**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



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

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: K.R.N Bhanu Teja**

**Roll No: 24210**

**Verified By :**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **WEEK** | **Title** | **pgno** |
| **1** | **1** | **Installation of java** | **7** |
| **2** |  | **Write a simple java program for printing Name, Class, Roll No, of a Student** |  |
| **3** | **2** | Write a java program to find simple interest where all inputs are taken from user | 11 |
| **4** |  | Write a java program to calculate factorial of a number , read the input from user |  |
| **5** |  | Write a java program to calculate the Fibonacci  Sequence of a input taken from user |  |
| **6** |  | Write a java program to convert temperature from Celsius to Fahrenheit |  |
| **7** |  | Write a java program to convert temperature from Fahrenheit to Celsius |  |
| **8** |  | Write a java program to calculate the area of rectangle |  |
| **9** |  | Write a java program to calculate the area of triangle by using heron’s formula |  |
| **10** | **3** | Create a 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()   Create three objects named Car1, Car2 and Car3 | 24 |
| **11** |  | Create a class bankAccount with elements deposit() and Withdrawl |  |
| **12** | **4** | **Write a java program with class named book. The class The class should contain various**  **Attributes such as title, author, year of publication.**  **It should also contain a**  **Constructor with parameters which initializes title, author and year of**  **Publication.**  **Create a method which displays the details of the book .**  **Display**  **The details of two books** | **32** |
| **13** |  | **Create a java program with class named “myclass” with a static variable**  **“count” of int type, initialized to zero and a constant variable “pi” of type**  **Double initialized to 3.14 as attributes of the 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** |  |
| **14** | **5** | Create a calculator using the operations including addition, subtraction  Multiplication and division using multilevel inheritance and display the desired | 39 |
| **15** |  | 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 |  |
| **16** | **6** | 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 car | 49 |
| **17** |  | 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% |  |
| **18** |  | Create a calculator class with overloaded methods to perform addition   1. Add two integers 2. Add two doubles 3. Add three integers |  |
| **19** |  | Create a Shape class with a method calculateArea() that is overloaded for  Different shapes. Then, create a subclass circle that overrides the  calculateArea() method for a circle |  |
| **20** |  | 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 implements the sound() method to make a specific sound for each animal |  |
| **21** | **7** | Write a Java program to create an interface Playable with a method  play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override  the play() method to play the respective sports. | 66 |
| **22** |  | 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. |  |
| **23** | **8** | Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes. | 80 |
| **24** |  | Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes. |  |

# WEEK-1

1. **Process of Installing JDK (Java Development Kit)**

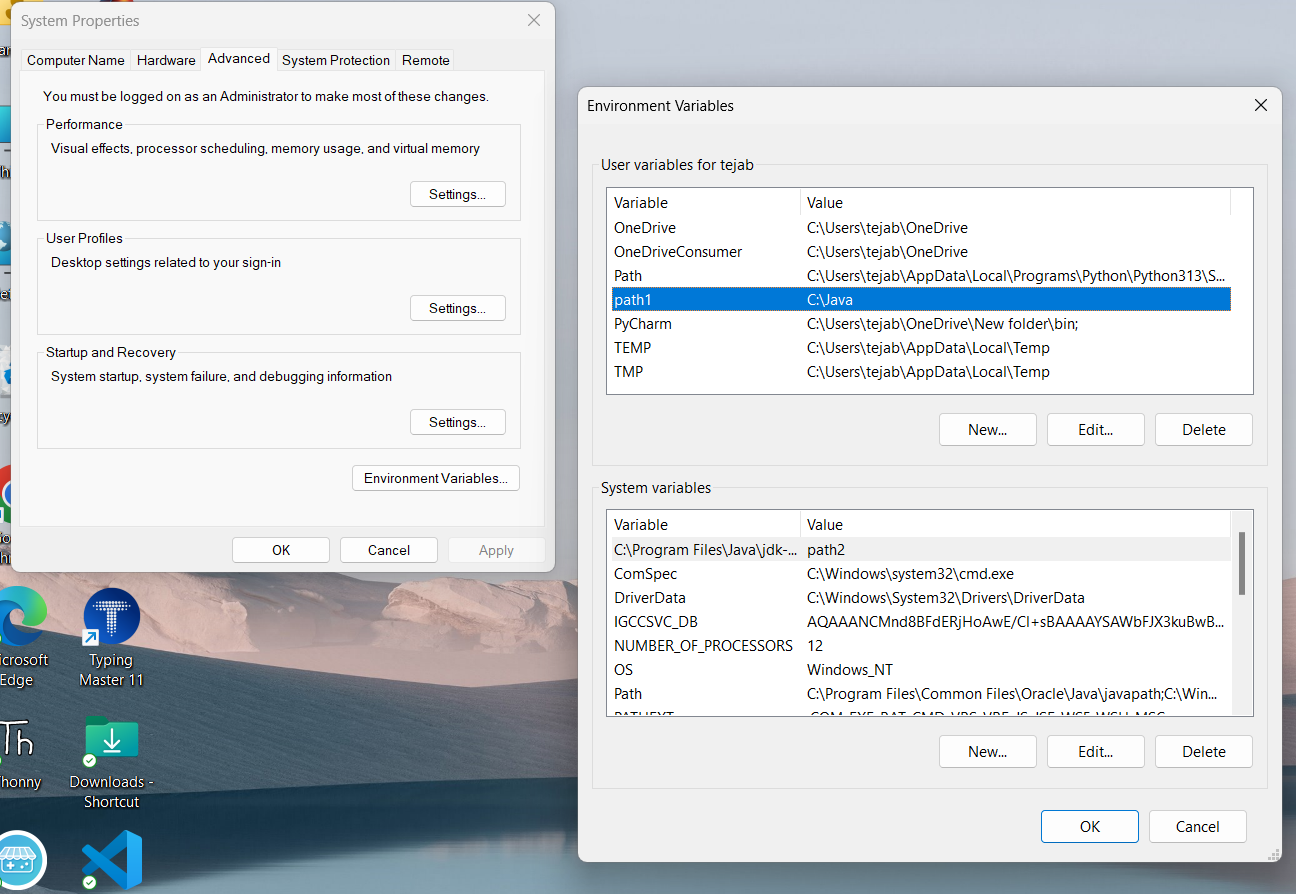
**Installing JDK (Java Development Kit):**

* 1. **Download JDK:**
* Go to the Oracle JDK download page in google and click on JDK-21 version which is Long term support (LTS) version.
* Click the download link as your operating system (Windows, macOS, or Linux).
  1. **Install JDK:**
* Once downloaded, run the installer.
* Follow the given instructions and keep clicking "Next" until it is done.
  1. **Set Environment Variables (Windows):**
* Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on **Environment Variables**.
* Click on path and new under **System Variables**:

**Variable value:** The folder address where JDK is installed (like

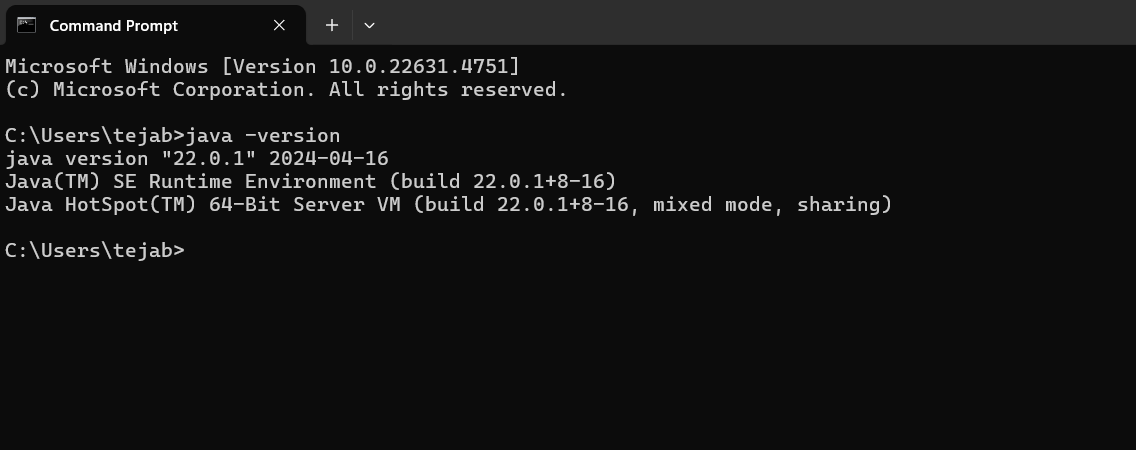
C:\Program Files\Java\jdk-21\bin)

* Find Path under **System Variables**, click **New**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)



**Checking JDK Version: -**

* 1. **Open Command Prompt:**
* Presswin+R, typecmd, and press Enter.
  1. **Check Version:**
* Type java -version and press Enter.
* Type javac --version and press Enter.



1. **Simple Java Program for printing Name, Class, Roll No, of a Student**

Aim:

Write your code in Notepad and execute it in cmd prompt

**CODE: -**

class Main

{

public static void main(String[] args)

{

System.out.println("Name: K.R.N Bhanu Teja");

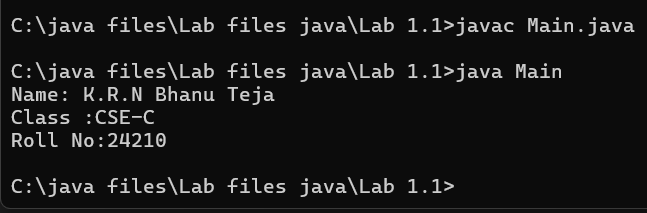
System.out.println("Class :CSE-C");

System.out.println("Roll No:24210");

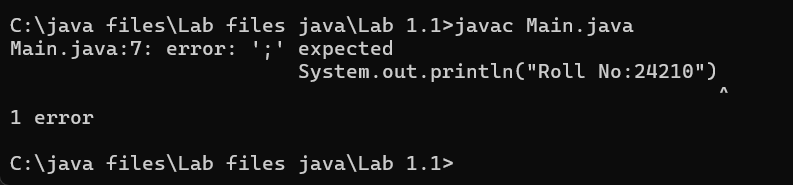
}

}

**Output: -**

****

Negative Case:



Errors

|  |  |  |
| --- | --- | --- |
| S No | Error Type | Rectification |
| 1 | Syntax error | Semicolon added |
| 2 | Runtime error | Copied correct path |
| 3 | Name error | rectified |

