Object Oriented Programming (23CSE111)

Assignment

Submitted by	
Name	S.Mohana Maheswari
Roll No	AV.SC.U4CSE24314
Year/Sem/Section	1st/2nd/CSE-A[D]
Date of Submission	
Submitted to	
Name	Dr. B Raj Kumar
Department	CSE
Designation	Asst. Professor

Marks	

S.NO	Programs	Date	Page No	Signature
Week 1		27-01-2025		
1	Write the steps to download and install Java.			
2	Write a java program to print the message “Welcome to java programming”.			
3	Write a java program that prints student details(name, roll number and section of a student).			
Week 2		3-02-2025		
1	Write a java program to calculate the area of a rectangle.			
2a)	Write a program to convert temperature from Celsius to Fahrenheit			
2b)	Write a program to convert temperature from Fahrenheit to Celsius.			
3	Write a program to calculate the simple interest			
4	Write a program to find the largest of three numbers using ternary operator			
5	Write a program to find the factorial of a number			

Week-3		11-02-2025		
1.	Create a java program with following instructions a)create a class with name car b) Create 4 attributes name car color, car brand, fuel type, milage. c) Create 3 methods named start, stop, services d) Create 3objects named car1, car2, car3. e) Create a constructor which should print “welcome to car garage”			
2.	Write a java program to create a class BackAccount with two methods deposit( ) and withdraw() b) In deposit( ) whenever an amount is deposited it has to be updated with current amount b) In withdraw( ) whenever an amount is withdrawn it has to be less than current amount else print “Insufficient funds”			
Week-4		02-03-2025		
1.	Write a java program with class named “Book”. The class should contain various attributes such as			

	“Title of the book , author , year of publication “. It should also contain a constructor with parameters details of the book. i.e. “ Title of the book, author and year of publication”. Display the details of two books by creating two objects.			
2.	To create a java program with class named Myclass with a staticvariable “Count” of “int type”, Initialized to 0 and a constant variable “pi” of type double initialized to 3.1415 as attributes of that class Now, define a constructor for “Myclass” that increments the “Count” variable each that an object of Myclass is created. Finally , print the final values of “Count” and “pi”variables.			
Week-5		09-03-2025		
1.	Create a calc using the operations including add, sub, mul, div using multilevel inheritance and display the desired output			

2.	Creating a Rental Sysytem			
Week-6		16-03-2025		
1.	Write a java program to create a Vehicle class with displayInfo() method , overridden in Car subclass to provide info about carcompany , model , price ,seating and petrol.			
2.	An automated admission system that verifies student eligibility for UG and PG with different criteria. 1.UG requires minimum of 60% 2.PG requires minimum of 70%			
3.	Create a calculator class with overloaded methods to perform additions 1.add two integers 2.add two double values 3.add three integers			
4.	Create a shape class with method calculateArea() that is overloaded for different shapes (eg: square, rectangle).Then create a subclass Circle that overrides			

	calculateArea() method for Circle.			

WEEK-1

1)How to install the java and download java software ?

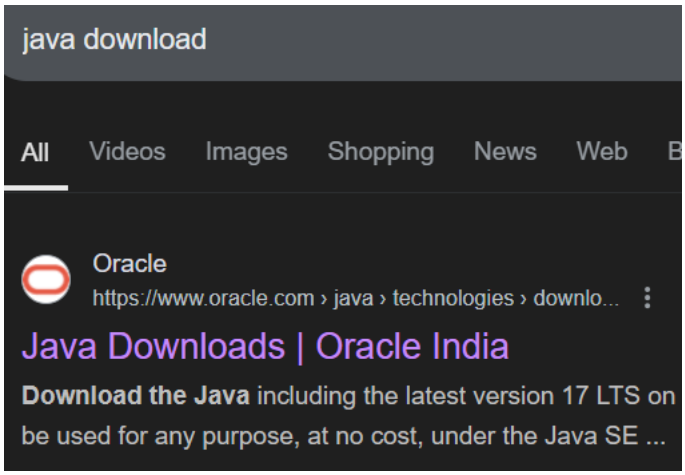
Aim: To Explain the process of Installing JDK (Java Development Kit)

Procedure:

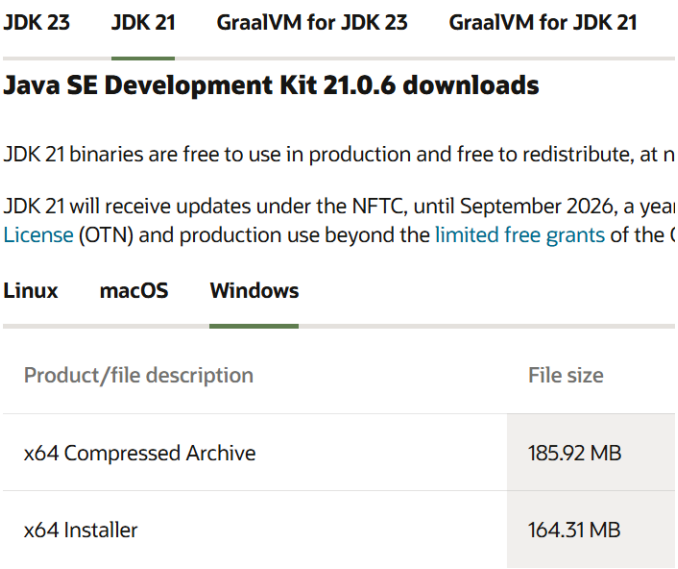
Installing of JDK (Java Development Kit):

1. First go to google and search java Download

2 . Go to the Oracle JDK download page in your web browser



3. click on JDK-21 version which is Long term support (LTS) version.

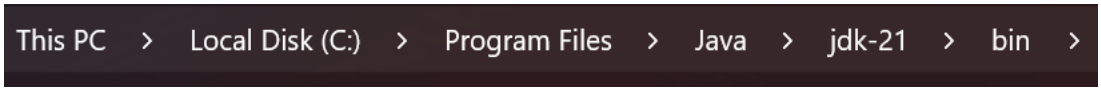


4. Click on the download link for your operating system (Windows, macOS, or Linux).

5.Next move on to installing JDK

6. Once downloaded, run the installer.Follow the instructions and keep clicking "Next" until it's done.

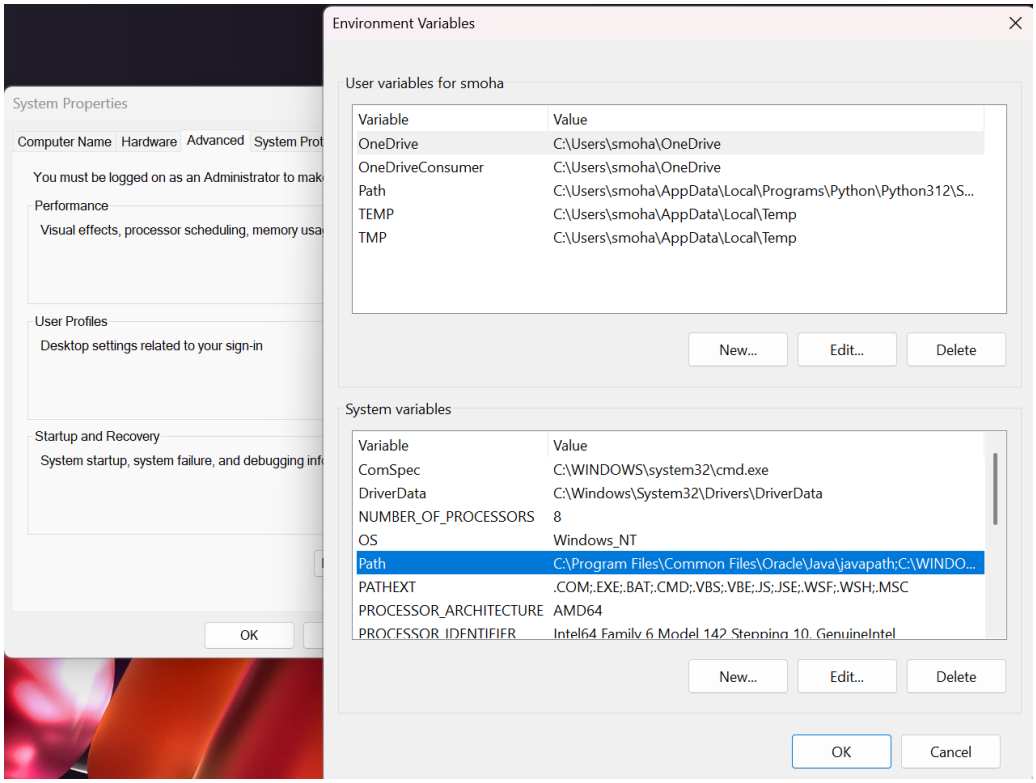
7.Follow this Navagation ,at last we get an link, then copy past that link in path



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

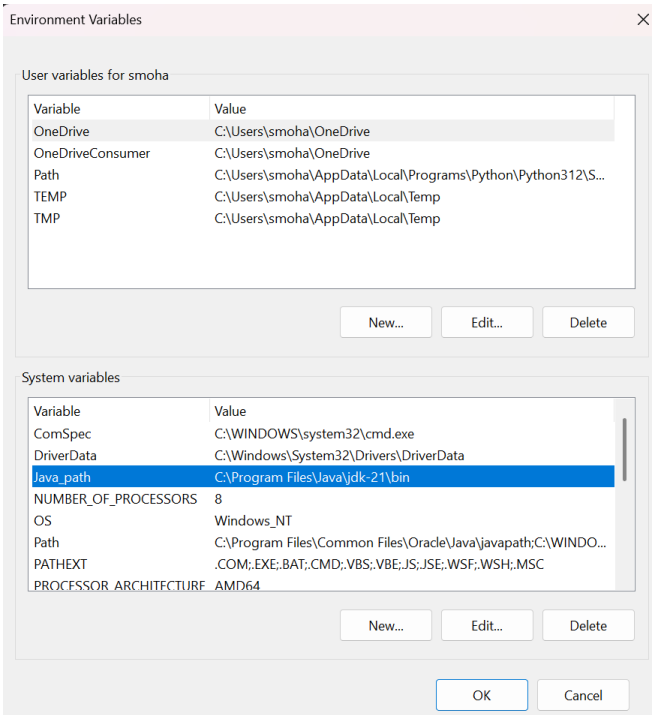
8.Search for Environment variables in search option which is below the pc. Then click on **Environment Variables**.

