. |  |  |  |
| **WEEK 8** |  |  |  |  |
| 1 | Write a java program to create an interface shape with the get perimeter ( ) method.create three classes rectangle ,circle and triangle that implements the shape interface implement the get perimeter( ) method for each of the three classes |  |  |  |
| 2 | 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 |  |  |  |

**WEEK 01**

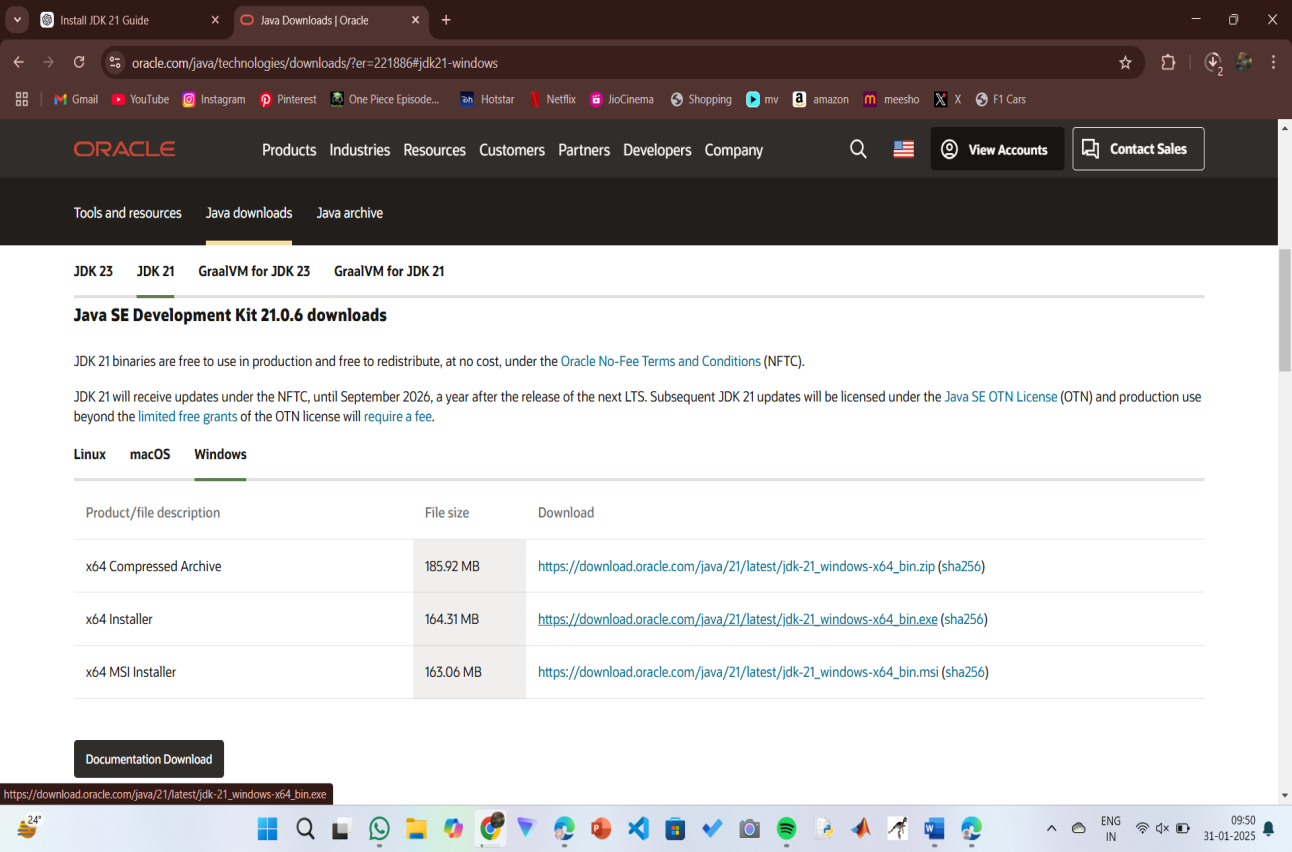
**PROGRAM-1:**

**AIM:** Download and Install Java Software

**PROCEDURE:**

**Step 1: Download JDK 23**

1. Open your web browser and go to the Oracle JDK Downloads page
2. Scroll down to the Java SE Development Kit 23 section.
3. Choose the Windows x64 Installer version.
4. Click on Download, then Wait for the download to complete



**Step 2:** **Install JDK 23**

1. Locate the downloaded jdk-23\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click Next on the setup wizard.
4. Choose the installation path (default is C:\Program Files\Java\jdk-21).
5. Click Next, then click Install.
6. Wait for the installation to complete.
7. Click Close once the installation is finished.



**Step 3: Setting up the path**

1) Go to “Windows C” Drive on Desktop

2) Choose Program Files, select Java, then JDK 23, then select Bin.

3) Select and copy the path at the address bar.



**Step 4: Open System Properties**

1. Press Windows + R, type sysdm.cpl , and click Ok-
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.



**Step 5: Set JAVA\_HOME**

1)Under System Variables, click New.

2)Set the Variable name as JAVA\_HOME.

3)Set Variable value as C:\Program Files\Java\jdk-23 (or your installation path).

4)Click OK.



**Step 6: Update PATH Variable**

1)In System Variables, find Path and click Edit.

2)Click New and add: C:\Program Files\Java\jdk-23\bin

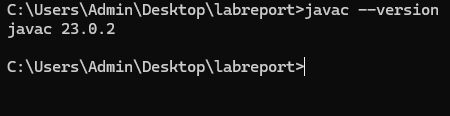
3)Click OK to save.