Week-2

1. **Aim:**

**Write a Simple Java Program for finding simple interest by taking input**

**from User**

**Code:**

**import java.util.\*;**

**class test1 {**

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

**{**

**float s;**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**System.out.println("Enter principle value:");**

**int p = sc.nextInt();**

**System.out.println("Enter time:");**

**int t = sc.nextInt();**

**System.out.println("Enter rate of interest:");**

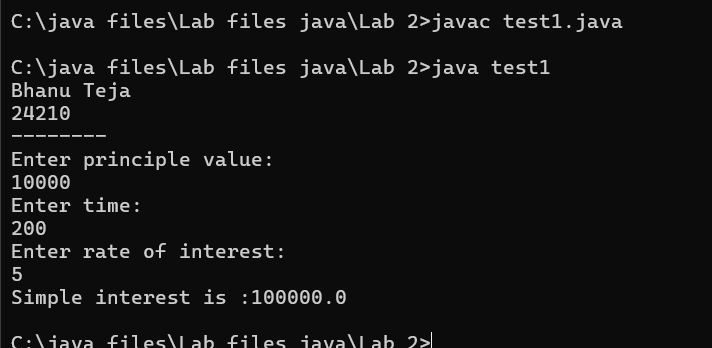
**float r = sc.nextFloat();**

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

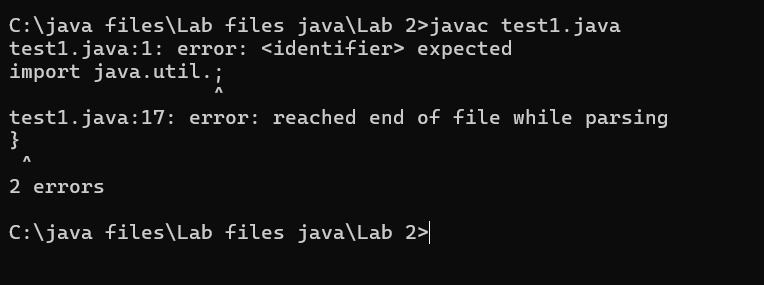
**System.out.println("Simple interest is :"+s);**

**}**

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Runtime error** | **Incorrect path** | **Copied correct path** |
| **2** | **Syntax error** | **{ missing** | **{ added** |
| **3** | **Logical error** | **Wrong formula** | **Formula rectified** |

**Important points:**

Simple interest = (p\*t\*r)/100

Scanner class is used to read the input from keyboard, files and strings in short it is used to read input from the user

**2.Aim: Write a simple java program to calculate factorial of a number and read the**

**input from user**

**code:**

**import java.util.\*;**

**class Fact {**

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

**int number;**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

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

**number = sc.nextInt();**

**int answer = factorial(number);**

**System.out.println("factorial of" + number + " is " + answer);**

**}**

**static int factorial(int n) {**

**if (n == 1) {**

**return 1;**

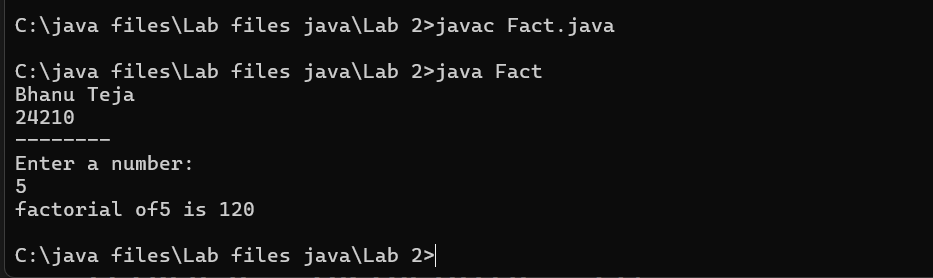
**}**

**return n \* factorial(n - 1);**

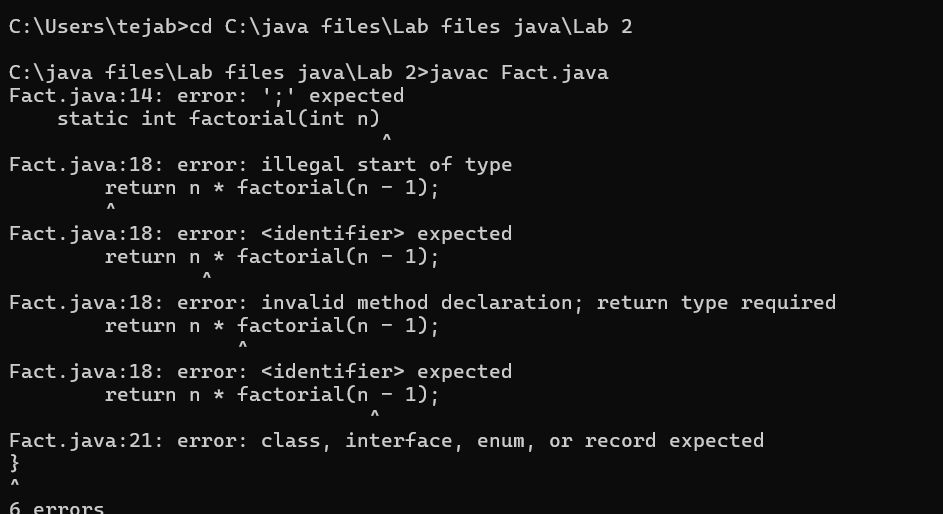
**}**

**}**

**Output:**

****

**Negative case:**

****

**Error table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Undeclared variable error** | **Missing variable** | **Variable declared** |
| **2** | **Missing import statement** | **Not importing packages** | **Packages imported** |
| **3** | **Logical error** | **Wrong formula** | **Formula rectified** |

**3.**

**Aim: Write a program to to calculate the fibonacii sequence and take the input**

**from user**

**Code:**

**import java.util.\*;**

**class fibo**

**{**

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

**{**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**int num;**

**int f3;**

**int f1 = 0;**

**int f2 = 1;**

**int i = 2;**

**System.out.print("Enter a number:");**

**num = sc.nextInt();**

**System.out.println(f1);**

**System.out.println(f2);**

**while(i<num)**

**{**

**f3 = f1+f2;**

**f1 = f2;**

**f2 = f3;**

**System.out.println(f3);**

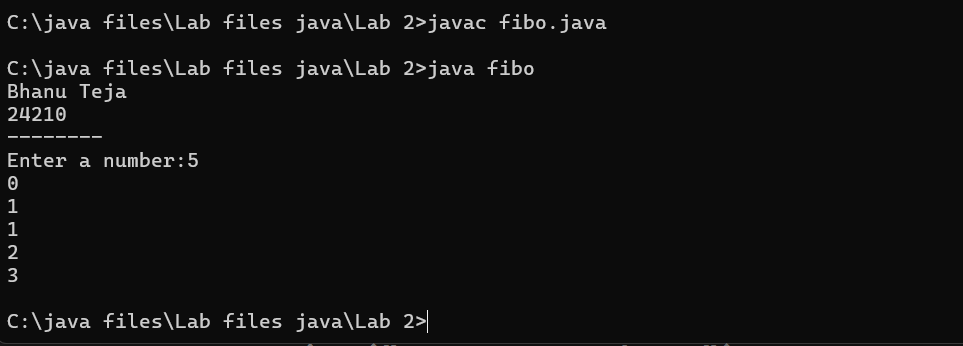
**i = i+1;**

**}**

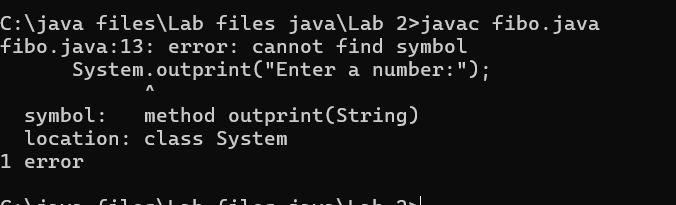
**}**

**}**

**Output:**

****

**Negative case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Formula rectified** |
| **2** | **Run-time error** | **Incorrect path** | **Added correct path** |
| **3** |  |  |  |

**Important points**

**Here the assignment operartion takes makes values are keep on updated for f1 and f2**

**4.A)**

**Aim: Write a java program to convert temperature from Fahrenheit to celsius**

**Code:**

**import java.util.\*;**

**class temp**

**{**

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

**{**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**float c;**

**System.out.println("Enter fahrenheit temperature:");**

**float f = sc.nextFloat();**

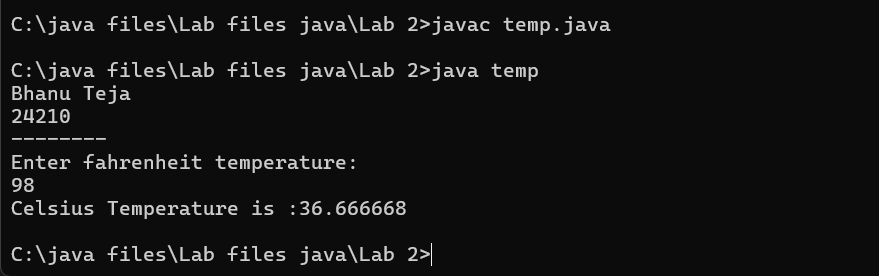
**c = (f-32)\*5/9;**

**System.out.println("Celsius Temperature is :"+c);**

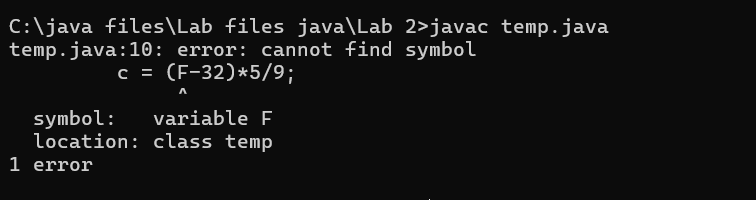
**}**

**}**

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Missing ”** | **“ is added** |
| **2** | **Missing import error** | **Util package missing** | **Util package added** |
| **3** | **Logical error** | **Incorrect formula** | **Formula rectified** |

**Important Points**

**Conversion of Fahrenheit to Celsius is c = (f-32)\*5/9**

**4.b**

**Aim: Write a java program to convert temperature from Celsius to Fahrenheit**

**Code:**

**import java.util.\*;**

**class heat**

**{**

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

**{**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**float f;**

**System.out.println("Enter celsius temperature:");**

**float c = sc.nextFloat();**

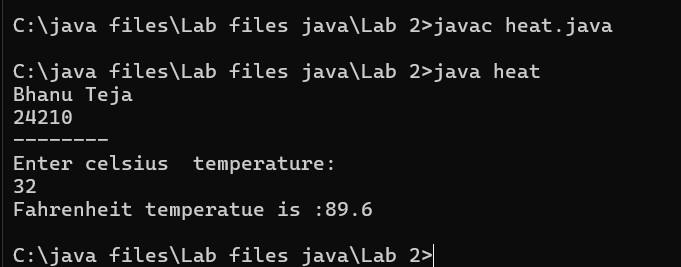
**f = (c\*9/5)+32;**

**System.out.println("Fahrenheit temperatue is :"+f);**

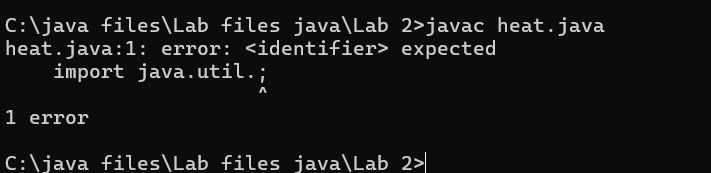
**}**

**}**

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Runtime error** | **Incorrect path selection** | **Correct path added** |
| **2** | **Logical error** | **Incorrect formula** | **Correct formula**  **rectified** |
| **3** | **Import package error** | **Incorrect importing of**  **Packages** | **Imported util.\*;**  **Package** |