9. this may dispay on our screen after we click on environment variables



10. To set up the Click **New** under **System Variables**:

- **Set Variable name as:** java_path
- **Variable value:** The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)
- After this click on ok



11. Checking of JDK Version:

1. **Open Command Prompt:**
    - Press win+R, type cmd, and press Enter.
2. **Check Version:**
    - Type java --version and press Enter.
    - Type javac --version and press Enter.

```
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Java\jdk-21\bin>javac --version
javac 21.0.6

C:\Program Files\Java\jdk-21\bin>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)

C:\Program Files\Java\jdk-21\bin>
```

2) Write a Java Program to print the message “Welcome to Java programming”.

AIM: To write a Java Program to print the message “Welcome to Java programming”.

CODE:

```
//Define a class named Example1
  class Example1{
        public static void main(String[]args){
              System.out.println("Welcome to java programming");
  }}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac Example1.java

C:\Users\smoha\Java Prgms>java Example1
Welcome to java programming
```

Write a java program that prints Name , Roll no and Section of a student

AIM: To Write a java program that prints Name , Roll no and Section of a student

CODE:

```
class Example2{
    public static void main(String[]args){
        System.out.println("Name:Mohana Maheswari");
        System.out.println("Section: CSE-A");
        System.out.println("Roll No : 24314");
    }}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac Example2.java

C:\Users\smoha\Java Prgms>java Example2
Name:Mohana Maheswari
Section: CSE-A
Roll No : 24314
```

WEEK-2

3.Write a java program to calculate area of rectangle:

Code:

```
import java.util.Scanner;


class rectangle {
   public static void main(String[] args) {


      Scanner input = new Scanner(System.in);
       System.out.print("Enter a : ");
      float a = input.nextFloat();
      System.out.println(" entered a value  = " + a);


      System.out.print("Enter b: ");
      float b = input.nextFloat();
      System.out.println(" entered b value = " + b);
      float A=a*b
      System.out.println("Area of Rectangle is "+A);
    }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  rectangle.java
rectangle.java:18: error: ';' expected
        float A=a*b
                 ^
1 error
```

Error: Here there is an one error that is semicolon(;) is not present at the end of the expression.

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error:expected	Keep the semicolon at the end.

Output: After the error is rectified with semicolon(;)

```
C:\Users\smoha\Java Prgms>javac  rectangle.java

C:\Users\smoha\Java Prgms>java  rectangle
Enter a : 4.2
 entered a value  = 4.2
Enter b: 2.0
 entered b value = 2.0
Area of Rectangle is 8.4
```

Important points:

Class Definition: The code defines a class named rectangle.

Main Method: The main method is the entry point for the program. It contains the logic for user input and calculations.

User Input: The code uses a Scanner object to get user input. It prompts the user to enter two float values (a and b).

Questions:

What is the purpose of the main method in Java?

- The main method in Java serves a crucial purpose as the entry point of any standalone Java application. When the Java Virtual Machine (JVM) starts running a program, it looks for the main method to begin execution.

2.Write a java program to convert temperature from fahrenhit to celsius:

Formula:(F-32)*5/9

Code:

```
import java.util.Scanner;


public class FahrenheitToCelsius {

    public static void main(String[] args) {

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

```
        System.out.print("Enter temperature in Fahrenheit: ");
        double fahrenheit = input.nextDouble();

        double celsius = (fahrenheit - 32 )* 5 / 9;

        System.out.println("Temperature in Celsius: " + celsius);

        input.close();
    }
}
```

```
C:\Users\smoha\Java Prgms>javac  FahrenheitToCelsius.java
FahrenheitToCelsius.java:7: error: package system does not exist
        system.out.print("Enter temperature in Fahrenheit: ");
               ^
1 error
```

Error:Here there is one error that is “s” in System is small letter, but it should be always Capital letter “S”

S.no	ERROR MESSAGE	ERROR RECTIFICTION
	Package system does not exist.	**Keep Catical S instead of small s.**

Output: **After the error is rectified**

```
C:\Users\smoha\Java Prgms>javac  FahrenheitToCelsius.java

C:\Users\smoha\Java Prgms>java  FahrenheitToCelsius
Enter temperature in Fahrenheit: 86
Temperature in Celsius: 30.0
```

Important points:

Import Statement:

- The import java.util.Scanner; statement imports the Scanner class from the java.util package. The Scanner class is used to read input from various input sources, including user input from the console.

Class Declaration:

- The class is named FahrenheitToCelsius, and it contains a main method.

Main Method:

- The main method is the entry point of the program. It is where the program execution starts.

Question:

How is the Scanner class used in this code?

- **Answer**: The Scanner class is used to get user input from the console. An instance of Scanner is created to read the input temperature in Fahrenheit

2.Write a java program to convert temperature from Celsius to Fahrenheit:

Formula:

Code:

```
import java.util.Scanner;


public class CelsiusToFahrenheit {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);


      System.out.print("Enter temperature in Celsius : ");
      double Celsius = input.nextDouble();


      double Fahrenheit = (Celsius*9/5)+32;


      System.out.println("Temperature in  Fahrenheit : " +  Fahrenheit );
```

```
        input.close();
    }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  CelsiusToFahrenheit.java
CelsiusToFahrenheit.java:5: error: cannot find symbol
        Scanner input = new scanner(System.in);
                            ^
  symbol:   class scanner
  location: class CelsiusToFahrenheit
1 error
```

Error: here the error is " S" in Scanner is small letter(s),but it should be in capital letter(S).

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error:cannot finf symbol.	S" in Scanner is small letter(s),but it should be in capital letter(S).

Output: **After the error is rectified**

```
C:\Users\smoha\Java Prgms>javac  CelsiusToFahrenheit.java

C:\Users\smoha\Java Prgms>java    CelsiusToFahrenheit
Enter temperature in Celsius : 20
Temperature in  Fahrenheit : 68.0
```

Important points:

Import Statement:

- The import java.util.Scanner; statement imports the Scanner class from the java.util package. The Scanner class is used to read input from various input sources, including user input from the console.

Class Declaration:

- The class is named CelsiusFahrenheitTo, and it contains a main method.

Main Method:

- The main method is the entry point of the program. It is where the program execution starts.

Question:

Why is it important to close the Scanner object in this code?

- **Answer**: Closing the Scanner object releases the resources associated with it, preventing resource leaks. This is done using the input.close() method.

4.Write the java program to calculate the simple interest:

Formula:P*T*R/100

Code:

```
import java.util.Scanner;


class simpleinterest {
   public static void main(String[] args) {


       Scanner input = new Scanner(System.in);



       System.out.print("Enter P : ");
       float P = input.nextFloat();
s       System.out.println(" entered P value  = " + P);



       System.out.print("Enter T: ");
       float T = input.nextFloat();
       System.out.println(" entered T value = " + T);



       System.out.print("Enter R: ");
       float R = input.nextFloat();
```

```
        System.out.println(" entered R value = " + R);


        float Simpleinterest=P*T*R/100;
        System.out.println("Simple interest is:"+Simpleinterest);
    }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  simpleinterest.java
simpleinterest.java:23: error: cannot find symbol
        float Simpleinterest=P*t*R/100;
                             ^
  symbol:   variable t
  location: class simpleinterest
1 error
```

Error: Here the error is variable “t” is not found ,instead we have to keep “T”.

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error:cannot find symbol	Variable “t” is not found ,instead we have to keep “T”.

Output: **After the error is rectified**

```
C:\Users\smoha\Java Prgms>javac  simpleinterest.java

C:\Users\smoha\Java Prgms>java  simpleinterest
Enter P : 9000
 entered P value  = 9000.0
Enter T: 5
 entered T value = 5.0
Enter R: 0.07
 entered R value = 0.07
Simple interest is:31.5
```

Important points:

Class Definition: The code defines a class named simpleinterest.

Main Method: The main method is the entry point for the program, containing the logic for user input and calculations.

User Input: The code uses a Scanner object to get user input. It prompts the user to enter three float values (P, T, and R).

Question:

How can you improve this code to handle invalid input?

- **Answer**: You can add a loop with a try-catch block to repeatedly prompt the user until valid input is entered:

5.Write a java program for largest of two number using ternary operator:

Code:

```
import java.util.Scanner;


class largest {
  public static void main(String[] args) {



    Scanner input = new Scanner(System.in);
    System.out.println("Enter your n1: ");
    int n1 = input.nextInt();


    System.out.println("Enter your n2: ");
    int n2 = input.nextInt();


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


    System.out.println("Largest Number: " + largest);
```

```
  }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  largest.java
largest.java:11: error: variable input is already defined in method main(String[])
    Scanner input = new Scanner(System.in);
            ^
1 error
```

Error: Variable input is already defines in method main(String []).

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Variable input is already defines in method main(String []).	Variable input is already defines in method main(String []).

Output:

```
C:\Users\smoha\Java Prgms>javac  largest.java

C:\Users\smoha\Java Prgms>java  largest
Enter your n1:
10
Enter your n2:
5
Largest Number: 10
```

Important points:

Class Definition: The code defines a class named largest.

User Input: The code uses a Scanner object to get user input. It prompts the user to enter two integer values (n1 and n2).

Conditional (Ternary) Operator: The code uses the ternary operator to determine which of the two input numbers is larger. The expression (n1 >= n2) ? n1 : n2 assigns the larger value to the largest variable.

Resource Management: The Scanner object is not closed in the code, which is good practice to prevent resource leaks.

Question:

How does the Scanner class work for getting user input in this code?

- **Answer**: The Scanner class is used to get user input from the console. An instance of Scanner is created to read the input values n1 and n2.

6.Write a java program to find factorial of a number:

Code:

```
import java.util.Scanner;


class factorial {
  public static void main(String[] args) {


    Scanner input = new Scanner (System.in);
    System.out.println("Enter n value: ");
    int number = input.nextInt();
    System.out.println("You entered " + number);
    int fact=1;
    for(int i=1;i<=number;i++){
    fact=fact*i;
    System.out.println(fact);
    }
  }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  factorial.java
factorial.java:9: error: cannot find symbol
    System.out.println("You entered " + number);
                                         ^
  symbol:   variable number
  location: class factorial
factorial.java:13: error: cannot find symbol
    for(int i=1;i<=number;i++){
                    ^
  symbol:   variable number
  location: class factorial
2 errors
```

Error:there are two errors number variable is not indicated properly.

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error:cannot find symbol	number variable should indicate properly.

Output:

```
C:\Users\smoha\Java Prgms>javac  factorial.java

C:\Users\smoha\Java Prgms>java  factorial
Enter n value:
5
You entered 5
1
2
6
24
120
```

Important points:

Class Definition: The code defines a class named factorial.

Main Method: The main method is the entry point for the program, containing the logic for user input and factorial calculation.

User Input: The code uses a Scanner object to get user input. It prompts the user to enter an integer value (number).

Factorial Calculation: The code calculates the factorial of the given number using a for loop. The factorial is computed as the product of all integers from 1 to the input number.

Question:

How is the factorial of a number calculated in this code?

- **Answer**: The factorial is calculated using a for loop that multiplies the value of fact by the loop variable i for each iteration from 1 to number.

WEEK-3

1) PROGRAME-1

AIM: To create a java program with the following instructions:

a)Create a class with name “Car”

b)Create 4 attributes, named: car_color, car_brand, fuel_type, mileage

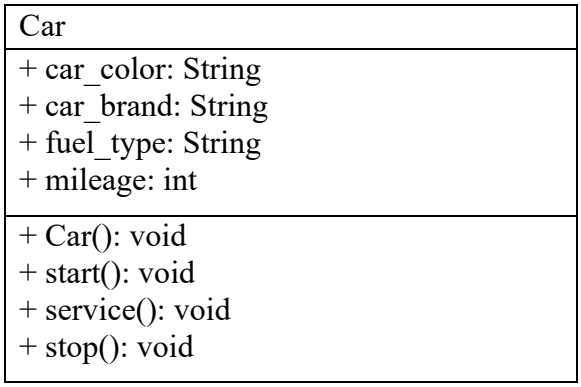
c)Create 3 methods, named: start(), service(), stop()

d)Create 3 objects, named: car1, car2, car3

Create a constructor, which should print, “Welcome to car garage” .

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

Class Diagram:



Code:

```
class car{
//creating the attributes required fo the class
String car_color,car_brand,fuel_type;
int mileage;
//constructor
```

```
car(String car_color, String car_brand, String fuel_type,int maleage){
this. car_color=car_color;
this.car_brand=car_brand;
this. fuel_type=fuel_type;
this.mileage=mileage;
}

//creating methods for the class

public void start(){
 System.out.println("The "+car_brand+ " "+"car gets started" +" "+ "which is in"+" "+
car_color +"color");
}
public void stop(){
 System.out.println("The car is stopped due to less"+" " + fuel_type);
}
public void service(){
 System.out.println("The car is in servicing " +"has "+ mileage+" "+ "mileage");
}

public static void main(String[] args) {
//creating the objects for the class
car car1=new car("navy blue","maruthi","petrol",300);
car1.start();

car car2=new car("navy blue","Honda","petrol",400);
car2.stop();

car car3=new car("black","maruthi","petrol",500);
```

```
car3.service();


}
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  car.java
car.java:27: error: ')' or ',' expected
car car1=new car("navy blue","maruthi""petrol",300);
                                      ^
1 error
```

Error: Here there should be coma in between that two quotation marks.

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error: ')' or ' , ' expected	Keep coma in between that two quotation marks.

Output:

```
C:\Users\smoha\Java Prgms>javac  car.java

C:\Users\smoha\Java Prgms>java  car
The maruthi car gets started which is in navy bluecolor
The car is stopped due to less petrol
The car is in servicing has 500 maleage
```

Important points:

 Constructor: The constructor initializes the attributes with values provided as arguments.

Methods: The class has three methods: start, stop, and service. Each method performs a specific action and prints a message related to the car.

Main Method: The main method creates objects of the car class and calls the methods on these objects to demonstrate their functionality.

Syntax Error: There is a syntax error in the code. The car1 object creation line has an extra double-quote character. It should be car car1 = new car("navy blue", "maruthi", "petrol", 300);.

Question:

How are objects of the car class created in the main method?

Answer: Objects of the car class are created using the new keyword, followed by a call to the constructor with the appropriate arguments. For example: car car1 = new car("navy blue", "maruthi", "petrol", 300);.

2) AIM: To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

a)deposit(): Whenever an amount is deposited, it has to be update the current amount.

b) withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print ("Insufficient funds")

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

Class Diagram:

BankAccount
- existing: float - input: Scanner + name: String
+ BankAccount() + deposit(): void + withdrawal(): void + main(args: String[]): void

Code:

```
import java.util.Scanner;

class BankAccount {

 // Class-level variable to store balance

    private float existing;

    private Scanner input; // Single Scanner instance for input
```

```
public  String name;
// Constructor
public BankAccount() {
    input = new Scanner(System.in);
    System.out.println("Enter the account holder name :");
    this.name=input.next();
    System.out.print("Enter existing amount in bank account: ");
    this.existing = input.nextFloat();
}
// Deposit method
public void deposit() {
    System.out.print("Enter amount to be deposited: ");
    float deposit = input.nextFloat();
    existing += deposit;
    System.out.println("Existing amount now is: " + existing);
}
// Withdrawal method
public void withdrawal() {
    System.out.print("Enter amount to be withdrawn: ");
    float withdrawal = input.nextFloat();
    if (existing < withdrawal) {
        System.out.println("Not sufficient balance.");
    } else {
        existing -= withdrawal;
        System.out.println("Remaining balance: " + existing);
    }
}
// Main method
public static void main(String[] args) {
```

```
        BankAccount customer1 = new BankAccount();
        customer1.deposit();
        customer1.withdrawal();
        System.out.println("thank you " + customer1.name + " for using our bank");
}}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  BankAccount.java
BankAccount.java:39: error: ';' expected
        System.out.println("thank you " + customer1.name + " for using our bank")
                                                                                ^
1 error

C:\Users\smoha\Java Prgms>
```

Error:

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error: ‘;’ expected	Spectify the semicolon at the end.

```
C:\Users\smoha\Java Prgms>javac  BankAccount.java

C:\Users\smoha\Java Prgms>java  BankAccount
Enter the account holder name :
Mohana
Enter existing amount in bank account: 10000
Enter amount to be deposited: 5000
Existing amount now is: 15000.0
Enter amount to be withdrawn: 3000
Remaining balance: 12000.0
thank you Mohana for using our bank
```

IMPORTANT POINTS:

- Java constructor is used to save the variables present in different or same class or methods.
- In Java, the this keyword refers to the current instance of a class. It is commonly used to distinguish between instance variables and parameters with the same name, or to refer to the current object from within a method or constructor.
- In Java, a method is a block of code that performs a specific task and can be invoked to execute that task. It typically consists of a method signature (name, return type, and parameters) and the body of the method, which contains the logic.

WEEK-4

1) PROGRAME-1

AIM: Write a java program with class named “book” the class should contain various attributes such as the title of the book, author, year of publication. It should also contain a constructor with parameter which initializes the title of the book, author, year of publication. Create a method which displays the details of the book ie title , author, year. Display the details of 2 books by creating 2 objects

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

Class Diagram:

Book
- titleOfTheBook: String - author: String - yearOfPublication: int
+ Book(title: String, author: String, year: int) |+ getTitle(): void + getAuthor(): void + getYearOfPublication(): void

```
class book{

String titleofthebook;

String Author;

int yearofpublication;


//creating constructor//


book(String titleofthebook,String Author,int yearofpublication){

this.titleofthebook=titleofthebook;

this.Author=Author;

this.yearofpublication=yearofpublication;
```

```
}

//creating a methods//
public void titleofbook(){
System.out.println("The Title of book is :"+titleofthebook);
}

public void Author(){
System.out.println("The Author of book is :"+Author);
}

public void yearofpublication(){
System.out.println("The book is published in the year :"+yearofpulication);
}

//creating objects//
public static void main(String[] args){
book b1=new  book("the story of honey","Priya",2020);
b1.titleofbook();
}
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  book.java
book.java:30: error: <identifier> expected
public static void(String[] args){
                  ^
1 error
```

Error:

S.no	ERROR MESSAGE	ERROR RECTIFICTION

1	Error: <indentifier> expected	Spectify the main function.

```
C:\Users\smoha\Java Prgms>javac  book.java

C:\Users\smoha\Java Prgms>java  book
The Title of book is :the story of honey
The Author of book is :Priya
The book is published in the year :2020
```

Important points:

Class Declaration: The class is named book and contains three instance variables: titleofthebook, Author, and yearofpublication.

Constructor: The constructor initializes the instance variables with the values passed as parameters.

Main Method: Creates an instance of the book class and calls the titleofbook() method to display the title.

Question:

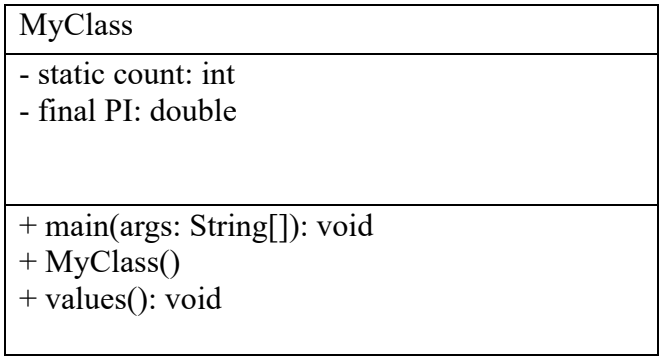
Identify the typo in the code and explain its impact.

- **Answer**: There is a typo in the yearofpublication() method. The instance variable yearofpulication should be yearofpublication. Due to this typo, the method will not print the year of publication correctly, and the code will not compile if the method is called.

2) AIM: To create a java program with class named “my class” with a static variable “count” of int type,initialized to zero and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class. Now define a constructor for my class that increments the count variable each time and object of my class is created. Finally print the final value of “count” and “pi.

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

Class Diagram:



Code:

```
class myclass{
//creating the variables

static int count=0;
final double pi=3.1415;
//creating a constructor
myclass(){
count++;
}
//method to print the values
public void values(){
System.out.println(+count);
System.out.println(+pi);
}
//object and the main function
public static void main(String[] args){
myclass one=new myclass();
one.values();
myclass two=new myclass();
two.values();
myclass three=new myclass();
three.values();
```

myclass four=new myclass();

four.values();

}}

Output:

```
C:\Users\smoha\Java Prgms>javac  myclass.java
myclass.java:23: error: ';' expected
myclass four=new myclass()
                         ^
1 error
```

Error:

Error:

S.no	ERROR MESSAGE	ERROR RECTIFICTION
1	Error: ‘;’ expected	Spectify the semicolon at the end.

Output:

```
C:\Users\smoha\Java Prgms>javac  myclass.java

C:\Users\smoha\Java Prgms>java  myclass
1
3.1415
2
3.1415
3
3.1415
4
3.1415
```

IMPORTANT POINTS:

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

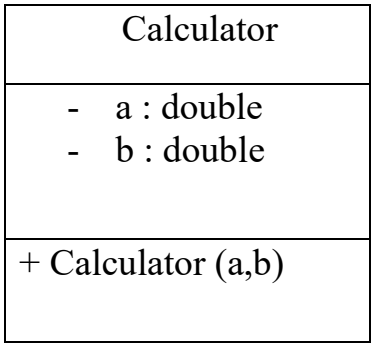
WEEK 05

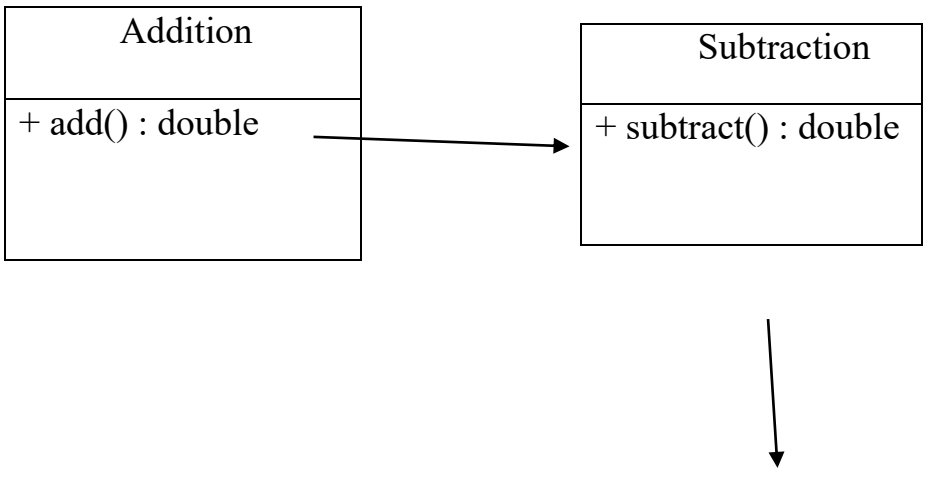
PROGRAM-1:

AIM: Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.

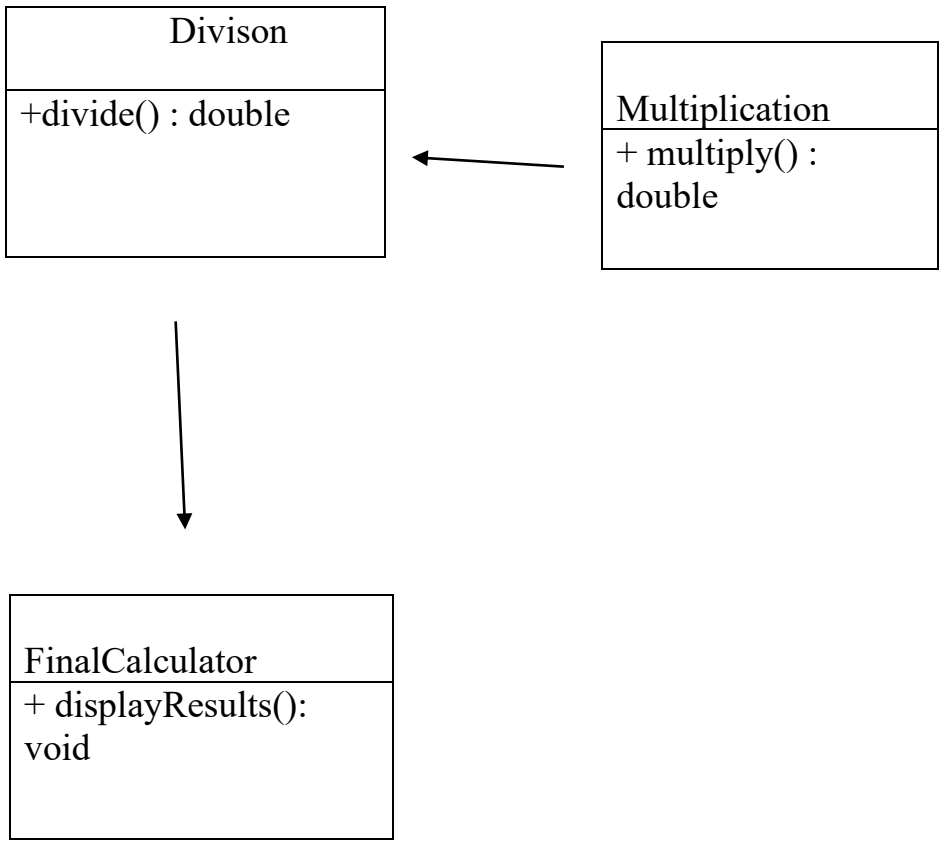
Hint: collect required variables using super class, create each class for a parameter and each class must contain a method.

CLASS DIAGRAM:





z



CODE:

```
1-class Calculator {
    protected double a, b;

    public Calculator(double a, double b) {
        this.a = a;
        this.b = b;
    }
}

class Addition extends Calculator {
    public Addition(double a, double b) {
        super(a, b);
    }
```

```
    public double add() {
        return a + b;
    }}
class Subtraction extends Addition {
    public Subtraction(double a, double b) {
        super(a, b);
    }
    public double subtract() {
        return a - b;
    }}
class Multiplication extends Subtraction {
    public Multiplication(double a, double b) {
        super(a, b);
    }
    public double multiply() {
        return a * b;
    }}
class Division extends Multiplication {
    public Division(double a, double b) {
        super(a, b);
    }
    public double divide() {
        if (b != 0) {
            return a / b;
        } else {
            System.out.println("Error: Division by zero");
            return Double.NaN;    }}}
public class FinalCalculator extends Division {
    public FinalCalculator(double a, double b) {
```

```
        super(a, b);
    }
    public void displayResults() {
        System.out.println("Addition: " + add());
        System.out.println("Subtraction: " + subtract());
        System.out.println("Multiplication: " + multiply());
        System.out.println("Division: " + divide());
    }
    public static void main(String[] args) {
        FinalCalculator calc = new FinalCalculator(10, 2);
        calc.displayResults();
    }}
```

```
C:\Users\smoha\Java Prgms>javac FinalCalculator.java

C:\Users\smoha\Java Prgms>java   FinalCalculator
Addition: 12.0
Subtraction: 8.0
Multiplication: 20.0
Division: 5.0
```

ERRORS:

Code Error	Code rectification
1. not providing the return method correctly. 2. Not mentioning super to obtain the super class constructor.	1. After declaring methods, we must provide the return method correctly. 2. To obtain the super class we need to mention super.

PROGRAM-2:

AIM: 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
3. 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.

IMPORTANT POINTS:

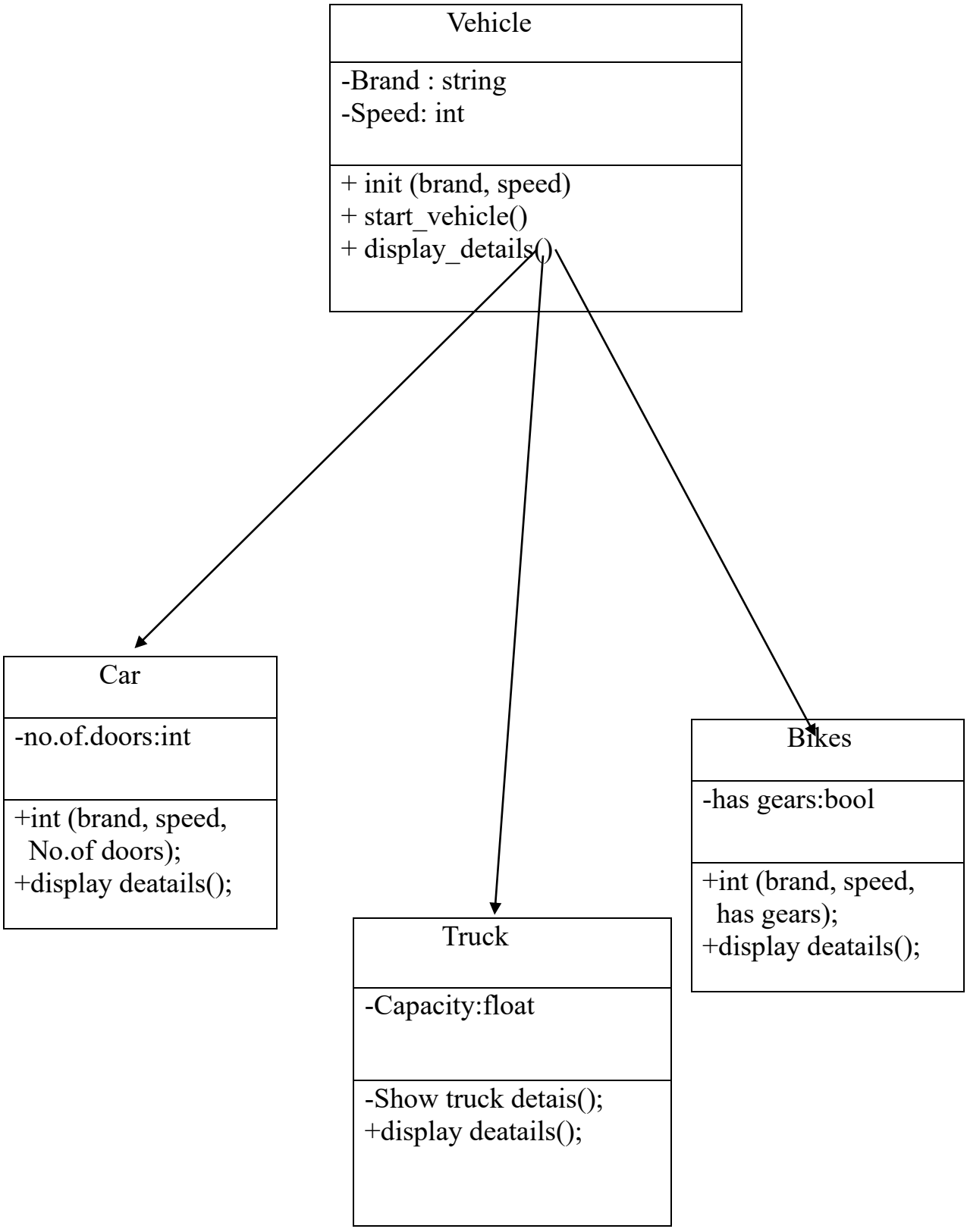
1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. the void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

Answer for Q1:

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

CLASS DIAGRAM:



CODE:

```
class Vehicle {
```

```
    private String brand;
    private int speed;

    Vehicle(String brand, int speed) {
        this.brand = brand;
        this.speed = speed;
    }void details() {
        System.out.println("Brand: " + brand);
        System.out.println("Speed: " + speed);
    }
}

class Car extends Vehicle {
    private int doors;
    private int capacity;

    public Car(String brand, int speed, int doors, int capacity) {
        super(brand, speed);
        this.doors = doors;
        this.capacity = capacity;
    }

    void carDetails() {
        System.out.println("Number of doors: " + doors);
        System.out.println("Capacity: " + capacity);
    }

    @Override
    void details() {
```

```
        super.details();
        carDetails();
    }
}

class Bike extends Vehicle {
    private boolean gears;

    Bike(String brand, int speed, boolean gears) {
        super(brand, speed);
        this.gears = gears;
    }

    void bikeDetails() {
        System.out.println(gears ? "This bike has gears." : "This bike does not have gear
system.");
    }

    @Override
    void details() {
        super.details();
        bikeDetails();
    }
}
class Truck extends Vehicle {
    private int tons;
    Truck(String brand, int speed, int tons) {
        super(brand, speed);
        this.tons = tons;
```

```
    }
    void truckDetails() {
        System.out.println("The capacity of truck is: " + tons + " tons.");    }
    @Override
    void details() {
        super.details();
        truckDetails();
    }
}

public class Rent {
    public static void main(String[] args) {
        Car c = new Car("Toyota", 100, 5, 5);
        c.details();

        Bike b = new Bike("KTM", 90, true);
        b.details();

        Truck t = new Truck("TATA", 80, 1);
        t.details();
    }
}
```

```
C:\Users\smoha\Java Prgms>javac  Rent.java

C:\Users\smoha\Java Prgms>java Rent.java
Brand: Toyota
Speed: 120
Number of doors: 5
Capacity: 5
Brand: KTM
Speed: 80
This bike has gears.
Brand: TATA
Speed: 100
The capacity of truck is: 1 tons.
```

ERROR TABLE:

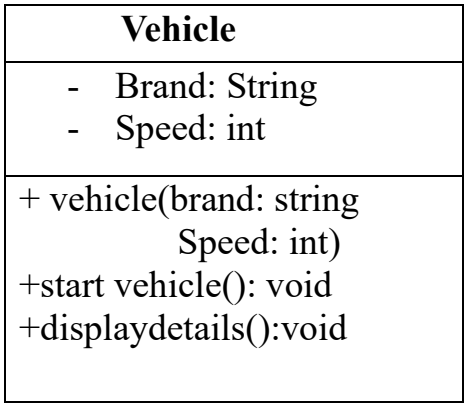
Code Error	Code rectification
1. Declaring two superclasses inside the same file. 2. Not declaring the variable using 'this' keyword inside the constructor.	1. Make two separate files to save the two super classes. 2. Declare the variable using this keyword to run the program.

WEEK-06

PROGRAM-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 (car company, seating capacity, petrol or not).

CLASS DIAGRAM:



CODE:

```
class Vehicle {
    String car_company;
    String car_model;
    long car_price;
    int seating_capacity;
    boolean petrol;

    Vehicle(String car_company, String car_model, long car_price, int seating_capacity,
boolean petrol) {
        this.car_company = car_company;
        this.car_model = car_model;
        this.car_price = car_price;
        this.seating_capacity = seating_capacity;
```

```
        this.petrol = petrol;
    }

    void displayInfo() {
        System.out.println("Car company: " + car_company);
        System.out.println("Car model: " + car_model);
        System.out.println("Car price: " + car_price);
        System.out.println("Car seating capacity: " + seating_capacity);
        System.out.println("Car uses petrol: " + petrol);
    } }
class Car extends Vehicle {
    Car(String car_company, String car_model, long car_price, int seating_capacity, boolean
petrol) {
        super(car_company, car_model, car_price, seating_capacity, petrol);
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating a Car object with correct arguments
        Car c1 = new Car("Hyundai", "Creta", 1500000, 5, false);
        c1.displayInfo();
    }
}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac  Main.java

C:\Users\smoha\Java Prgms>java Main
Car company: Hyundai
Car model: Creta
Car price: 1500000
Car seating capacity: 5
Car uses petrol: false
```

ERRORS:

Code error	Code rectification
1. Incorrect class name for main method(Truck). 2. Inconsistent car model output in displayinfo().	1.Rename Truck to Main or place main inside car or vehicle. 2. Ensure Car correctly passes Toyota" to super(car_model,color,fueltype)

IMPORTANT POINTS:

1.Inheritance: The Car class extends the Vehicle class, demonstrating inheritance in Java.

2.Constructor Chaining: The Car class calls the parent constructor using super(car_model, color, fuel_type); to initialize inherited attributes.

3.Method Overriding: The Car class overrides the displayInfo() method from Vehicle and calls super.displayInfo() to reuse the parent method before adding its own output.

4.Incorrect main **Class Name:**The main method is inside Truck, which is unrelated to Vehicle and Car. The class should be renamed for clarity.

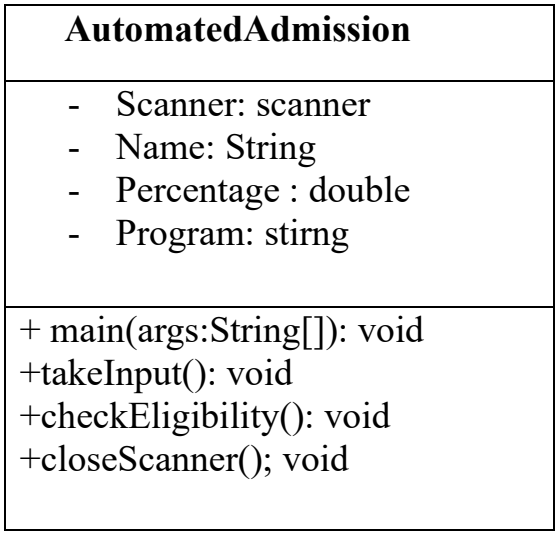
PROGRAM-2:

AIM: A college is developing an automated admission system that verifies students eligibility(UG) and postgraduation(PG) programs. Each program has different eligibility criteria based on the students percentage in their previous qualification.

1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

CLASS DIAGRAM:



CODE:

```
class Student {
    String name;
    double percentage;

    Student(String name, double percentage) {
        this.name = name;
        this.percentage = percentage;
    }

    void studentsInfo() {
        System.out.println("Student Name: " + name);
        System.out.println("Percentage: " + percentage);
```

```
    }  }
class UG extends Student {
    UG(String name, double percentage) {
        super(name, percentage);
    }
    void checkEligibility() {
        if (percentage >= 60) {
            System.out.println(name + " is eligible for admission in UG.");
        } else {
            System.out.println(name + " is not eligible for admission in UG.");
 } } }
class PG extends Student {
    PG(String name, double percentage) {
        super(name, percentage);
    }
    void checkEligibility() {
        if (percentage >= 70) {
            System.out.println(name + " is eligible for admission in PG.");
        } else {
            System.out.println(name + " is not eligible for admission in PG.");
        }  }  }

public class AutomatedAdmission {
    public static void main(String[] args) {
        UG ug = new UG("Maheswari", 80);
        ug.studentsInfo();
        ug.checkEligibility();

        PG pg = new PG("Lakshmi", 75);
```

```
        pg.studentsInfo();
        pg.checkEligibility();
    }
}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac  AutomatedAdmission.java

C:\Users\smoha\Java Prgms>java AutomatedAdmission
Student Name: Maheswari
Percentage: 80.0
Maheswari is eligible for admission in UG.
Student Name: Lakshmi
Percentage: 75.0
Lakshmi is eligible for admission in PG.
```

ERRORS:

Code error	Code rectification
1.Scanner nextLine() issue after nextDouble(): After scanner.nextDouble(), the newline character remains in the buffer, causing nextLine() to be skipped. **2.Program type input case sensitivity issue**: If the user enters ug or pg in lowercase, it may cause incorrect comparisons.	**1**.Add scanner.nextLine(); after nextDouble(); to consume the leftover newline. **2.**Use program.toUpperCase() to ensure case-insensitive comparison.

IMPORTANT POINTS:

1.User Input Handling: Uses Scanner to take user input for name, percentage, and program type.

2.Decision Making with Conditions: Uses if-else statements to check eligibility criteria.

3.String Handling: Converts program input to uppercase (toUpperCase()) to handle case variations.

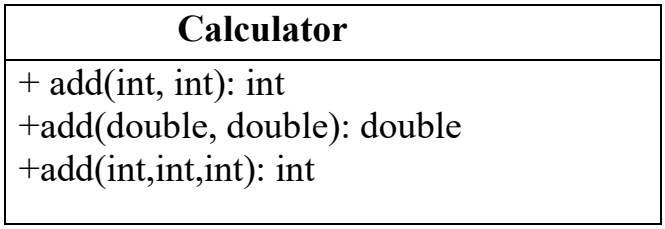
4.Closing Scanner: Properly closes scanner using scanner.close(); to prevent resource leaks.

PROGRAM-3:

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

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

CLASS DIAGRAM:

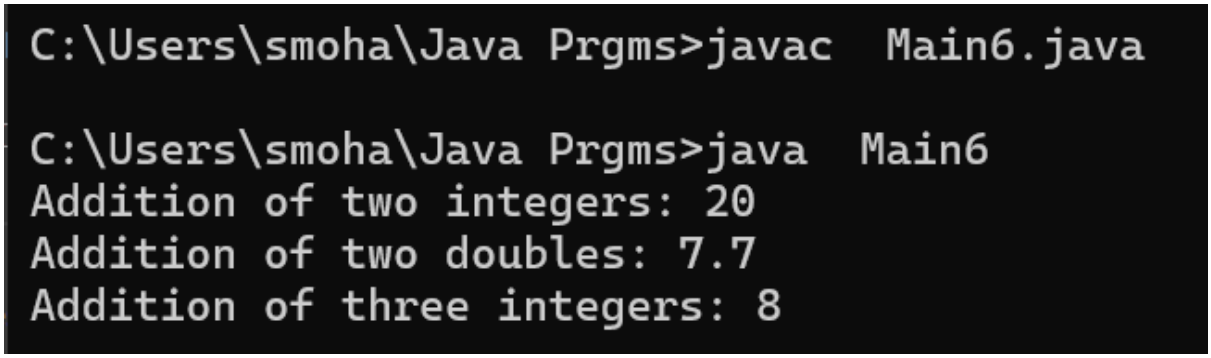


CODE:

```
class Calculator6 {
   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 Main6 {
    public static void main(String[] args) {
        Calculator6 calculator = new Calculator6();
        System.out.println("Addition of two integers: " + calculator.add(5, 15));
        System.out.println("Addition of two doubles: " + calculator.add(5.5, 2.2));
        System.out.println("Addition of three integers: " + calculator.add(1, 2, 5));
    } }
```

OUTPUT:



ERRORS:

Code error	Code rectification
1.Method parameters missing spaces. E.g.,"inta, intb"should be "int a, int b" 2.Inconsistent indentation in method bodies	1. Add proper spacing between parameters: (int a, int b) 2. Fix indentation: Consistent 4 space o indentation.

IMPORTANT POINTS:

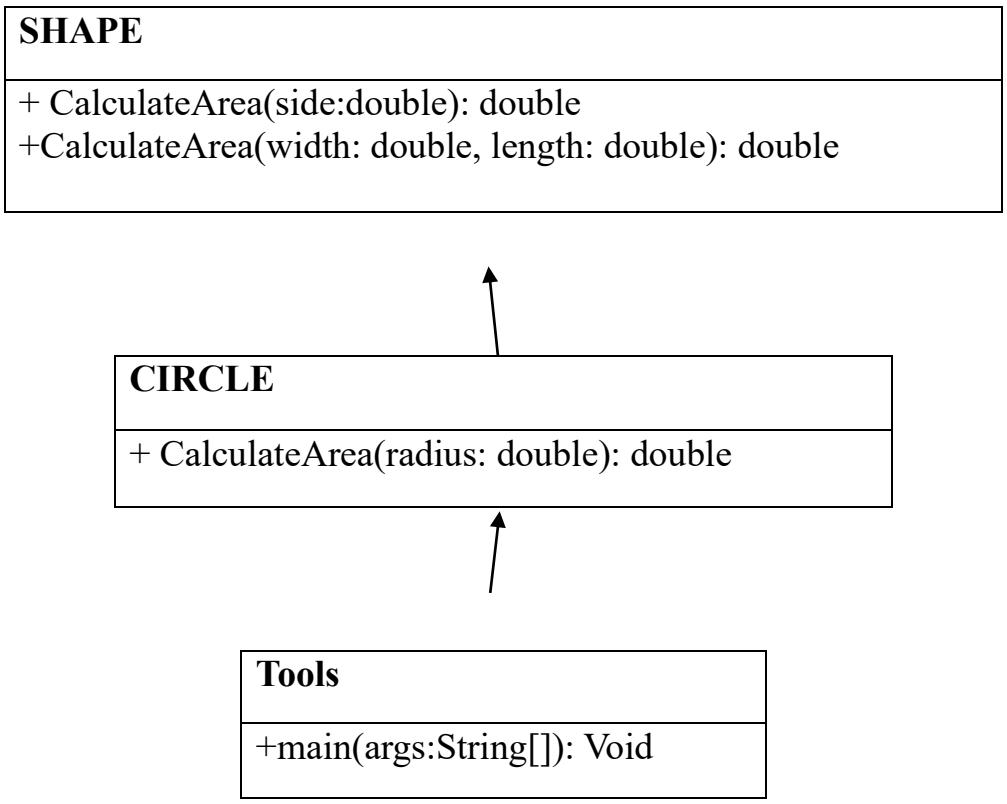
1.Method Overloading: The add method is overloaded with different parameter types and counts, demonstrating compile-time polymorphism.

2.Automatic Method Selection: Java selects the appropriate add method based on the argument types during compilation.

PROGRAM-4:

AIM: Create a shape class with a method to calculate area i.e., overloaded for different shapes eg: Squares, Recatangle. 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 double calculateArea(double length, double width) {
        return length * width;
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}

class Tools {
    public static void main(String[] args) {
        Shape shape = new Shape();
        Circle circle = new Circle(5);

        System.out.println("Area of square: " + shape.calculateArea(4));
        System.out.println("Area of rectangle: " + shape.calculateArea(4, 6));
        System.out.println("Area of circle: " + circle.calculateArea());
    }
```

}

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac  Tools.java

C:\Users\smoha\Java Prgms>java  Tools
Area of square: 16.0
Area of rectangle: 24.0
Area of circle: 78.53981633974483
```

ERRORS:

Code error	Code rectification
1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)). 2. Circle class method does not override theparent class method properly.	1. Use s.calculateArea(4) and c.calculateArea(2) to call the method correctly. 2. Ensure @Override is used, and the method signature should match correctly.

IMPORTANT POINTS:

1.Inheritance: Circle class extends Shape, inheriting its methods.

2.Method Overloading: Shape has multiple calculateArea methods with different parameters.

3.Method Overriding: Circle overrides calculateArea from Shape to implement its own formula.

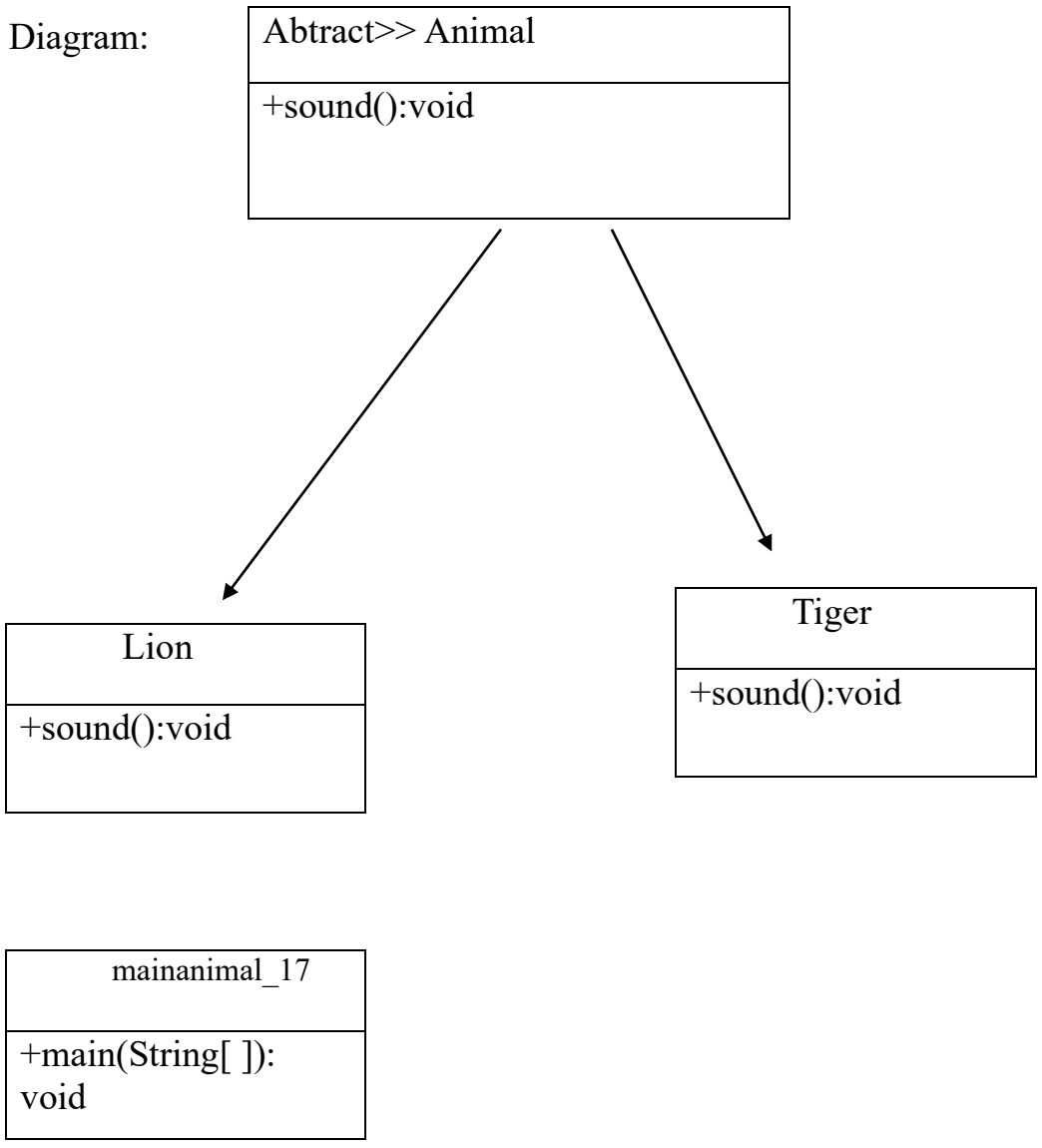
4.Polymorphism: The overridden method in Circle demonstrates runtime polymorphism.

5.Proper Object Reference: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).

WEEK-07

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

Class Diagram:



CODE:

```
abstract class animal{
  abstract void sound();
}
```

```
class  Lion  extends animal{
   public void sound(){
      System.out.println("Lion makes roar sound");
}}
 class Tiger extends animal{
   public void sound(){
    System.out.println("Tiger makes moaning sound");
}}
class mainanimal_17{
 public static void main(String[] args) {
System.out.println("Name:S Mohana Maheswari"+"; "+"Rollno:AV.SC.U4CSE24314"+"
;"+"Section:CSE-A");
  Tiger obj1=new Tiger();
  Lion obj2=new Lion();
  obj1.sound();
obj2.sound();
}}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac  mainanimal_17.java
mainanimal_17.java:2: error: ';' expected
  abstract void sound()
                      ^
1 error
```

ERROR:

Code Error	Code rectification
1. error: ';' expected	1. we must end line with semicolon.

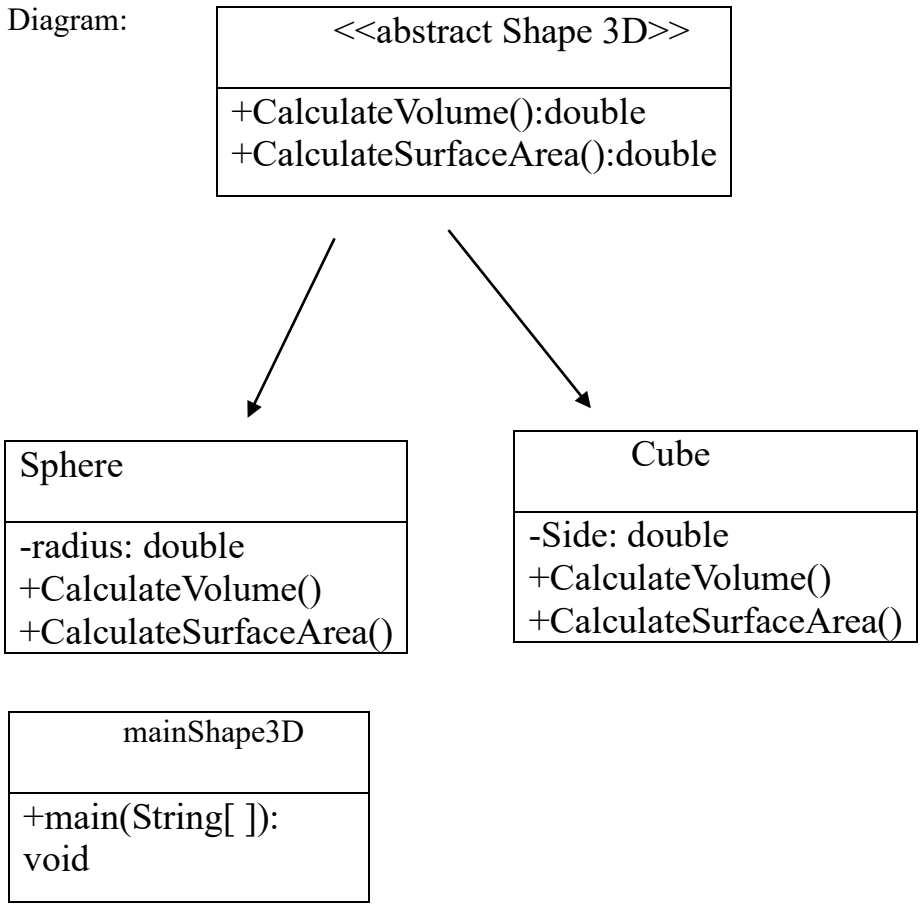
```
C:\Users\smoha\Java Prgms>javac  mainanimal_17.java

C:\Users\smoha\Java Prgms>java   mainanimal_17.java
Name:S Mohana Maheswari; Rollno:AV.SC.U4CSE24314 ;Section:CSE-A
Tiger makes moaning sound
Lion makes roar sound
```

IMPORTANT POINTS:

1. animal is an **abstract class** with an abstract method sound().
2. Lion and Tiger are **subclasses** that **override** the sound() method.
3. Lion's sound() method prints: *"Lion makes roar sound"*.
4. Tiger's sound() method prints: *"Tiger makes moaning sound"*.
5. The mainanimal_17 class demonstrates **abstraction** and **method overriding** in Java.

2.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:



CODE:

```
abstract class shape3D{
   abstract void calculateVolume();
   abstract void calculateSurfaceArea();
}
class  sphere extends shape3D{
      double radius;
        sphere(int radius){
            this.radius=radius;
}
 public void calculateVolume(){
    double volumeofsphere=(4.0/3.0)*3.14*radius*radius*radius;
       System.out.println("volume of sphere:"+volumeofsphere);
}
 public void  calculateSurfaceArea(){
   double surfaceareaofsphere=4*3.14*radius*radius;
       System.out.println("surface area of sphere:"+surfaceareaofsphere);
}}
class  Cube extends shape3D{
    int side;
       Cube(int side){
         this.side=side;
}
  public void calculateVolume(){
          int volumeofCube=side*side*side;
              System.out.println("volume of cube:"+volumeofCube);
}
 public void  calculateSurfaceArea(){
   int surfaceareaofcube=6*side*side;
```

```
      System.out.println("surface area of cube:"+surfaceareaofcube);
}}
class mainShape3D{
  public static void main(String[] args) {
System.out.println("Name:S Mohana Maheswari"+"; "+"Rollno:AV.SC.U4CSE24314"+"
;"+"Section:CSE-A");
  Cube obj1=new Cube(2);
  sphere obj2=new sphere(4);
obj1.calculateSurfaceArea();
obj2.calculateSurfaceArea();
obj1.calculateVolume();
obj2.calculateVolume();
}}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac   mainShape3D.java
mainShape3D.java:41: error: cannot find symbol
  Cube obj1=new cube(2);
               ^
  symbol:   class cube
  location: class mainShape3D
1 error
```

ERROR:

Code Error	**Code rectification**
1. cannot find symbol	1. our class name is Cube (with capital C) SO we shouls use capital “C” instead of small “c”.

OUTPUT:

```
C:\Users\smoha\Java Prgms>javac   mainShape3D.java

C:\Users\smoha\Java Prgms>java  mainShape3D
Name:S Mohana Maheswari; Rollno:AV.SC.U4CSE24314 ;Section:CSE-A
surface area of cube:24
surface area of sphere:200.96
volume of cube:8
volume of sphere:267.94666666666666
```

IMPORTANT POINTS:

shape3D is an **abstract class** with two abstract methods: calculateVolume() and calculateSurfaceArea().

sphere and Cube are **concrete subclasses** that extend shape3D and **implement the abstract methods**.

sphere uses the formulas:

- Volume: 43πr3\frac{4}{3} \pi r^334πr3
- Surface Area: 4πr24 \pi r^24πr2

Cube uses the formulas:

- Volume: side3side^3side3
- Surface Area: 6×side26 \times side^26×side2

The mainShape3D class demonstrates **polymorphism and abstraction** by calling methods on different shapes.

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 n) and a concrete method to display the pattern title

• Implement two subclasses

1. Star pattern prints a right angle triangle of Star(*)

2. Numberpattern-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.

Expected output:

Pattern 1:

*

* *

* * *

* * * *

* * * * *

Pattern 2:

1

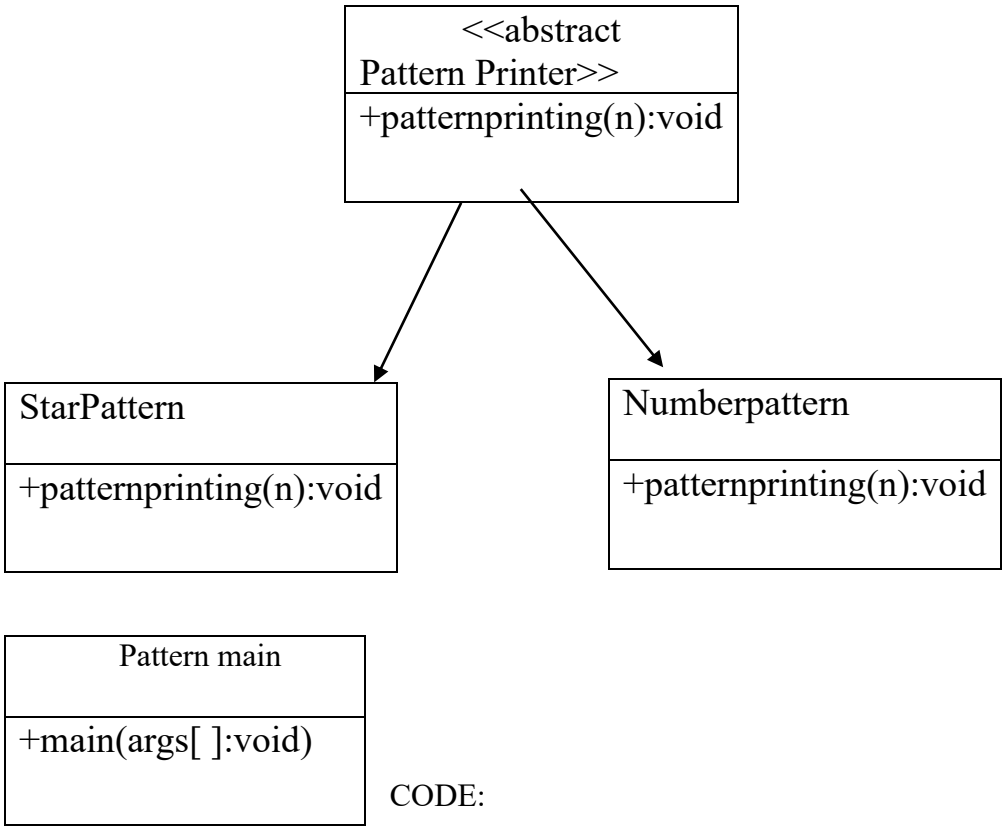
1 2

1 2 3

1 2 3 4

1 2 3 4 5

CLASS DIAGRAM:



CODE:

```
abstract class Patternprinter
{
   abstract  void patternprinting(int n);
}
class StarPattern extends Patternprinter
{
void patternprinting(int n)
{
   System.out.println("Star pattern");

   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 patternprinting(int n)
   {
      System.out.println("Number pattern");
      for(int i=1;i<=n;i++)
      {
         for(int j=1;j<=i;j++)
         {
         System.out.print(j);
         }
```

```
            System.out.println();
        }}}
public class patternmain
{
    public static void main (String args[])
    {
      StarPattern S=new StarPattern();
      Numberpattern N=new Numberpattern();
      S.patternprinting(5);
      N.patternprinting(5);
}}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  patternmain.java
patternmain.java:52: error: reached end of file while parsing
}
 ^
1 error
```