**Step 7:Verify Installation**

1. Open Command Prompt.
2. Type the following command: **java --version** and press Enter.

**OUTPUT:**



**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

Code :-

class first

{

    public static void main (String ar[ ] )

    {

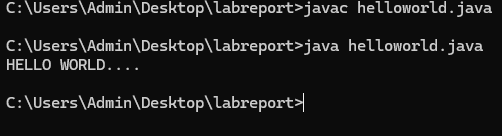
        //to print hello world

        System.out.println("HELLO WORLD....");

    }

}

**Output:**



**PROGRAM-3:**

**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

**STUDENT DETAILS:**

Code :-

class student

{

    public static void main(String ar[])

    {

        System.out.println("Name    : KARTHIK ");

        System.out.println("Class   : CSE-C");

        System.out.println("Roll.No : AV.SC.U4CSE24343");

        System.out.println("Branch  : COMPUTER SCIENCE ENGINEERING");

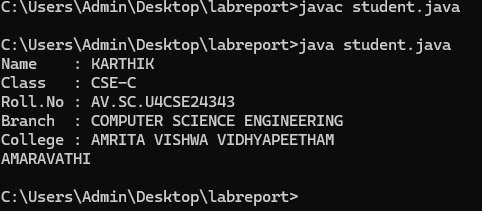
        System.out.println("College : AMRITA VISHWA VIDHYAPEETHAM");

        System.out.println("AMARAVATHI");

    }

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. writing small “S”in place of”S”   In system.out.println()  2)not giving strings to the name and scetion | 1. code is rectified by keeping capital “S” 2. Giving strings to name and section |

WEEK – 2

**Program – 1**

AIM :- Write a Java program to calculate fibonacci of a number

Code :-

import java.util.Scanner;

class fibonacci

{

 public static void main(String ar[])

 {

    Scanner sc = new Scanner(System.in);

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

    int x=sc.nextInt();

    int a=0,b=1,c;

    System.out.println(a);

    System.out.println(b);

    for(int i=0;i<=x;i++)

    {

        System.out.println(a+"");

        c=a+b;

        a=b;

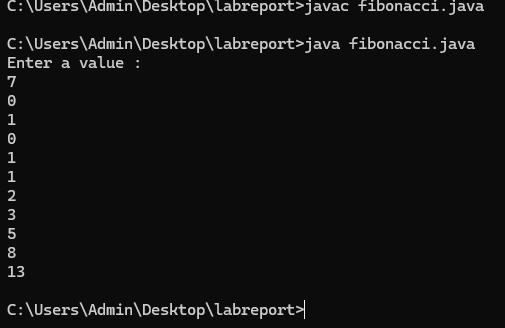
        b=c;

    }

 }

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Giving space between next and Double. 2. Not giving parenthesis after closing the input. | 1. Should not give space between next and Double. 2. We must put parenthesis after closing the input. |

**2.AIM:** Write a Java program to calculate Simple interest.

**CODE:**

import java.util.Scanner;

class SI

{

    public static void main(String ar[])

    {

        float si;

        Scanner input=new Scanner(System.in);

        System.out.println("Enter Principle amount : ");

        int p=input.nextInt();

        System.out.println("Enter Rate of Interest : ");

        float r=input.nextFloat();

        System.out.println("Enter Time in Years : ");

        int t=input.nextInt();

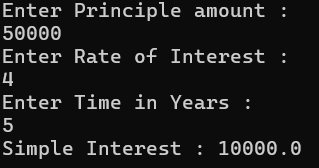
        si=(p\*t\*r)/100;

        System.out.println("Simple Interest : "+si);

    }

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Giving space between next and Double. 2. Not giving parenthesis after closing the input. | 1. Should not give space between next and Double. 2. We must put parenthesis after closing the input. |

**3.AIM:** Write a JAVA Code to find Factorial of a given number

**CODE:**

class factorial

{

    public static void main(String ar[])

    {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        int x=1;

        for(int i=1;i<=n;i++)

        {

            x=x\*i;

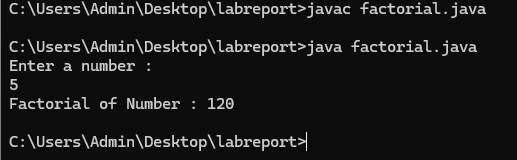
        }

        System.out.println("Factorial of Number : "+x);

    }

}

**OUTPUT:**

****

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While using for iteration, not giving the conditions correctly. 2. Declaring the data type as double instead of int. | 1. We should give iterative statements correctly. 2. We should give the data type as int for integers. |

**4.AIM:** : Write a JAVA Code to convert Celsius temperature into Fahrenheit temperature and vice versa.

**CODE:**

class temp

{

public static void main (String ar[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter 0 to convert Celcius into Fahrenheit Temperature.");

System.out.println("Enter 1 to convert Fahrenheit into Celcius Temperature.");

int n = sc.nextInt();

if (n==0)

{

System.out.println("Enter Temperature : ");

float x = sc.nextFloat();

double t=(x\*1.8)+32;

System.out.println("Converted Temperature : "+t);

}

else if (n==1)

{

System.out.println("Enter Temperature : ");

float x = sc.nextFloat();

double t=(x-32)/1.8;

System.out.println("Converted Temperature : "+t);

}

else

{

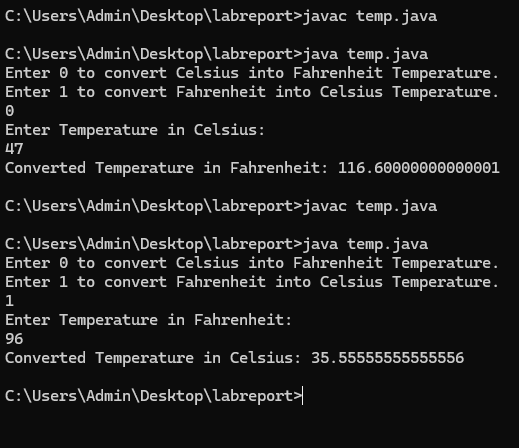
System.out.println("Enter Valid Input");