**Important points**

**Conversion of celsius to Fahrenheit is (c\*9/5)+32**

**5.**

**Aim: Write a simple java program to find the area of rectangle:**

**Code:**

**import java.util.\*;**

**class area**

**{**

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

**{**

**int area;**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**System.out.println("Enter length:");**

**int l = sc.nextInt();**

**System.out.println("Enter breadth:");**

**int b = sc.nextInt();**

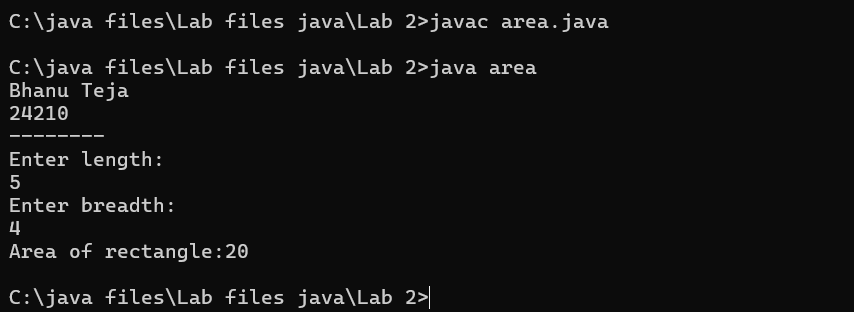
**area = l\*b;**

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

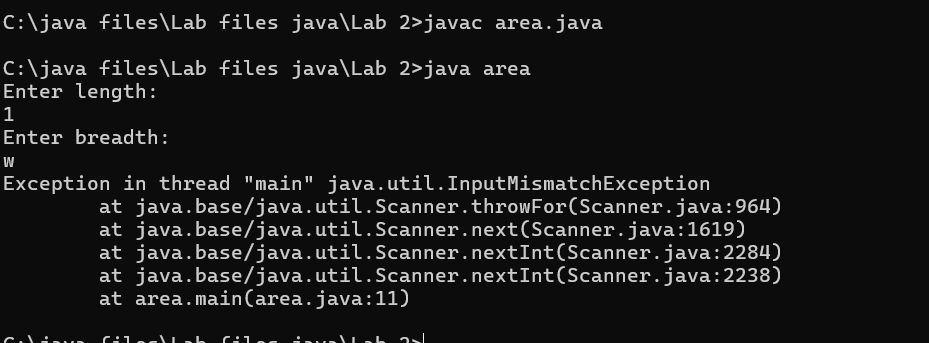
**}**

**}**

**Output:**

****

**Negative case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Semi colon missing** | **Semi colon added** |
| **2** | **Missing import error** | **Import package missing** | **Import package added** |
| **3** | **Runtime error** | **Incorrect path selection** | **Rectified correct path** |

**Important points:**

Area of rectangle = length\*breadth

**6.Aim:**

**Write a program to find the area of triangle by using heron’s formula take the input from the user**

**Code:**

**import java.util.\*;**

**import java.lang.\*;**

**class heron**

**{**

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

**{**

**double s, c,a,b,p;**

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

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**System.out.println("Enter the values of a , b and c:");**

**a = sc.nextDouble();**

**b = sc.nextDouble();**

**c = sc.nextDouble();**

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

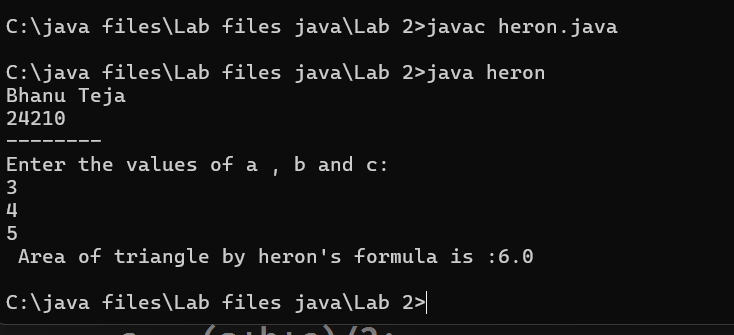
**p = Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));**

**System.out.println(" Area of triangle by heron's formula is :"+p);**

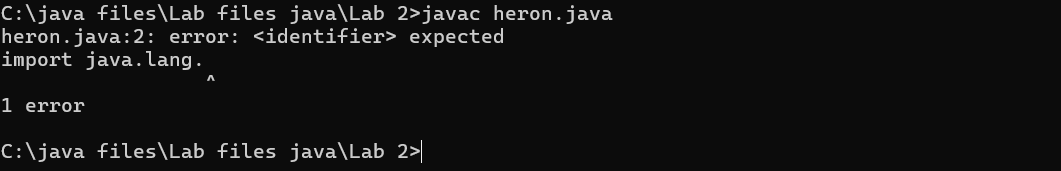
**}**

**}**

**OUTPUT:**

****

**Negative Case:**

****

**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Formula rectified** |
| **2** | **Name error** | **Undeclared variable** | **Variable declared** |
| **3** | **Import package error** | **Incorrect package** | **Package Recttified** |

**Important points**

**Import java.lang.Math is used to access the built in Math Class which provides a collection**

**of static methods**

**For performing various mathematical operations**

**WEEK 3**

1. **Aim:**

**To create java program with following instructions**

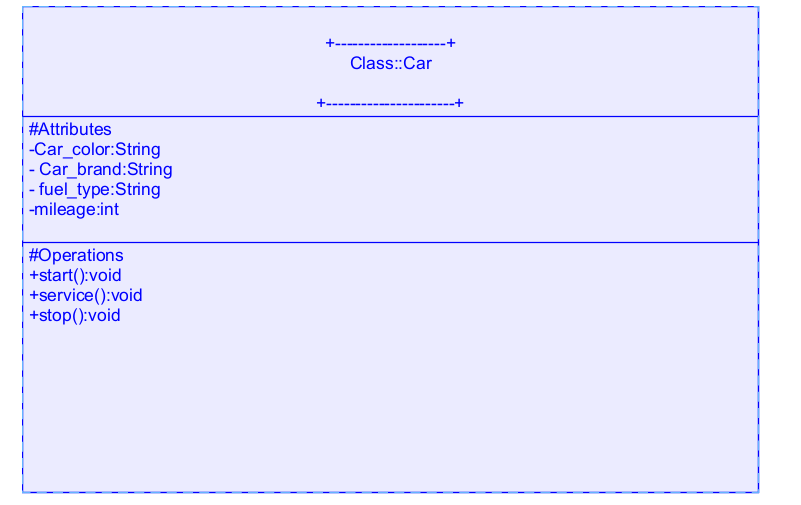
**1.Create a class with name car**

**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**

**Class Diagram:**

****

**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[])

{ System.out.println("\nBHANU TEJA\n\n");

car car1 = new car();

car1.Car\_color = "Blue";

car1.Car\_brand = "Audi";

car1.fuel\_type = "Deisel";

car1.mileage = 100;

car1.start();

car car2 = new car();

car2.Car\_color = "Red";

car2.Car\_brand = "Tesla";

car2.fuel\_type = "EV";

car2.mileage = 200;

car2.stop();

car car3 = new car();

car3.Car\_color = "Yellow";

car3.Car\_brand = "BMW";

car3.fuel\_type = "Petrol";

car3.mileage = 300;

car3.service();

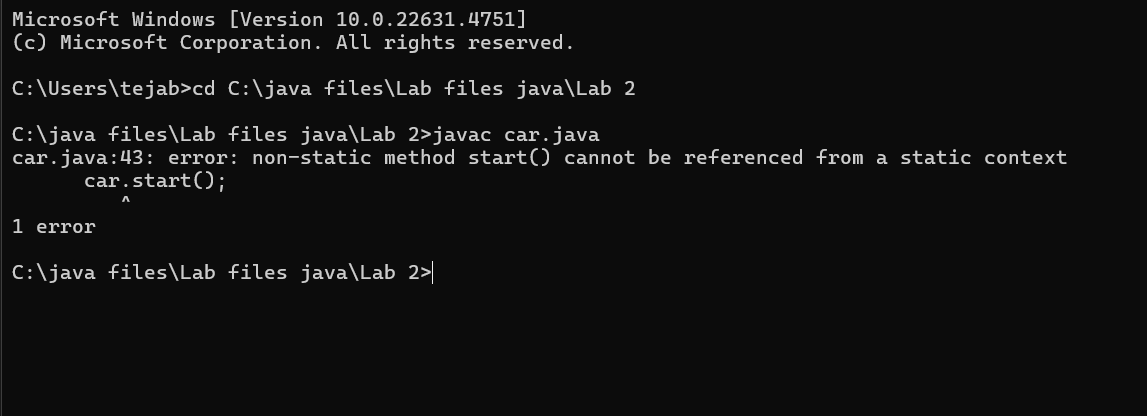
}

}

**Output:**

****

**Negative Case**

****

**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Error Type** | **Cause of error** | **Rectification** |
| **1** | **Syntax Error** | **Missing ‘{‘** | **‘{‘ added** |
| **2** | **Compile time Error** | **Mispelled Variable call** | **Rectified with**  **Correct variable name** |

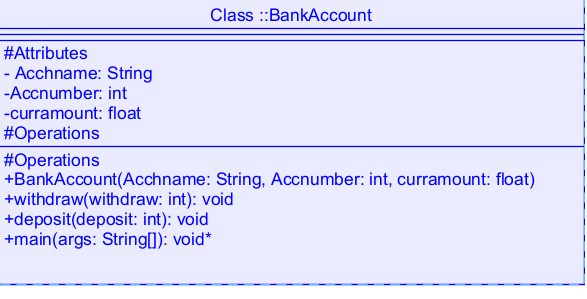
**Important points**

Methods: The set of instructions that can be called for execution using a method name.

**2.**

**Aim: To create a class bankAccount with methods deposit() and withdrawl**

**Class Diagram**

****

**Code:**

class BankAccount

{

public String Acchname;

public int Accnumber;

public float curramount;

BankAccount(String Acchname, int Accnum,float curramount)

{

this.Acchname = Acchname;

this.Accnumber = Accnumber;

this.curramount = curramount;

System.out.println("Enter Account holder name:"+Acchname);

System.out.println("Enter Account number:"+Accnum);

System.out.println("Enter current amount:"+curramount);

}

public void withdraw(int withdraw)

{

if(withdraw>curramount)

{

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

}

else

{

curramount = curramount-withdraw;

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

System.out.println("Current amount is:"+curramount);

}

}

public void deposit(int deposit)

{

System.out.println("Deposited amount is :");

curramount = curramount+deposit;

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

System.out.println("Total current amount is:"+curramount);

}

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

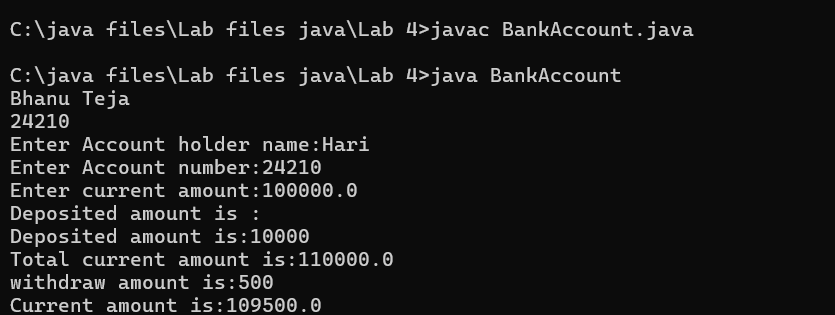
BankAccount b = new BankAccount("Hari",24210,100000);

b.deposit(10000);

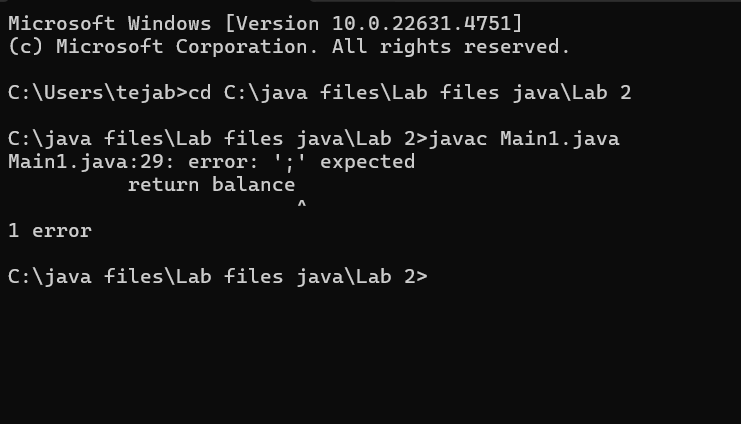
b.withdraw(500);

}}

**Output:**

****

**Negative Case**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **s.no** | **Error name** | **Cause of error** | **Rectification** |
| **1** | **Name Error** | **Undefined name** | **Correct variable**  **Name replaced** |
| **2** | **Syntax Error** | **Missing Parenthesis** | **Parenthesis Added** |
| **3** | **Logical Error** | **Incorrect Condition** | **Condition Rectified** |

**Important points:**

**Constructor:** The Constructor creates and initializes objects of a class. They are called

when an object is created to a class.

**This Keyword:** The This keyword refers to the current instance of a class.It is used to

Access class variables and met

**Week-4**

**1**

**Aim: Write a java program with class named book. The class should contain various**

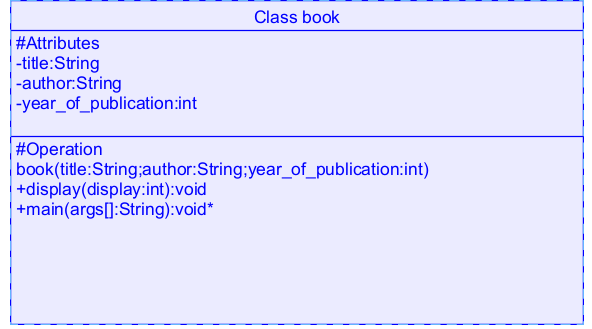
**Attributes such as title, author, year of publication. It should also contain a**

**Constructor with parameters which initializes title, author and year of**

**Publication. Create a method which displays the details of the book .Display**

**The details of two books**

**Class Diagram**

****

**Code**

**class book**

**{**

**public String title;**

**public String author;**

**public int year\_of\_publication;**

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

**{**

**this.title = title;**

**this.author = author;**

**this.year\_of\_publication = year\_of\_publication;**

**}**

**public void display()**

**{**

**System.out.println("Title of book is:"+title);**

**System.out.println("Author of book is:"+author);**

**System.out.println("Year of publication is:"+year\_of\_publication);**

**}**

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

**{**

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

**book b1 = new book("Python","Bhanu",2023);**

**b1.display();**

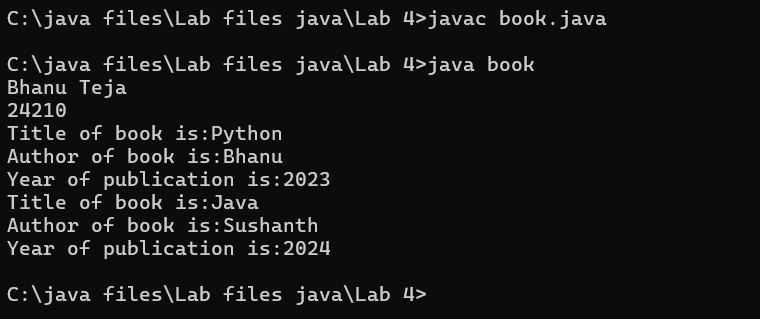
**book b2 = new book("Java","Sushanth",2024);**

**b2.display();**

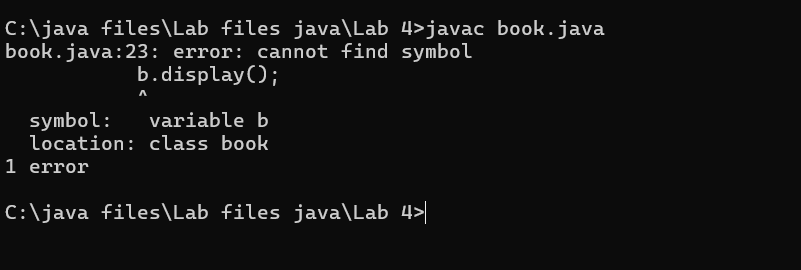
**}**

**}**

**Output:**

****

**Negative Case**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason** | **Rectification** |
| **1** | **Syntax error** | **Missing {** | **Added {** |
| **2** | **Static method called non statically** | **Fault in calling the method** | **Rectified method**  **Calling** |
| **3** | **Run-time error** | **Incorrect Selection of**  **Path** | **Correct path**  **Selected** |

**Important points:**

**Constructor:**

Constructor are special methods used to initialize objects when they are created

**2.Create a java program with class named “myclass” with a static variable**

**“count” of int type, initialized to zero and a constant variable “pi” of type**

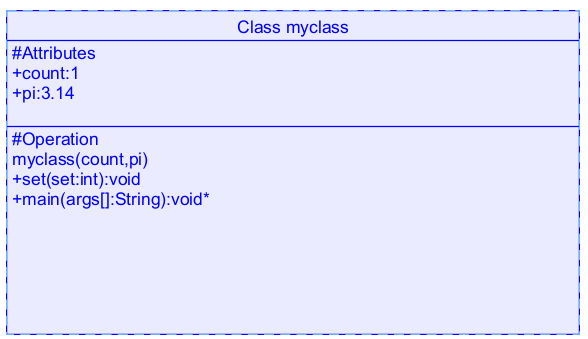
**Double initialized to 3.14 as attributes of the class. Now define a constructor**

**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**

**Class Diagram:**



**Code:**

**class myclass**

**{**

**static int count=0;**

**static double pi=3.14;**

**myclass()**

**{**

**count = count+1;**

**}**

**public void set()**

**{**

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

**System.out.println("Pi value is:"+pi);**

**}**

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

**{**

**System.out.println("Bhanu Teja");**

**System.out.println("24210");**

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

**myclass m = new myclass();**

**m.set();**

**myclass m1 = new myclass();**

**m1.set();**

**myclass m2 = new myclass();**

**m2.set();**

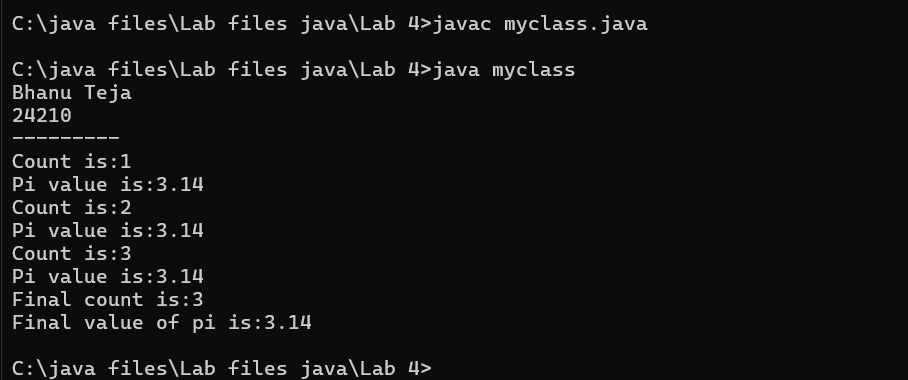
**System.out.println("Final count is:"+count);**

**System.out.println("Final value of pi is:"+pi);**

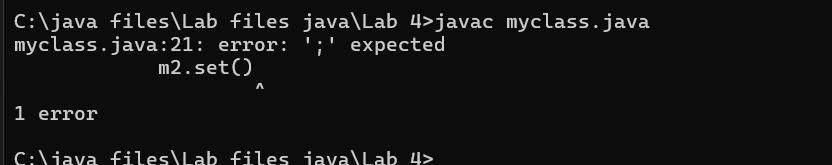
**}**

**}**

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error Type | Cause | Rectification |
| 1 | Name error | Incorrect variable called | Rectified with correct variable |
| 2 | Syntax error | Missing semi-colon | Semi-colon added |
| 3 | Run time error | Incorrect path | Selected correct path |

Important Points

The count value increases and the loop repeats until the condition go wrong

**WEEK-5**

**1.AIM:** Create a calculator using the operations including addition, subtraction

Multiplication and division using multilevel inheritance and display the desired

Output

**Class Diagram:**



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[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

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));