}

}

}

**OUTPUT:**

****

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While printing the variable not giving + sign. 2. Not closing the scanner. | 1. We should give correct indentation. 2. Closing the scanner is must. |

**5.AIM**: : Write a JAVA Code to calculate Area of Rectangle and Area of Triangle using herons formula.

**CODE:**

class area

{

public static void main(String ar[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter Length of Rectangle : ");

float x = sc.nextFloat();

System.out.println("Enter Breadth of Rectangle : ");

float y = sc.nextFloat();

float z=x\*y;

System.out.println("Area of Rectangle : "+z);

System.out.println(“To find Area of Triangle”);

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

double a=sc.nextDouble

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

double b=sc.nextDouble();

System.out.println("c : ");

double c=sc.nextDouble();

double s = a+b+c/2;

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

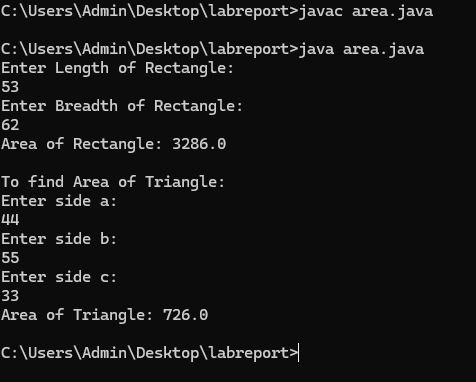
System.out.println("Area of Triangle : "+s);

input.close();

}

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | Code rectification |
| 1. While printing the variable not giving + sign. 2. Not closing the scanner. | 1. We should give correct indentation. 2. Closing the scanner is must. |

**WEEK -3:**

**AIM:** : Write a JAVA Code to create java program with following instructions

1.Create a class with name car

2. Create four attributes named car\_color ,Car\_brand,fuel\_type,mileage

3. Create three methods named start(), stop(). Service()

4. Create three objects named car1,car2 and car3

**CODE:**

import java.util.\*;

class car

{

    public String Car\_color;

    public String Car\_brand;

    public String fuel\_type;

    public int mileage;

   public void start()

   {

       System.out.println("Car Started:");

       System.out.println("Car color is :"+Car\_color);

       System.out.println("Car Brand is:"+Car\_brand);

       System.out.println("Car fuel type is:"+fuel\_type);

       System.out.println("Car mileage is:"+mileage);

   }

    public void service()

   {

       System.out.println("Car Started:");

       System.out.println("Car color is :"+Car\_color);

       System.out.println("Car Brand is:"+Car\_brand);

       System.out.println("Car fuel type is:"+fuel\_type);

       System.out.println("Car mileage is:"+mileage);

   }

    public void stop()

   {

       System.out.println("Car Started:");

       System.out.println("Car color is :"+Car\_color);

       System.out.println("Car Brand is:"+Car\_brand);

       System.out.println("Car fuel type is:"+fuel\_type);

       System.out.println("Car mileage is:"+mileage);

   }

public static void main(String args[])

{

    car car1 = new car();

    car1.Car\_color = "Pink";

    car1.Car\_brand = "Audi";

    car1.fuel\_type = "Deisel";

    car1.mileage = 100;

   car1.start();

    car car2 = new car();

    car2.Car\_color = "Black";

    car2.Car\_brand = "Tesla";

    car2.fuel\_type = "EV";

    car2.mileage = 200;

    car2.stop();

    car car3 = new car();

    car3.Car\_color = "Blue";

    car3.Car\_brand = "BMW";

    car3.fuel\_type = "Petrol";

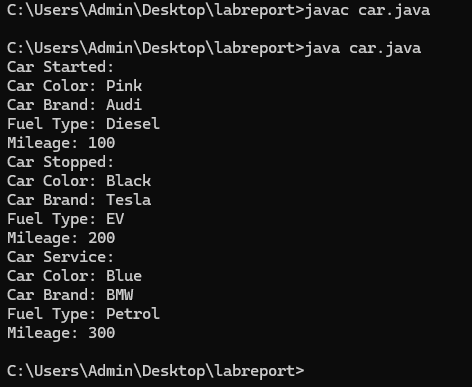
    car3.mileage = 300;

    car1car3.service();

 }

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| car.java:44: error: ';' expected  car1.start() | By giving semicolon at last to statement. |

**Class diagram:**

|  |
| --- |
| **car**  **-----------------------**  **-car\_color:string**  **-car\_brand:string**  **-fuel\_type:string**  **-milage:double**  **----------------------**  **+start():void**  **+stop():void**  **+service():void** |

**Program – 2**

**AIM:**  Write a JAVA Code to create a class bankAccount with methods deposit() and withdrawl

**CODE:**

public class BankAccount {

    public double balance;

    public BankAccount(double initialBalance) {

        balance = initialBalance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposited " + amount + ". New balance is " + balance);

        } else {

            System.out.println("Deposit amount must be positive.");

        }

    }

    public void withdraw(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("Withdrew " + amount + ". New balance is " + balance);

        } else {

            System.out.println("Invalid withdrawal amount. Check your balance.");

        }

    }

    public static void main(String[] args) {

        BankAccount account = new BankAccount(100); // Initial balance of 100

        account.deposit(100000);

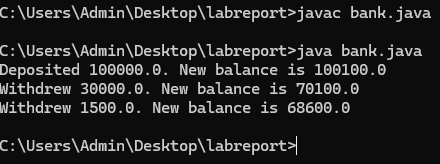
        account.withdraw(30000);

        account.withdraw(1500); // Invalid withdrawal

    }

}

**OUTPUT:**

****

ERROR TABLE :

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Parenthesis is missing | Added parenthesis to the statement. |

Class diagram:

|  |
| --- |
| BankAccount  ----------------------------------------------------------  -balance: double  ----------------------------------------------------------  +BankAccount(intialBalance: double)  +deposit(amount: double):void  +withdraw(amount: double):void |

**WEEK -4:**

**Program – 1**

**AIM:**  Write a java program with class named book. A class should contain various attributes such as title, author, year of publication. It should also contain a constructor with parameters which initializes title, author and year of publication. Create a method which displays the details of the book. Display the details of two books.