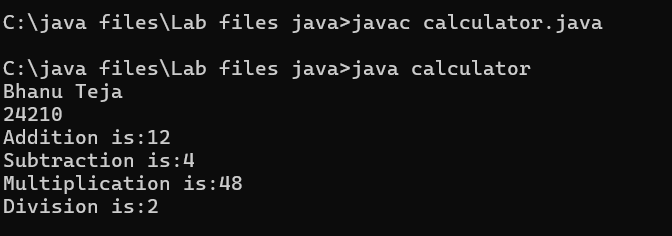
System.out.println("Division is:"+obj.div(8,4));

}

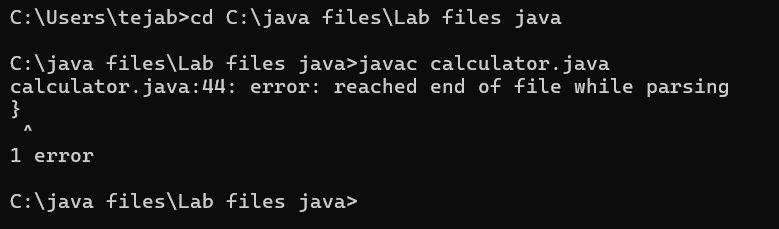
}

Output

Positive case



**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) {

System.out.println("Bhanu Teja");

System.out.println("24210");

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 is:"+brand);

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

System.out.println("No of doors are:"+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 is:"+brand);

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

System.out.println("No of gears are:"+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 is:"+brand);

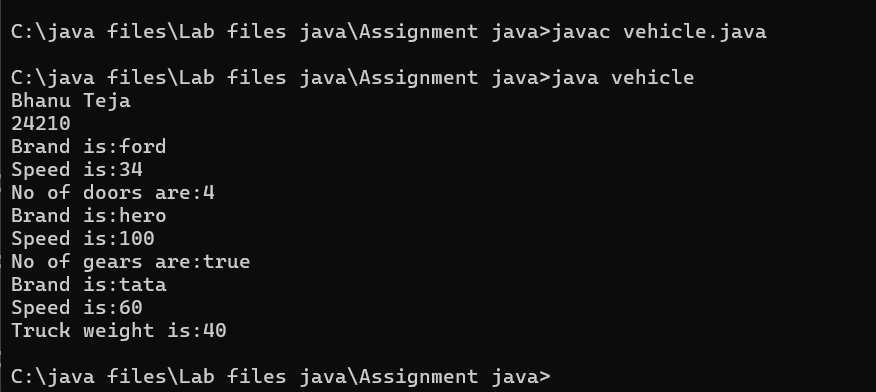
System.out.println("Speed is:"+speed);

System.out.println("Truck weight is:"+weight);

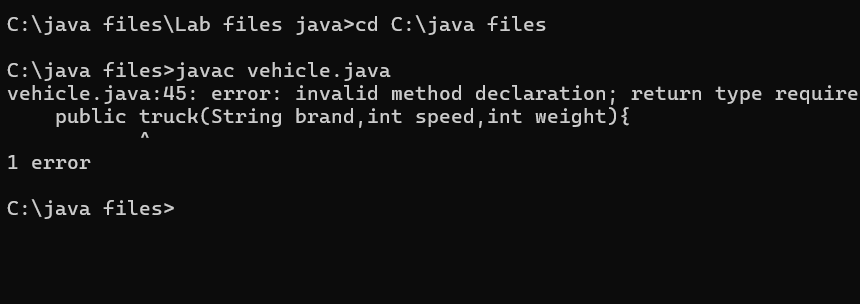
}

}

Output:



Negative Case:



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Class format error | Incorrect format of classes | Correct format rectified |
| 3 | Run time error | Exception at runtime | Exception rectified |

**Important Points**

Hierarchical Inheritence:

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

Single parent class

**WEEK-6**

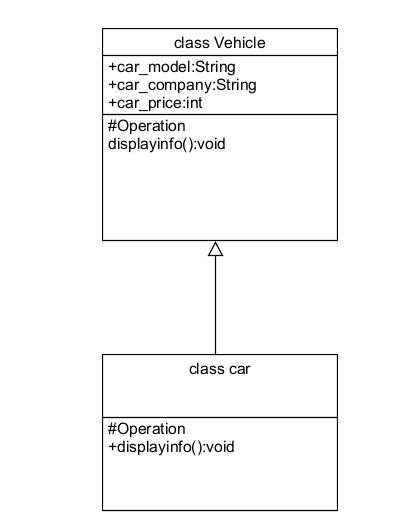
1. **Aim:**

Write a java program to create a vehicle class with a method displayInfo().

Override this method in the car subclass to provide specific information

About car

**Class Diagram**



Code:

class vehicle

{

public String car\_model;

public String car\_company;

public int car\_price;

void displayinfo()

{

System.out.println("Bhanu Teja 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[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

car obj1 = new car();

obj1.car\_model="Swift";

obj1.car\_company="Suzuki";

obj1.car\_price=1000000;

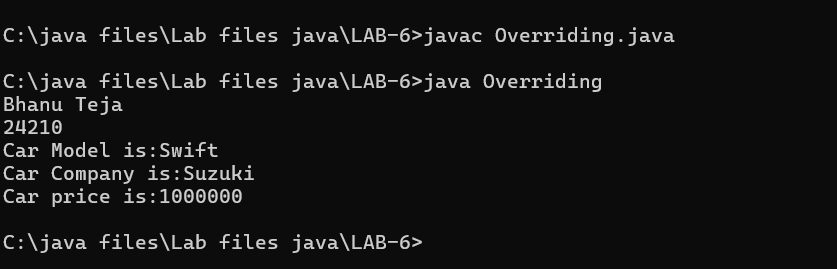
obj1.displayinfo();

}

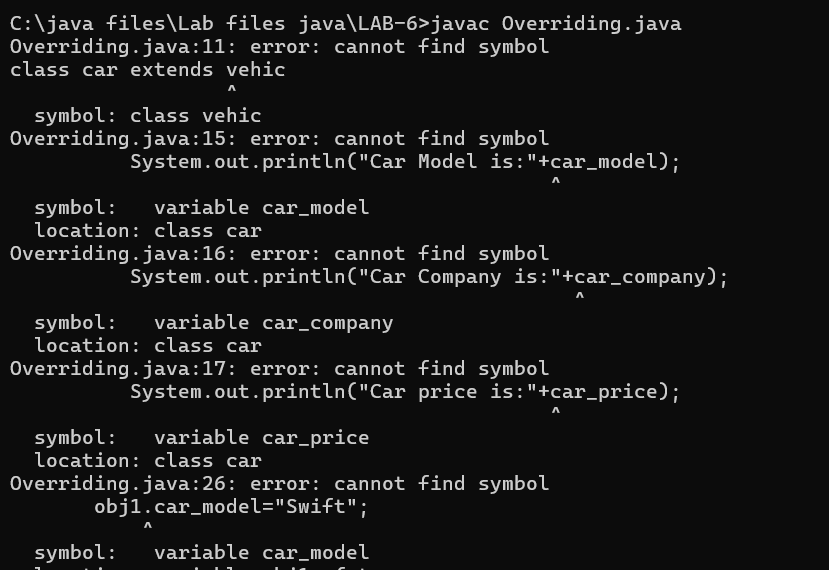
}

**Output:**

**Positive case:**



**Negative Case:**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error type | cause | Rectification |
| 1 | Name error | Incorrect variable declaration | Rectified with correct variable |
| 2 | Syntax error | Incorrect use of print statement | Rectified print statement |
| 3 | Run-time error | Incorrect file saved | Rectified with correct file |

**Important Points**

Method Overriding:

Method Overriding allows a subclass to provide a specific implementation of a

Method that is already defined in its superclass , enabling runtime

polymorphism

**2.**

**Aim:** A college is developing an automated admission system that verifies

Student eligibility for UG and PG programs. Each program has different

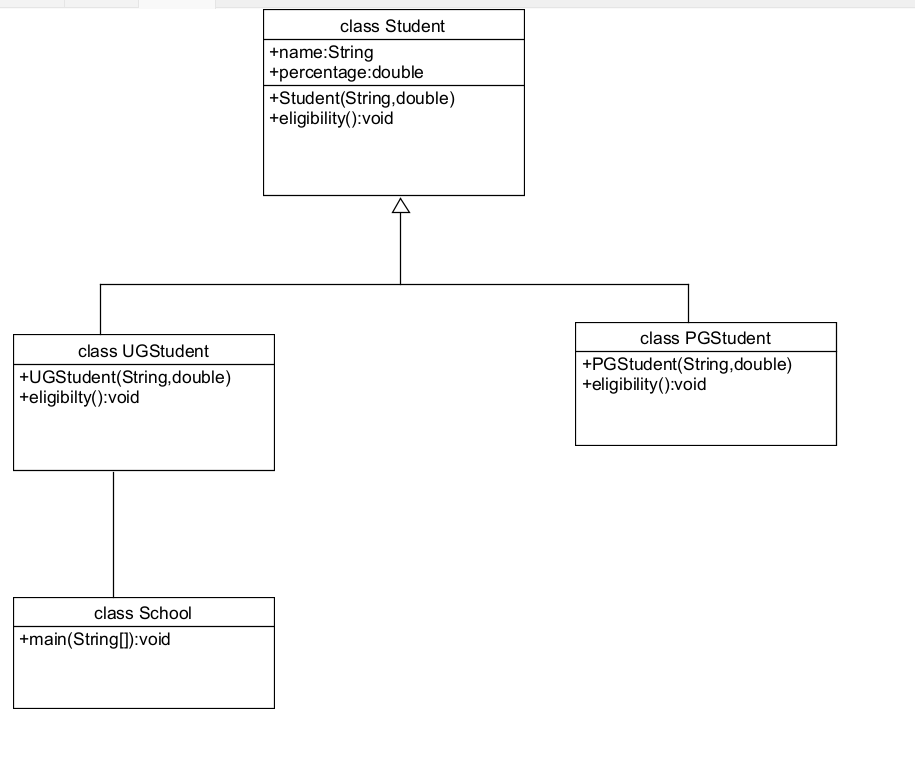
Eligibility criteria based on the student’s percentage in their previous

Qualification

UG admissions require a minimum of 60%

PG admissions require a minimum of 70%

**Class Diagram**



**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[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

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

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

obj1.eligibility();

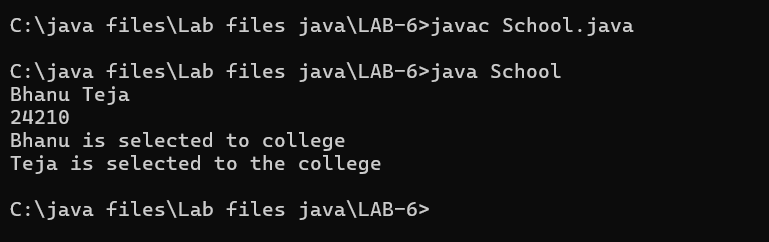
obj2.eligibility();

}

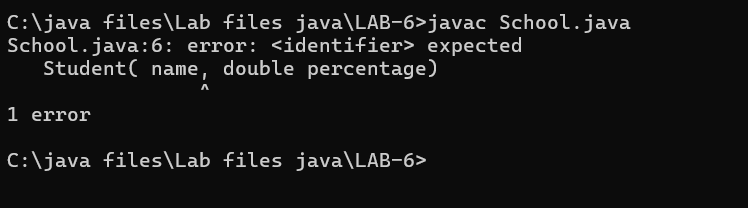
}

**Output:**

**Positive case**



**Negative Case:**



**Error table**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | cause | Rectification |
| 1 | Syntax Error | Missing “;” | Semicolon added |
| 2 | Name Error | Incorrect method call | Correct method was rewritten |
| 3 | Run-time Error | Incorrect path | Rectified correct path |

**Important points:**

Super() keyword:

We use super class method to call a superclass method from within a

subclass

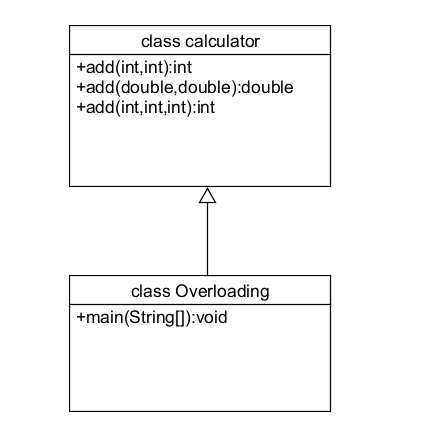
**3**

**Aim:**

Create a calculator class with overloaded methods to perform addition

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

**Class Diagram**



**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[]){

System.out.println("Bhanu Teja");

System.out.println("24210");

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));

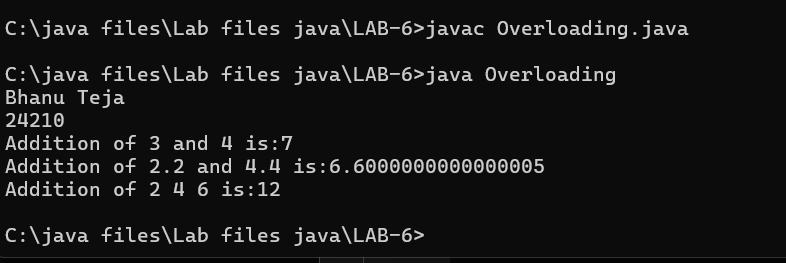
System.out.println("Addition of 2 4 6 is:"+c.add(2,4,6));

}

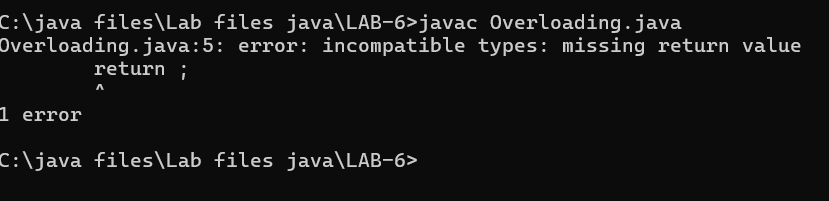
}

**Output:**

**Positive case:**



**Negative case:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S NO | Error type | Cause | Rectification |
| 1 | Return type error | Missing return type | Added return  value |
| 2 | Nullpointer exception | Incorrect object creation | Replaced with correct variable |
| 3 | Logical error | Incorrect formula | Correct formula  rectified |

**Important Points:**

**Method Overloading:**

Method overloading allows defining multiple methods within the same class

that share the same name but have different parameter lists

The datatypes in the parameters of method may be of same type or different

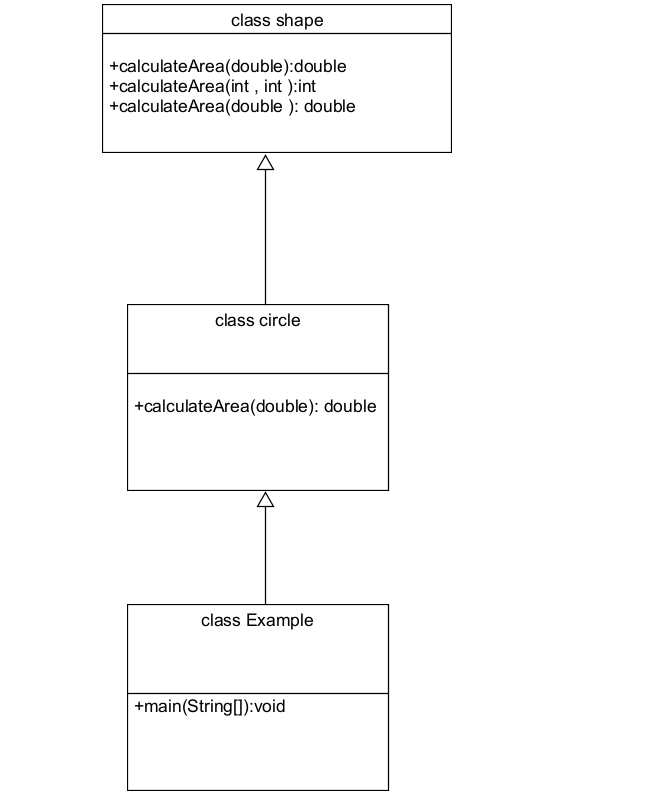
4

**Aim**: Create a Shape class with a method calculateArea() that is overloaded for

Different shapes. Then, create a subclass circle that overrides the

calculateArea() method for a circle

class Diagram



**Code:**

class shape

{

public double calculateArea(double side)

{

return side\*side;

}

public int calculateArea(int length, int width)

{

return length\*width;

}

}

class circle extends shape

{

public double calculateArea(double radius)

{

return 3.14\*radius\*radius;

}

}

class Example

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

circle obj1 = new circle();

shape obj2 = new shape();

System.out.println("The area of side 6 is:"+obj2.calculateArea(2.0));

System.out.println("The area of circle with radius 3 :"+

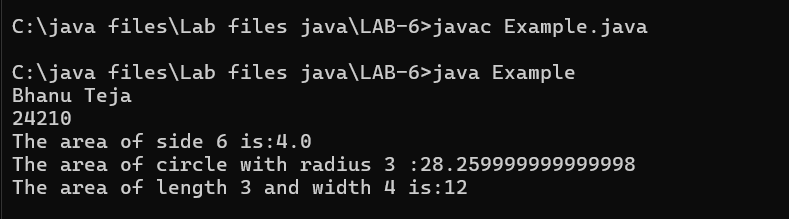
obj1.calculateArea(3.0));

System.out.println("The area of length 3 and width 4 is:"+obj2.calculateArea(3,4));

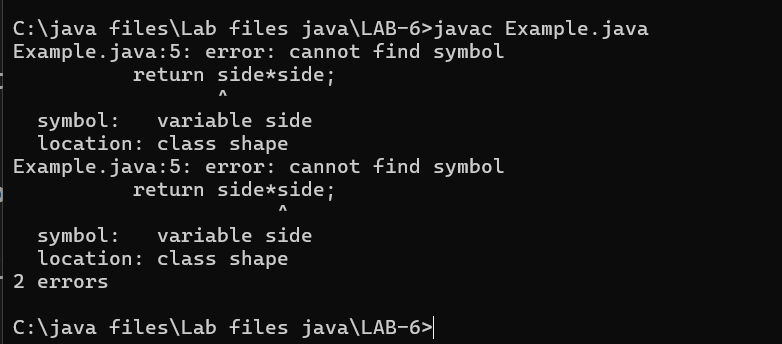
}

}

Output:



**Negative case:**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| S no | Error type | cause | Rectification |
| 1 | Syntax error | Missing parenthesis | Adding parenthesis |
| 2 | Runtime error | Incorrect path | Corrected path |
| 3 | Name error | Incorrect use of variable | Variable name rectified |

**Important points:**

The return keyword is used to return the value during the method call with the same data type Expected

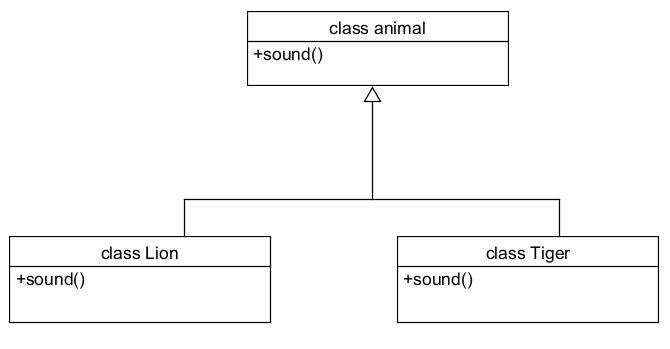
**Week-7**

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 implements the sound() method to make a specific sound for each animal

Class Diagram:



**Code:**  
abstract class animal

{

abstract void sound();

}

class Lion extends animal

{

void sound()

{

System.out.println("Lions Roars in the jungle");

}

}

class Tiger extends animal

{

void sound()

{

System.out.println("Tiger can make sounds");

}

}

class Abstraction

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

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

Lion obj1 = new Lion();

obj1.sound();

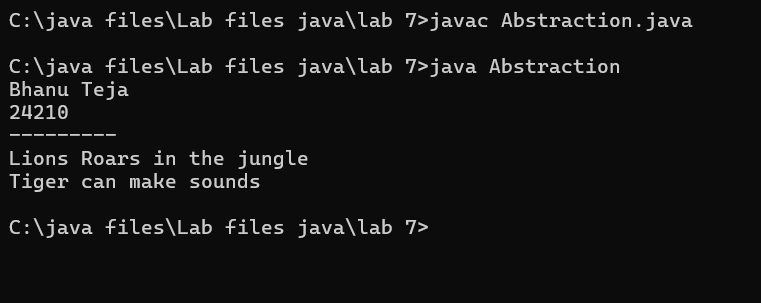
Tiger obj2 = new Tiger();

obj2.sound();

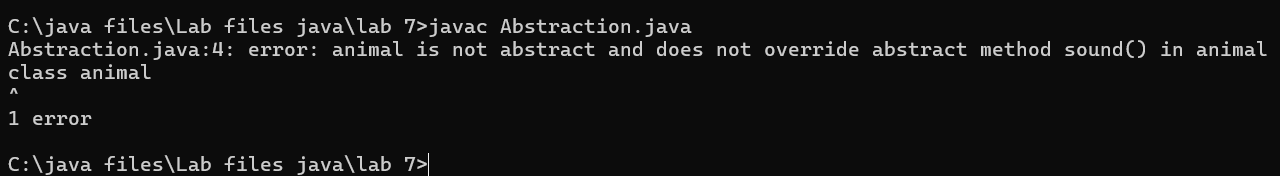
}

}

**Output:**

**Positive case:**  


**Negative case:**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Cause** | **Rectification** |
| **1** | Static reference to non-static method | Calling non-static method from static context | Create object correctly |
| **2** | Instantiation of abstract class(OOP error) | Trying to create an object of an abstract class | Use subclass or interface implementation |
| **3** | Syntax error | Missing parenthesis | Adding parenthesis |

**Important points:**

The abstract class animal defines the sound() method that must be

implemented by subclasses.