**CODE:**

class addition

{

public int add(int a, int b)

{

return a+b;

}

}

class subtraction extends addition{

public int sub(int a, int b)

{

return a-b;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

return a\*b;

}

}

class division extends multiplication

{

public int div(int a,int b)

{

return a/b;

}

}

class calculator2

{

public static void main(String args[])

{

division d = new division();

System.out.println("Addition is:"+d.add(8,2));

System.out.println("Subtraction is:"+d.sub(8,2));

System.out.println("Mult"+d.mult(5,6));

System.out.println("Divison is:"+d.div(4,2));

}

}

public class book

{

    String title;

    String author;

    int year;

    // Constructor

    public book(String title, String author, int year) {

        this.title = title;

        this.author = author;

        this.year = year;

    }

    // Method

    public void displayDetails() {

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

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

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

        System.out.println();

    }

    public static void main(String[] args) {

        //objects

        book book1 = new book("book 1", "author 1", 1999);

        book book2 = new book("book 2", "author 2", 2000);

        //book details

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

        book1.displayDetails();

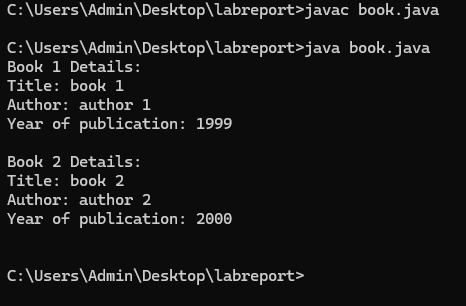
        System.out.println("Book 2 Details:");

        book2.displayDetails();

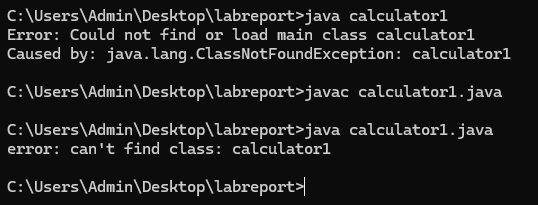
    }

}

**OUTPUT:**

****

Negative case:

****

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| book.java:28: error: cannot find symbol  book1.display(); | By adding,  book1.displayDetails(); |

Class diagram:

|  |
| --- |
| book |
| - title: String  - author: String  - year: int |
| + Book(title: String, author: String, year: int)  + displayDetails(): void |
| + main(args: String[]): void |

**Program – 2**

**AIM:**  Create a java program with class name “myclass” with static variable count of int type. Initialize it to zero and a constant variable pi of type double. Initialize pi to 3.14 as attributes of 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. Create three objects.

**CODE:**

public class myclass

{

    static int count = 0;

    final double PI = 3.14;

    myclass() {

        count++;

    }

    void display() {

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

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

    }

    public static void main(String[] args) {

        myclass obj1 = new myclass();

        myclass obj2 = new myclass();

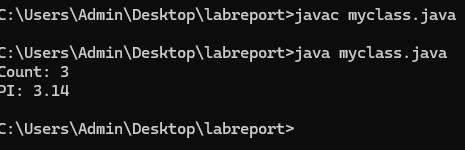
        myclass obj3 = new myclass();

        obj3.display();

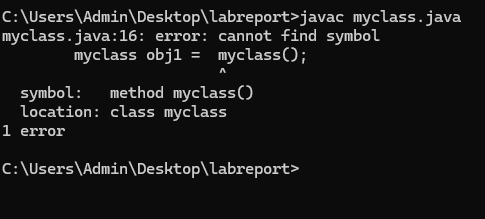
    }

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| myclass.java:7: error: illegal start of expression  count+; | By adding,  count++; |

Class diagram:

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

IMPORTANT POINTS:

1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

**WEEK -5:**

1. **AIM:**  create a calculator using the operation including adding, subtraction,multiplication,and division using multilevel inheritance and display desired output

every class one method (int a ,int b)

pass values as parameter during method call

**CODE:**

class addition

{

public int add(int a, int b)

{

return a+b;

}

}

class subtraction extends addition{

public int sub(int a, int b)

{

return a-b;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

return a\*b;

}

}

class division extends multiplication

{

public int div(int a,int b)

{

return a/b;

}

}

class calculator2

{

public static void main(String args[])

{

division d = new division();

System.out.println("Addition is:"+d.add(8,2));

System.out.println("Subtraction is:"+d.sub(8,2));

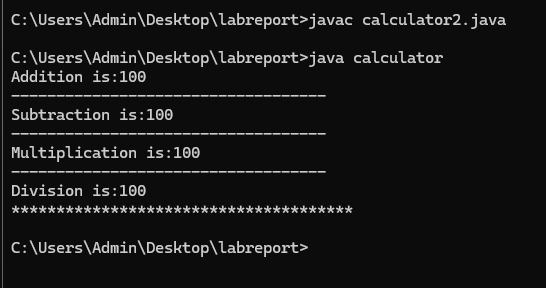
System.out.println("Mult"+d.mult(5,6));

System.out.println("Divison is:"+d.div(4,2));

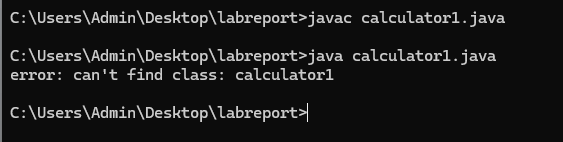
}

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| can't find main(String[]) method in class: Calculator | By adding this method |