Lion and Tiger extend animal and provide their own versions of the sound() method.

The main() method demonstrates runtime polymorphism by calling sound() on each object

**2.Aim:** Write a java program to create an abstract class shape3D with

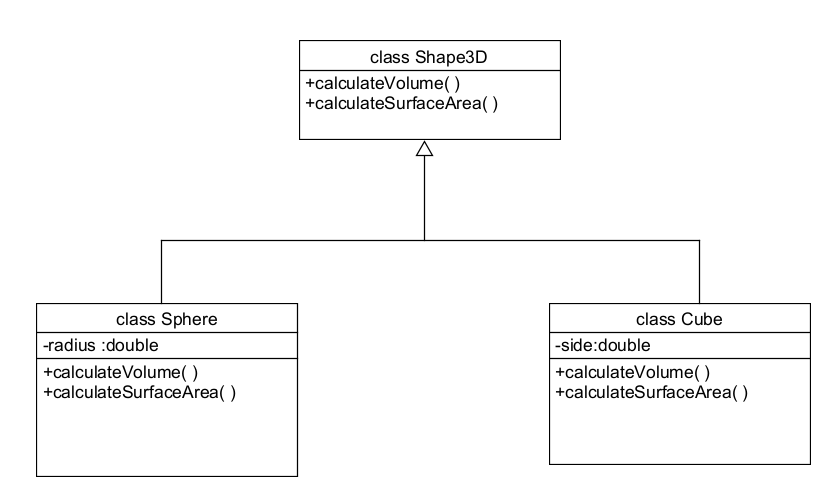
abstract methods calculateVolume() and calculateSurfaceArea().Create subclasses

sphere and cube that extend the shape3D class and implement

the respective methods to calculate the volume and surface area of each

shape

**Class Diagram**



**Program**

abstract class Shape3D

{

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D

{

private int radius;

Sphere(int radius)

{

this.radius = radius;

}

public double calculateVolume()

{

return 1.33\*2.14\*radius\*radius\*radius;

}

public double calculateSurfaceArea()

{

return 4\*3.14\*radius\*radius;

}

}

class Cube extends Shape3D

{

private int a;

Cube(int a)

{

this.a = a;

}

public double calculateVolume()

{

return a\*a\*a;

}

public double calculateSurfaceArea()

{

return 6\*a\*a;

}

}

class Mainn

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("CSE24210");

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

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

Sphere obj1 = new Sphere(4);

Cube obj2 = new Cube(3);

System.out.println(obj1.calculateVolume());

System.out.println(obj1.calculateSurfaceArea());

System.out.println(obj2.calculateVolume());

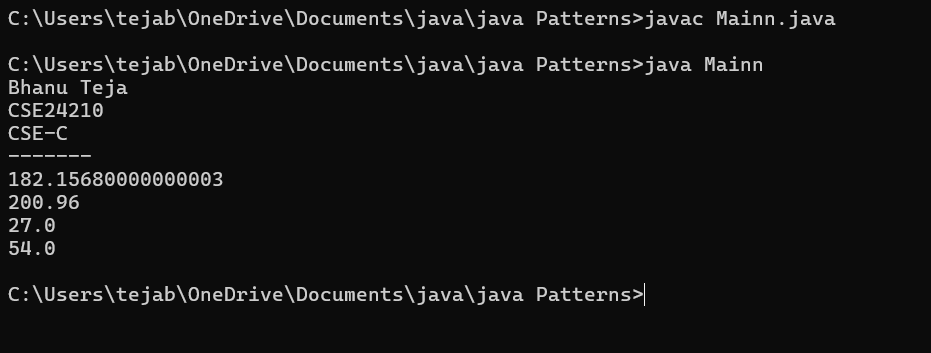
System.out.println(obj2.calculateSurfaceArea());

}

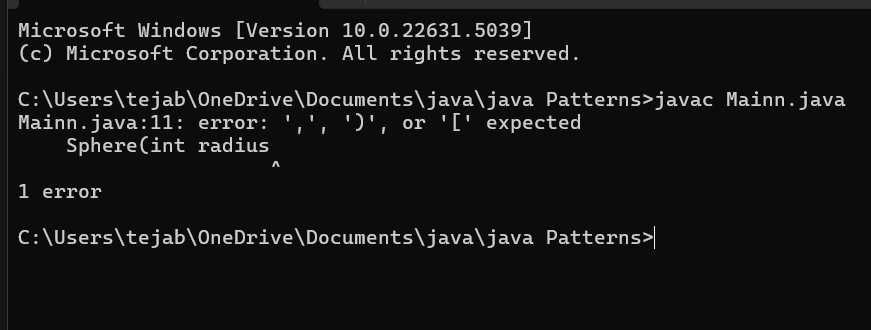
}

**Output**

**Positive case:**



**Negative case:**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Error Type** | **Cause** | **Rectification** |
| **1** | Return type missing | Method doesn’t specify return type | Adding the correct return type |
| **2** | Instantiation of abstract class | Trying to create an object of abstract class without any subclass or interface | Use subclass or interface implementation |
| **3** | Static reference to Non-static Method | Calling non-static method from static context | Create object or make method static |

**Important points**

Shape3D is an abstract class with abstract methods to calculate volume and surface area.

Sphere and Cube extend Shape3D and provide specific implementations for volume and surface area.

The main method creates objects and prints the calculated values — demonstrating polymorphism and abstraction.

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

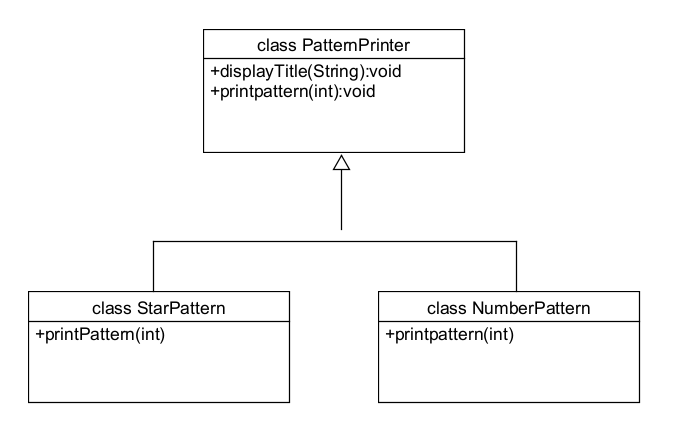
1.star pattern- prints a right angled triangle of stars(\*)

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.

Class Diagram



Code

abstract class PatternPrinter

{

abstract void printpattern(int n);

public void display(String title)

{

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

}

}

class starpattern extends PatternPrinter

{

void printpattern(int n)

{

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

{

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

{

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

}

System.out.println();

}

}

}

class numberpattern extends PatternPrinter

{

void printpattern(int n)

{

int num=1;

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

{

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

{

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

num++;

}

System.out.println();

}

}

}

class Main2

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

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

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

int rows=5;

PatternPrinter star = new starpattern();

star.display("Number pattern");

star.printpattern(rows);

PatternPrinter number = new numberpattern();

number.display("Number pattern");

number.printpattern(rows);

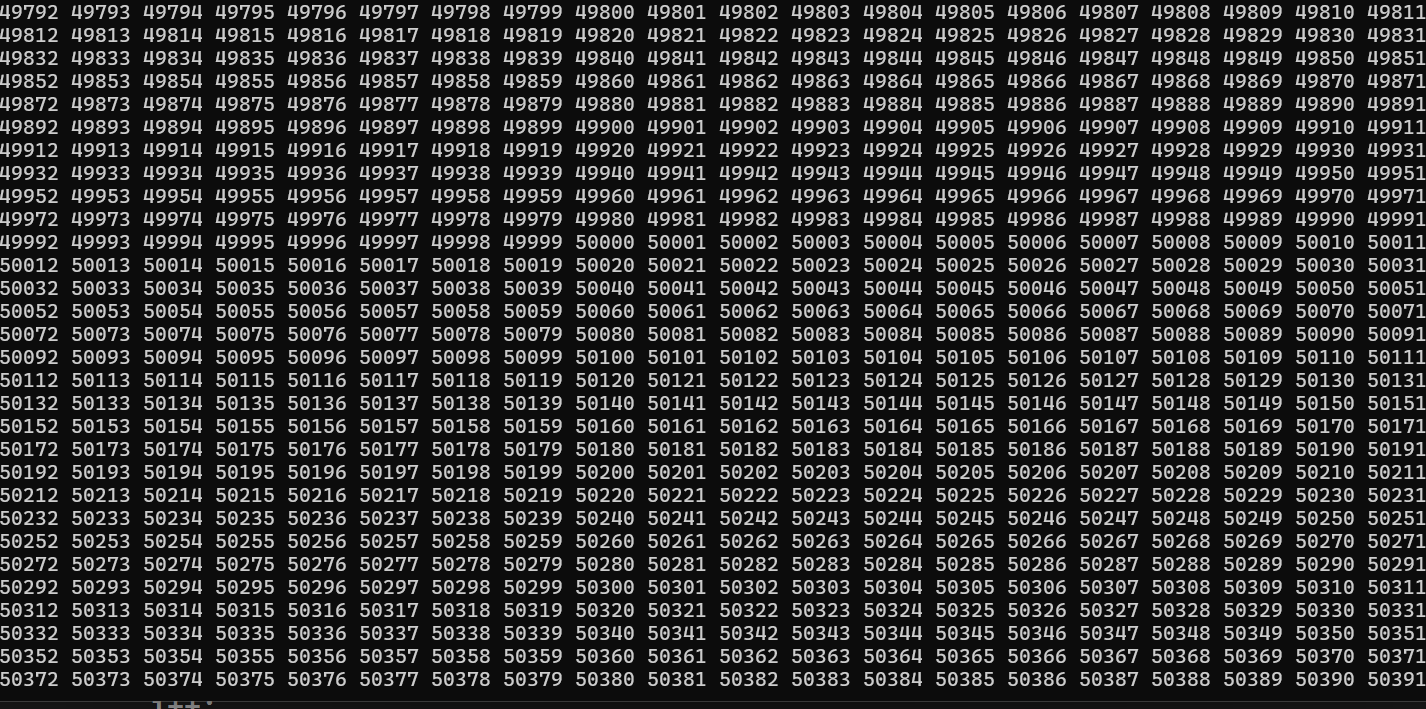
}

}

Output

Positive case:



**Negative Case:**  


**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error Type | Cause | Rectification |
| 1 | Logical error | Incorrect use of operator | Using correct operator |
| 2 | Run time error | Incorrect path | Selected correct path |
| 3 | Syntax error | Missing semicolon | Semicolon added |

**Important points:**

Abstraction is used to define a common method printpattern() in the abstract class PatternPrinter, allowing different patterns to be printed

through subclasses.The starpattern and numberpattern classes implement their own version

of printpattern(), demonstrating polymorphism.

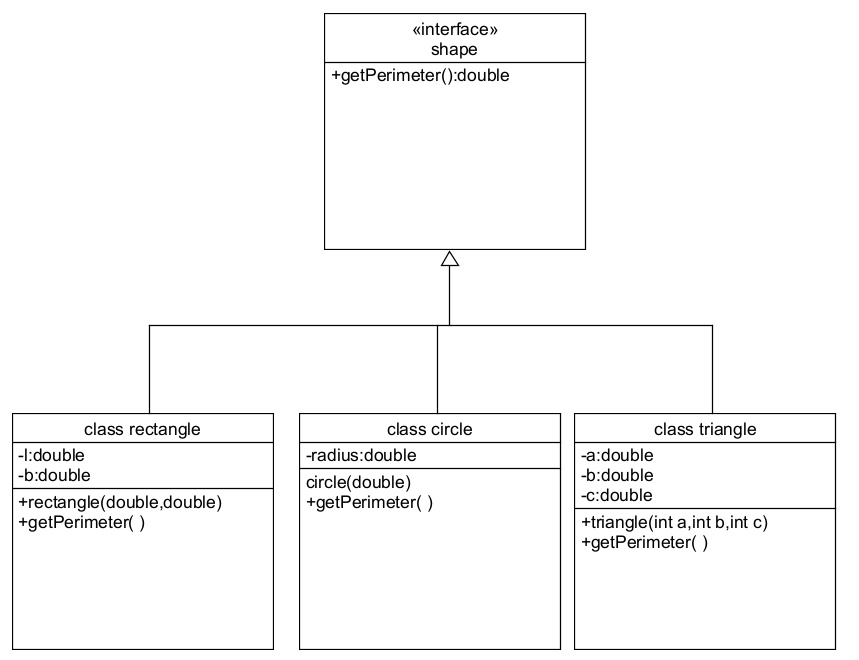
The display() method in the abstract class allows for a consistent header

before printing any pattern, making the output structured and easy to manage.

**Week-8**

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

**Class diagram:**



**Code:**

interface shape

{

public double getperimeter();

}

class rectangle implements shape

{

private double l;

private double b;

rectangle(double l, double b)

{

this.l=l;

this.b = b;

}

public double getperimeter()

{

return 2\*(l+b);

}

}

class circle implements shape

{

private double r;

circle(double r)

{

this.r = r;

}

public double getperimeter()

{

return 2\*3.14\*r;

}

}

class triangle implements shape

{

private double a;

private double b;

private double c;

triangle(double a, double b, double c)

{

this.a = a;

this.b = b;

this.c = c;

}

public double getperimeter()

{

return a+b+c;

}

}

class Main

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("24210");

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

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

rectangle r = new rectangle(4,2);

System.out.println(r.getperimeter());

circle c = new circle(7);

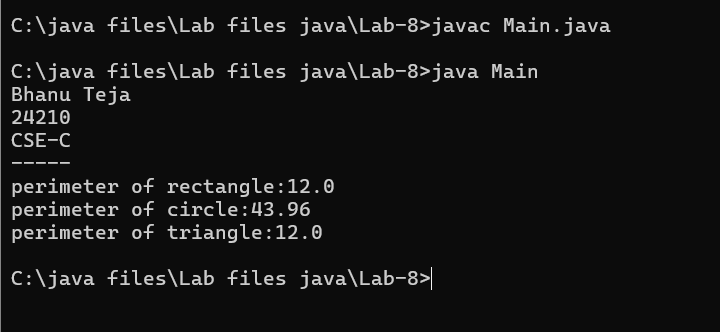
System.out.println(c.getperimeter());

triangle t = new triangle(3,4,5);

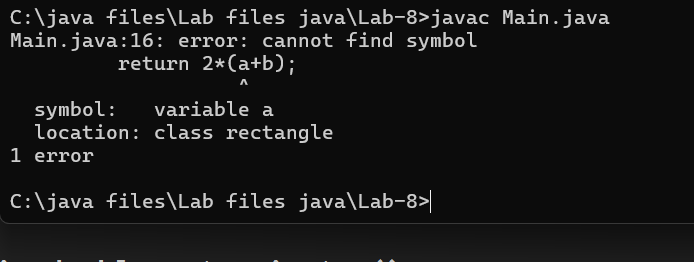
System.out.println(t.getperimeter());

}}

Output:  
**Positive case**



**Negative Case:**



**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| 1 | Compilation error | Return type missing | Add correct return type |
| 2 | Access modifier issues(OOP error) | Trying to access private members | Change the modifier or add getter/setter |
| 3 | Syntax Error | Mismatched brackets | Close properly all brackets |

**Important points:**

**Flexibility and Extensibility:** This design makes it easy to add more shapes

(like Square, Ellipse) in the future by simply implementing the Shape interface without modifying existing code.

**Interface Implementation:** The Shape interface is implemented by three

classes: Rectangle, Triangle, and Circle, each defining the getPerimeter()

method for their respective shapes.

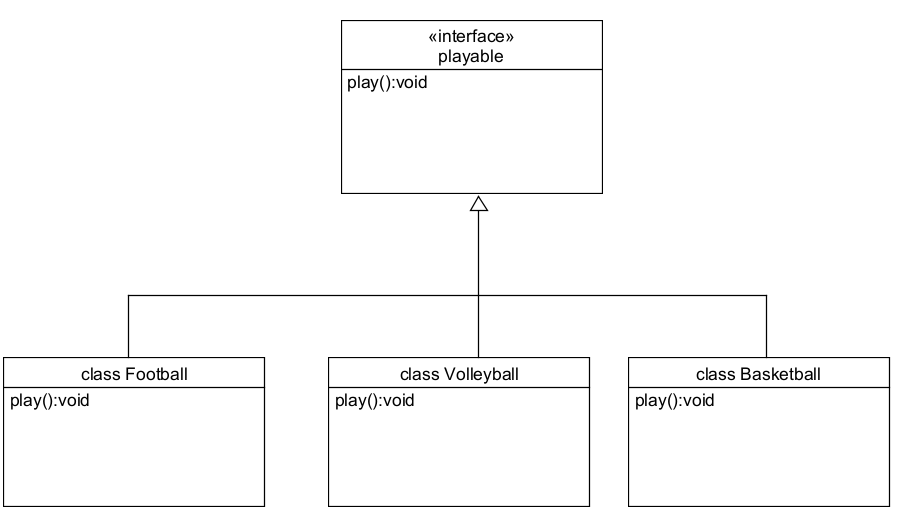
**2.Aim:** Write a Java program to create an interface Playable with a method

play() that takes no arguments and returns void. Create three classes Football, Volleyball,

and Basketball that implement the Playable interface and override

the play() method to play the respective sports.

Class diagram:



Code:

interface playable

{

public void play();

}

class Football implements playable

{

public void play()

{

System.out.println("Foot ball");

}

}

class Volleyball implements playable

{

public void play()

{

System.out.println("Volley ball");

}

}

class Basketball implements playable

{

public void play()

{

System.out.println("Basket ball");

}

}

class Main1

{

public static void main(String args[])

{

System.out.println("Bhanu Teja");

System.out.println("CSE24210");

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

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

playable obj1 = new Football();

playable obj2 = new Volleyball();

playable obj3 = new Basketball();

obj1.play();

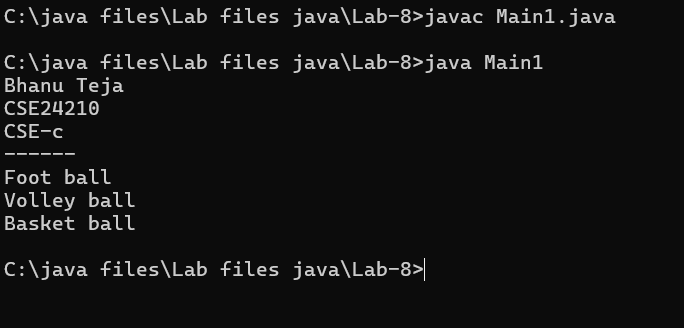
obj2.play();

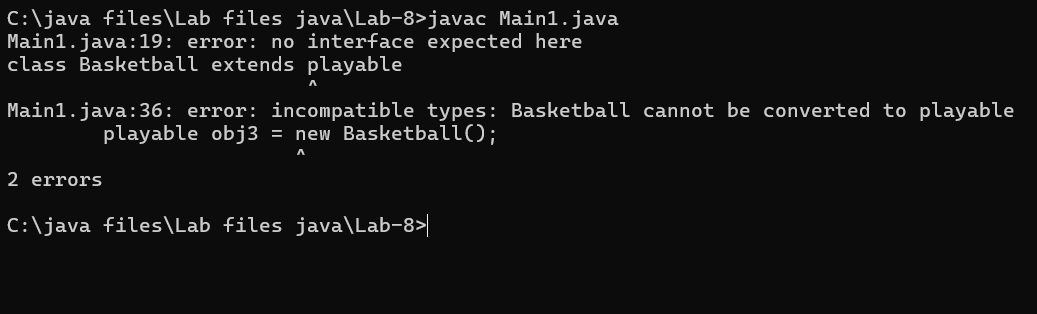
obj3.play();

}

}

**Output**

**Positive case:**  


**Negative Case:**  


**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| **1** | File not found exception | Wrong selection of path | Select path correctly |
| **2** | Instantiation of abstract class | Trying to create an object of abstract class without any subclass or interface | Use subclass or interface implementation |
| **3** | Static reference to Non-static Method | Calling non-static method from static context | Create object or make method static |

**Important points:**  
**Interface Implementation**: The playable interface defines the play() method, which is implemented by

football, volleyball, and basketball classes, each representing a different sport.

**Polymorphism in Action:**  
The objects of football, volleyball, and basketball are all treated as playable types, allowing the play()

method to be called polymorphically.