Classs diagram:

|  |
| --- |
| Class : addition |
| + add( int a ,int b) |
|  |

|  |
| --- |
| Class : subtraction |
| + sub( int a ,int b) |

|  |
| --- |
| Class : multiplication |
| + multi( int a ,int b) |

|  |
| --- |
| Class : divission |
| + div( int a ,int b) |

important points :

1. used contructor and ssigned values as int a and int b.
2. used multilevel unheritance .

used this keyword to assign values

**Program – 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

**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("maruti",133,7);

bike obj2=new bike("unicorn",75,true);

truck obj3=new truck("zimban",120,16);

}

}

class car extends vehicle{

int noofdoors;

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

super(brand, speed);

this.noofdoors=noofdoors;

System.out.println(brand);

System.out.println(speed);

System.out.println(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);

System.out.println(speed);

System.out.println(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);

System.out.println(speed);

System.out.println(weight);

}

public static void main(String[] args) {

car obj1=new car("maruti",133,7);

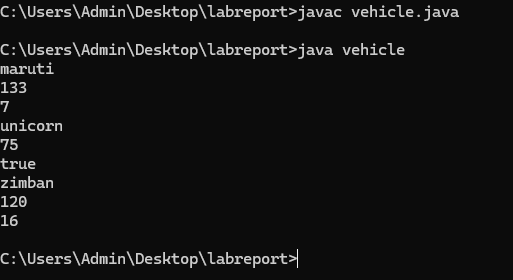
bike obj2=new bike("unicorn",75,true);

truck obj3=new truck("zimban",120,16);

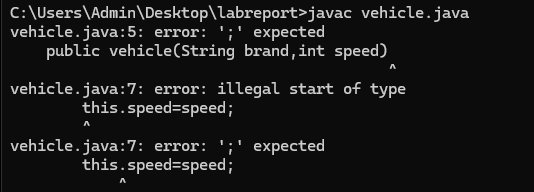
}

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Vehicle.java:28: error: cannot find symbol  c1.showCarDetails; | By adding  c1.showCarDetails(); |

Class Diagram :-

+----------------------------+

| Vehicle |

+----------------------------+

| - brand: String |

| - speed: int |

+----------------------------+

| +Vehicle(String, int) |

| +start(): void |

+----------------------------+

▲

------------------------------|------------------------------

| | |

+---------------+ +---------------+ +------------------+

| Car | | Bike | | Truck |

+---------------+ +---------------+ +------------------+

| - doors: int | | - hasGears:| | - capacity: int|

| | | boolean | | |

+-----------------+ +---------------+ +------------------+

| +Car(...) | | +Bike(...) | | +Truck(...) |

| +showCar...()| | +showBike...()| | +showTruck...()|

+---------------+ +------------------+ +-------------------+

**Important points:**

1.Inheritance:

Obtaining the qualities from parent class.

2.hirechial inheritance: different child class obtaing qualities from one parent.

**WEEK -6:**

**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 Car extends Vehicle2 {

Car(String brand, int speed, int capacity) {

super(brand, speed, capacity);

}

public void displayInfo() {

System.out.println("Car Details:");

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

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

System.out.println("Speed: " + speed + " km/h");

System.out.println("Capacity: " + capacity + " people");

}

}

public class vehicle2 {

public static void main(String[] args) {

Vehicle2 vehicle = new Vehicle2("Luna", 78, 5);

vehicle.displayInfo();

System.out.println();

Car car = new Car("bonda", 120, 7);

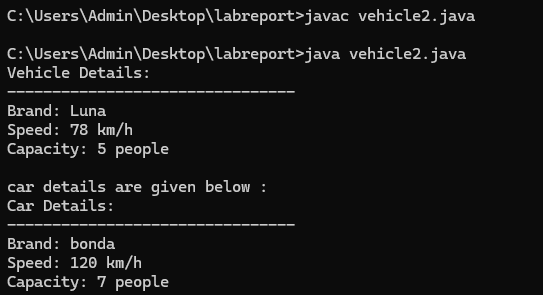
System.out.println("car details are given below :");

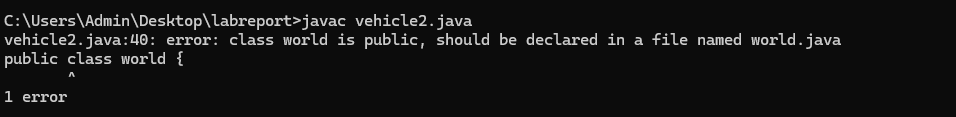
car.displayInfo();

}

}

**OUTPUT:**



Negative case:

Class Diagram :-

+---------------------------+

| Vehicle |

+----------------------------+

| +displayInfo(): void |

+---------------------------+

▲

|

+---------------------------+

| Car |

+---------------------------+

| +displayInfo(): void |

+--------------------------+

+------------------+

| info |

+------------------+

| +main(String[]): void |

+------------------+

Important points:

1. **Inheritance** allows the Car class to reuse the Vehicle class functionality without repeating code.

2. **Method Overriding** enables the Car class to provide its own implementation of the displayInfo() method.

3. **Polymorphism** makes the code flexible, allowing different classes (e.g., Car, Bike) to provide customized behavior for displayInfo().

**2.**

**AIM**: 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 University {

String name;

int percentile;

University(String name, int percentile) {

this.name = name;

this.percentile = percentile;

}

public void office() {

System.out.println("Admission criteria are mentioned below.");

}

}

class PG extends University {

PG(String name, int percentile) {

super(name, percentile);

}

@Override

public void office() {

if (percentile >= 70) {

System.out.println(name + " is eligible for PG admission.");

} else {

System.out.println(name + " is not eligible for PG admission.");

}

}

}

class UG extends University {

UG(String name, int percentile) {

super(name, percentile);

}

@Override

public void office() {

if (percentile >= 60) {

System.out.println(name + " is eligible for UG admission.");

} else {

System.out.println(name + " is not eligible for UG admission.");

}

}

}

public class Admissions {

public static void main(String[] args) {

UG ugStudent = new UG("krishna", 95);

PG pgStudent = new PG("manoj", 96);

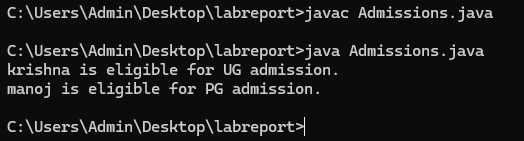
ugStudent.office();

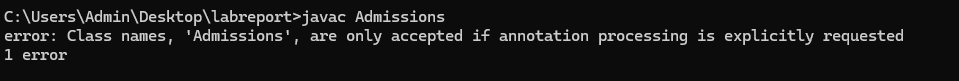
pgStudent.office();

}

}

**OUTPUT:**



Negative case:

Class Diagram :-

+---------------------+

| College |

+---------------------+

| + main(String[]): void |

+---------------------+

| - Scanner object used |

| - String a (program) |

| - double b (percentage)|

| - if-else logic for eligibility |

+---------------------+

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| College.java:21: error: ';' expected  elif (a.equals("PG")) {  ^  College.java:28: error: 'else' without 'if'  else {  ^ | Replacing elif with else if. |

IMPORTANT POINTS:

1.**Inheritance** allows PG and UG classes to reuse the University class’s attributes and constructor.

2. **Method Overriding** customizes the office() method in PG and UG classes to implement specific admission criteria.

3. **Polymorphism** enables the same office() method to behave differently for PG and UG objects based on their typ

3.

**AIM**: Create a Shape class with a method calculateArea() that is overloaded forDifferent shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle

**CODE:**

class Shape {

public void calculateArea(int length, int width) {

int area = length \* width;

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

}

public void calculateArea(double base, double height) {

double area = 0.5 \* base \* height;

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

}

public void calculateArea(int side) {

int area = side \* side;

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

}

}

class Circle extends Shape {

public void calculateArea(double radius) {

double area = 3.14 \* radius \* radius;

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

}

}

public class calculateArea {

public static void main(String[] args) {

Shape shape = new Shape();

shape.calculateArea(5, 45);

shape.calculateArea(7.0, 34.0);

shape.calculateArea(7);

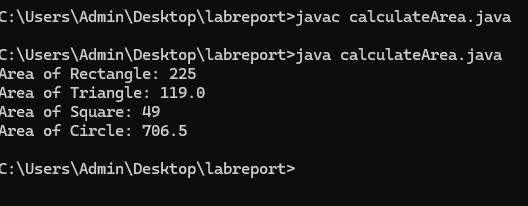
Circle circle = new Circle();

circle.calculateArea(15.0);

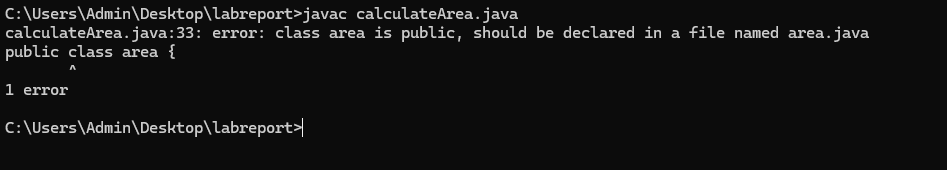
}

}

**OUTPUT:**



Negative case:



Class Diagram :-

+-----------------------------+

| Shape |

+-----------------------------+

| + calcArea(double side) |

| + calcArea(double length, double breadth) |

+-----------------------------+

▲

|

+-----------------------------+

| Circle |

+-----------------------------+

| + calcArea(double radius) |

+-----------------------------+

+-----------------------------+

| Area |

+-----------------------------+

| + main(String[] args): void |

+-----------------------------+

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Parenthesis is missing in line 16 | Adding parenthesis to the statement. |

IMPORTANT POINTS :

1. Method Overloading allows the Shape class to calculate areas for different shapes (rectangle, triangle, square) using multiple calculateArea() methods with different parameters.
2. Method Overriding in the Circle class provides a custom implementation of calculateArea() for calculating the area of a circle.
3. Polymorphism enables objects of different types (e.g., Shape, Circle) to call the appropriate version of calculateArea() based on the object type and input parameters.

**4.**

**AIM**:Create a calculator class with overloaded methods to perform addition

1. Add two integers 2. Add two doubles 3. Add three integers

**CODE:**

public class Addition {

    void addition(int a, int b) {

        System.out.println("Addition of two integers :"+(a+b));

    }

    void addition(double a, double b){

        System.out.println("Addition of two double numbers :"+(a+b));

    }

    void addition(int a,int b,int c){

        System.out.println("Addition of three integers :"+(a+b+c));

    }

}

class overload{

    public static void main(String[] args) {

    Addition obj = new Addition();

    obj.addition(10, 15);

    obj.addition(20, 15, 15);

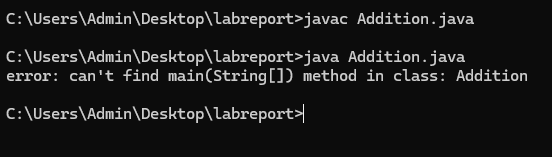
    obj.addition(55.5, 44.5);

}

}

**OUTPUT:**

Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Missing semicolon in line 14 | By adding semicolon to that line |

IMPORTANT POINTS :-

1. The program demonstrates method overloading — multiple methods with the same name but different parameters.
2. The Addition class has three addition() methods handling different data types and number of arguments.
3. Overloading improves code readability and reusability by grouping similar operations under one method name.
4. The main() method in the overload class creates an object and calls all overloaded versions of addition().

**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 extends the animal class and implement the sound( ) method to make a specific sound for each animal.

Class Diagram :-

+----------------+

| Animal | <<abstract>>

+----------------+

| +sound(): void | <<abstract>>

+----------------+

/\_\

|

-------------------------

| |

+-------------+ +--------------+

| Lion | | Tiger |

+-------------+ +--------------+

| +sound():void| | +sound():void|

+-------------+ +--------------+

**CODE:**

abstract class Animal {

abstract void sound();

}

class Lion extends Animal {

@Override

void sound() {

System.out.println("Lion roars!");

}

}

class Tiger extends Animal {

@Override

void sound() {

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

}

}

class AnimalSoundTest {

public static void main(String[] args) {

Animal lion = new Lion();

Animal tiger = new Tiger();

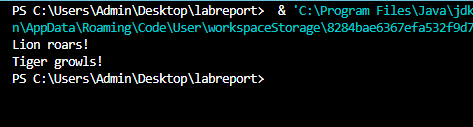
lion.sound();

tiger.sound();

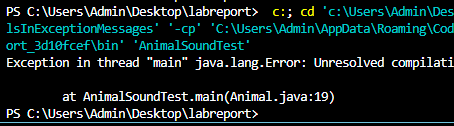
}

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Trying to create an object of Animal, which is an abstract class  Forgot to override the sound() method in Lion class | Instantiate a subclass (Lion or Tiger) instead of Animal  Add @Override method in Lion that implements sound() |

IMPORTANT POINTS :-

 Abstract Class:

* Animal is declared as abstract which means it cannot be instantiated directly.
* It can contain both abstract and non-abstract methods (though we only use an abstract method here).

 Abstract Method:

* sound() is abstract in Animal, meaning all subclasses must provide their own implementation.

 Polymorphism:

* In main, we use the superclass reference (Animal) to refer to subclass objects (Lion, Tiger). This allows runtime polymorphism.

**2.AIM**: Write a java program to create an abstract class shape 3D with an abstract method calculate volume( ) and calculate surface area( ). create subclasses sphere and cube that extends the shape 3D class and implement the respective methods to calculate the volume and surface area of each shape.

Class Diagram :-

+----------------------+

| Shape3D | <<abstract>>

+----------------------+

| +calculateVolume(): double |

| +calculateSurfaceArea(): double |

+----------------------+

/\_\

|

-----------------------------------

| |

+------------------+ +------------------+

| Sphere | | Cube |

+------------------+ +------------------+

| - radius: double | | - side: double |

+------------------+ +------------------+

| +calculateVolume() | +calculateVolume()

| +calculateSurfaceArea() | +calculateSurfaceArea()

+------------------+ +------------------+

**CODE:**

abstract class Shape3D {

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

int radius;

Sphere(int radius) {

this.radius = radius;

}

@Override

public double calculateVolume() {

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

}

@Override

public double calculateSurfaceArea() {

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

}

}

class Cube extends Shape3D {

int edge;

Cube(int edge) {

this.edge = edge;

}

@Override

public double calculateVolume() {

return Math.pow(edge, 3);

}

@Override

public double calculateSurfaceArea() {

return 6 \* Math.pow(edge, 2);

}

}

class Main {

public static void main(String[] args) {

Sphere s = new Sphere(4);

System.out.println("Sphere Volume: " + s.calculateVolume());

System.out.println("Sphere Surface Area: " + s.calculateSurfaceArea());

Cube c = new Cube(3);

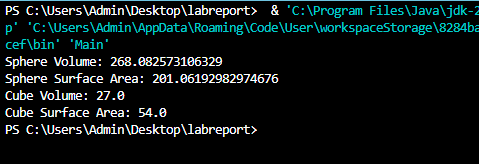
System.out.println("Cube Volume: " + c.calculateVolume());

System.out.println("Cube Surface Area: " + c.calculateSurfaceArea());

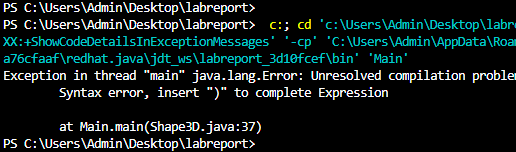
}

}

**OUTPUT:**

****

Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Forgot to implement all abstract methods in subclass  Method signature mismatch or missing () | Make sure Sphere and Cube implement **both** calculateVolume() and calculateSurfaceArea()  Ensure method call uses correct syntax: calculateVolume() not calculateVolume |

IMPORTANT POINTS :-

**Use of Abstract Class**

* Shape3D is an abstract class with two abstract methods:
  + calculateVolume()
  + calculateSurfaceArea()
* It **cannot be instantiated** directly.
* Forces subclasses like Sphere and Cube to provide **specific implementations** of volume and surface area calculations.

**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 a) and a concrete method to display the pattern title.
* Implement two subclasses

Star Pattern – To print a right angled triangle with stars.

Number Pattern – To print a right angled triangle of increasing numbers.

Class Diagram :-

+--------------------------------+

| PatternPrinter | (abstract class)

+--------------------------------+

| +displayTitle(title): void |

| +printPattern(a): void | (abstract)

+--------------------------------+

/\

|

---------------------------------

| |

+---------------------+ +----------------------+

| StarPattern | | NumberPattern |

+---------------------+ +----------------------+

| +printPattern(a): void | | +printPattern(a): void |

+---------------------+ +----------------------+

**CODE:**

abstract class PatternPrinter {

void displayTitle(String title) {

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

}

abstract void printPattern(int a);

}

class StarPattern extends PatternPrinter {

void printPattern(int a) {

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

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

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

}

System.out.println();

}

}

}

class NumberPattern extends PatternPrinter {

void printPattern(int a) {

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

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

System.out.print(i);

}

System.out.println();

}

}

}

public class Main {

public static void main(String[] args) {

PatternPrinter pattern1 = new StarPattern();

pattern1.displayTitle("Star Triangle");

pattern1.printPattern(5);

System.out.println();

PatternPrinter pattern2 = new NumberPattern();

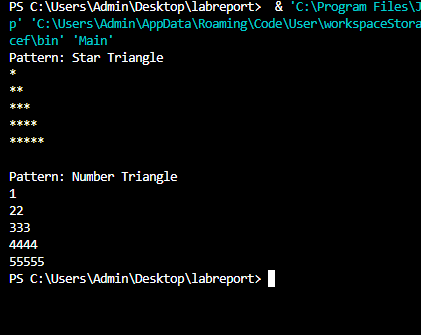
pattern2.displayTitle("Number Triangle");

pattern2.printPattern(5);

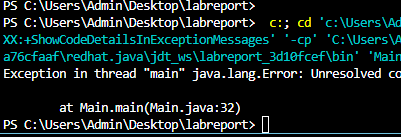
}

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Placing comma instead of semicolon. | Rewriting the code using semicolon. |

IMPORTANT POINTS :-

1. PatternPrinter is an abstract class with a concrete and an abstract method.

2. Subclasses StarPattern and NumberPattern implement the printPattern method differently.

3. The Main class demonstrates polymorphism by using the abstract class reference.

**WEEK -8:**

**1.AIM**: Write a java program to create an interface shape with the get perimeter ( ) method.create three classes rectangle ,circle and triangle that implements the shape interface implement the get perimeter( ) method for each of the three classes

Class Diagram :-

+-------------+

| Shape | <interface>

+-------------+

| +getPerimeter(): double |

+-------------+

/ | \

| |

+----------------+ +----------------+ +----------------+

| Rectangle | | Circle | | Triangle |

+----------------+ +----------------+ +----------------+

| -length: double | | -radius: double| | -side1: double |

| -width: double | +----------------+ | -side2: double |

+----------------+ | -side3: double |

| +getPerimeter() | +----------------+

+----------------+ | +getPerimeter() |

+----------------+

**CODE:**

interface Perimeter {

double getPerimeter();

}

class Rectangle implements Perimeter {

public int l;

public int b;

public Rectangle(int l, int b) {

this.l = l;

this.b = b;

}

public double getPerimeter() {

return 2 \* (l + b);

}

}

class Circle implements Perimeter {

public int r;

public Circle(int r) {

this.r = r;

}

public double getPerimeter() {

return 2 \* 3.14 \* r;

}

}

class Triangle implements Perimeter {

public int a, b, c;

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

this.a = a;

this.b = b;

this.c = c;

}

public double getPerimeter() {

return a + b + c;

}

}

class Shape2 {

public static void main(String[] args) {

Perimeter r = new Rectangle(3, 6);

System.out.println("The perimeter of the given Rectangle: " + r.getPerimeter());

Perimeter c = new Circle(4);

System.out.println("The perimeter of the given Circle: " + c.getPerimeter());

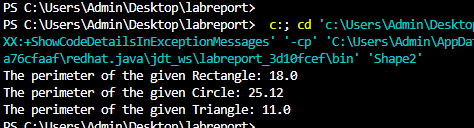
Perimeter t = new Triangle(2, 4, 5);

System.out.println("The perimeter of the given Triangle: " + t.getPerimeter());

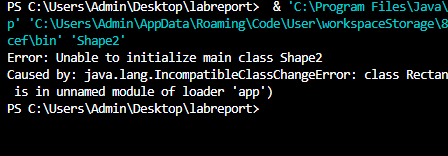
}

}

**OUTPUT:**



Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Forgot to define the Shape interface before implementing it.  Forgot to implement the getPerimeter() method in one of the classes. | Ensure interface Shape is defined before other classes.  Implement getPerimeter() in each class that implements Shape. |

IMPORTANT POINTS :-

 **Interface Basics**:

* An interface in Java defines a contract: any class that implements it must provide implementations for all its methods.
* Shape interface has only one method: getPerimeter().

 **Encapsulation**:

* All properties like length, width, etc., are private. This is good practice to enforce encapsulation.

**2.AIM**: 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

Class Diagram :-

+---------------+

| Playable | <interface>

+---------------+

| +play(): void |

+---------------+

/ | \

/ | \

/ | \

+-------------+ +---------------+ +---------------+

| Football | | Volleyball | | Basketball |

+-------------+ +---------------+ +---------------+

| +play(): void| | +play(): void | | +play(): void |

+-------------+ +---------------+ +---------------+

**CODE:**

interface Bird {

abstract void fly();

abstract void makesound();

}

interface Playable {

abstract void play();

}

class Football implements Playable {

public void play() {

System.out.println("some people play football in the Ground.");

}

}

class Volleyball implements Playable {

public void play() {

System.out.println("some people play volleyball in the Stadium.");

}

}

class Basketball implements Playable {

public void play() {

System.out.println("some people play basket ball in the Rain.");

}

}

class play2 {

public static void main(String[]args) {

Playable f = new Football();

Playable v = new Volleyball();

Playable b = new Basketball();

f.play();

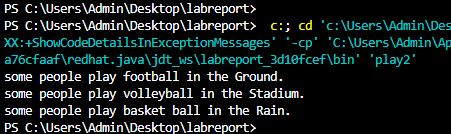
v.play();

b.play();

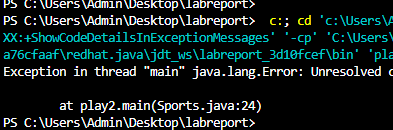
}

}

**OUTPUT:**

****

Negative case:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| A class implements Playable but does not define play()  Trying to create an object of an interface directly | Implement play() method in all implementing classes.  Use concrete classes like new Football() instead. |

IMPORTANT POINTS :-

* Playable is an **interface** that defines a contract: any class that implements it **must provide** an implementation for the play() method.
* Promotes **abstraction** and **decouples** the action (play) from specific sports
* The method in the interface: void play();
* All implementing classes must have **exactly** the same method signature