```
C:\Users\smoha\Java Prgms>javac  patternmain.java

C:\Users\smoha\Java Prgms>java  patternmain
Star pattern
*
**
***
****
*****
Number pattern
1
12
123
1234
12345
```

ERROR:

1. error: reached end of file while parsing }.	1.we must close the curly braces.

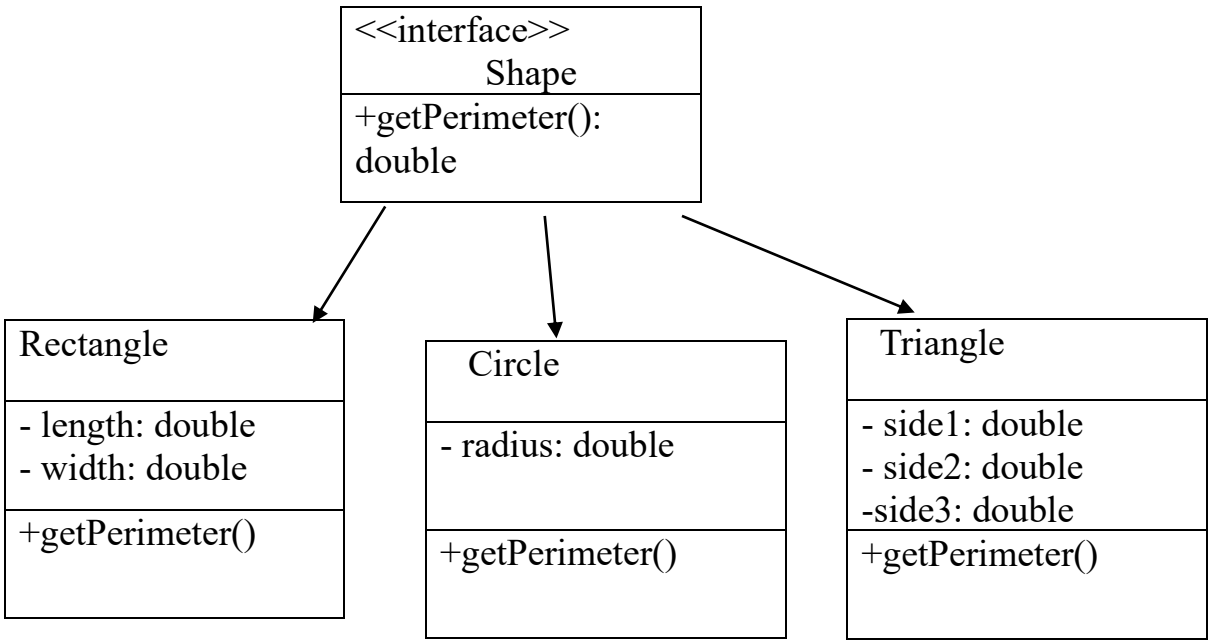
IMPORTANT POINTS:

1. **Abstract Class Usage**
    - Patternprinter is an **abstract class** that defines a blueprint method patternprinting(int n) which must be implemented by subclasses.
2. **Inheritance and Polymorphism**
    - StarPattern and Numberpattern are **concrete subclasses** that inherit from Patternprinter.
    - Each subclass **overrides** the patternprinting method to provide its own implementation.
3. **Method Overriding**
    - Both subclasses implement the same method name patternprinting(int n) but with different logic: one prints stars, the other prints numbers.
4. **Pattern Logic**
    - **Star Pattern** prints a right-angled triangle of asterisks (*).
    - **Number Pattern** prints a right-angled triangle with increasing numbers from 1 to i.
5. **Main Method**
    - patternmain creates objects of StarPattern and Numberpattern.
    - It calls their respective patternprinting() methods to display both patterns.

WEEK-08

1)Write a Java program to create an interface Shape with the getPerimeter() method. Create three getPerimeter() method for each of the three classes.

CLASS DIAGRAM:



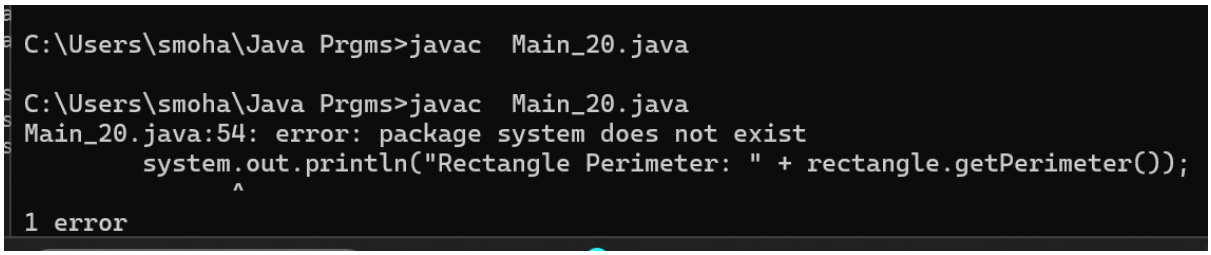
CODE:

```
interface Shape {

  double getPerimeter();

}

class Rectangle implements Shape {

    private double length;

    private double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    public double getPerimeter() {
```

```
        return 2 * (length + width);
    }}
class Circle implements Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double getPerimeter() {
        return 2 * Math.PI * radius;
    }}
class Triangle implements Shape {
    private double side1, side2, side3;
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }
    public double getPerimeter() {
        return side1 + side2 + side3;
    }}
public class Main_20 {
    public static void main(String[] args) {
 System.out.println("Name: S Mohana Maheswari; Roll No: AV.SC.U4CSE24314; Section: CSE-A");
        Shape rectangle = new Rectangle(5, 10);
        Shape circle = new Circle(8);
```

```
        Shape triangle = new Triangle(4, 5, 6);


        system.out.println("Rectangle Perimeter: " + rectangle.getPerimeter());
        System.out.println("Circle Perimeter: " + circle.getPerimeter());
        System.out.println("Triangle Perimeter: " + triangle.getPerimeter());
    }
}
```

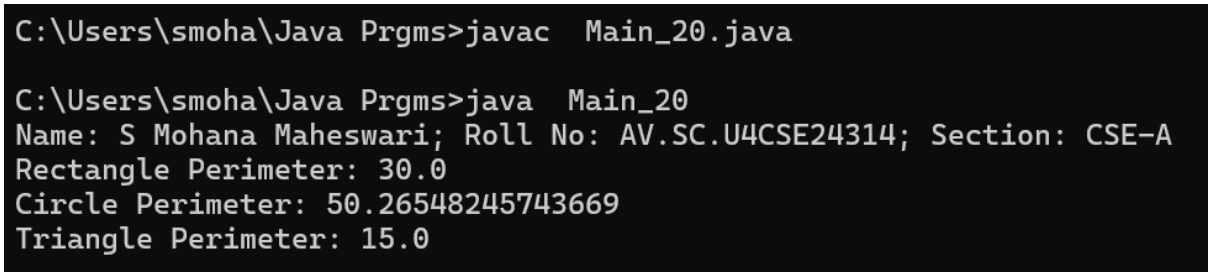
OUTPUT:



ERROR:

Code Error	Code rectification
1. package system does not exist	1. in system “S” should be capital S.

OUTPUT:



IMPORTANT POINTS:

Interface Shape:Declares the method getPerimeter(), which must be implemented by any class that implements this interface.

Three Shape Classes:Rectangle, Circle, and Triangle all implement Shape and provide their own formula for calculating the perimeter.

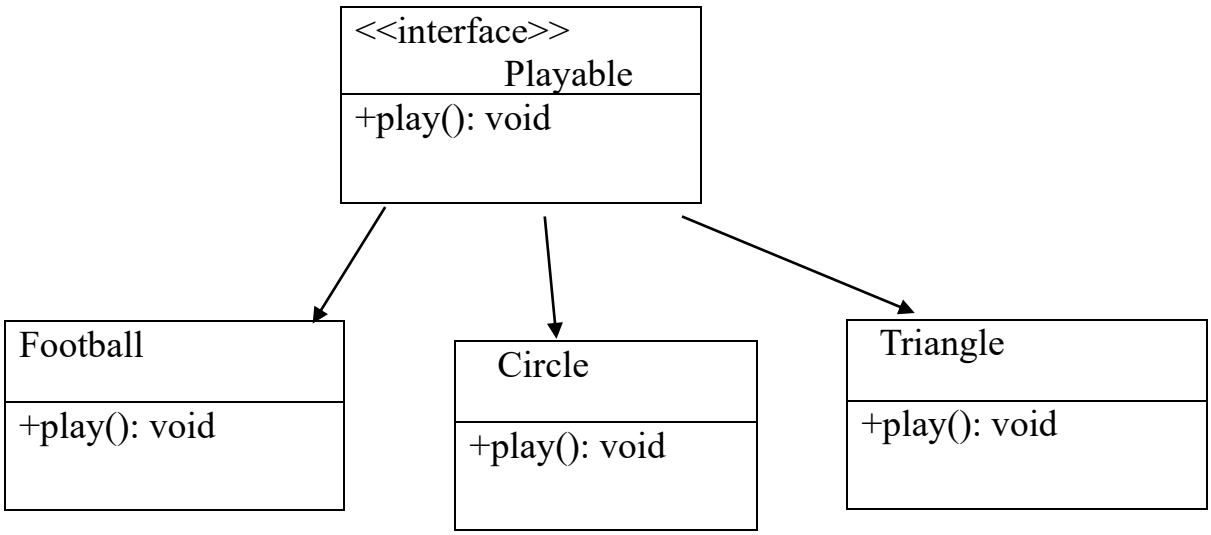
Polymorphism:A Shape reference is used to store different types of shape objects, showing the concept of **runtime polymorphism**.

Object-Oriented Principles:The code uses **encapsulation** (private variables) and **constructors** to initialize object values.

Output Display and Typo:Student info is printed first.There's a small **typo**: system.out.println should be System.out.println.

2)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 {
   void play();
}
class Football implements Playable {
   @Override
   public void play() {
      System.out.println("Playing Football: Kicking the ball towards the goal");
   }}
class Volleyball implements Playable {
   @Override
   public  play() {
```

```
        System.out.println("Playing Volleyball: Bumping, setting, and spiking the ball");
    }}
class Basketball implements Playable {
    @Override
    public void play() {
        System.out.println("Playing Basketball: Dribbling and shooting the ball");
    }}
public class Main_25{
    public static void main(String[] args) {
        System.out.println("Name: S Mohana Maheswari; Roll No: AV.SC.U4CSE24314;
Section: CSE-A");
        Playable football = new Football();
        Playable volleyball = new Volleyball();
        Playable basketball = new Basketball();
        football.play();
        volleyball.play();
        basketball.play();
    }}
```

OUTPUT:

```
C:\Users\smoha\Java Prgms>

C:\Users\smoha\Java Prgms>javac  Main_25.java
Main_25.java:14: error: invalid method declaration; return type required
    public  play() {
            ^
1 error
```

ERROR:

Code Error	Code rectification
1. error: invalid method declaration; return type required	1. mention void return type.

```
C:\Users\smoha\Java Prgms>javac  Main_25.java

C:\Users\smoha\Java Prgms>java  Main_25
Name: S Mohana Maheswari; Roll No: AV.SC.U4CSE24314; Section: CSE-A
Playing Football: Kicking the ball towards the goal
Playing Volleyball: Bumping, setting, and spiking the ball
Playing Basketball: Dribbling and shooting the ball
```

IMPORTANT POINTS:

1. The playable interface abstracts the play() method, ensuring different classes implement it differently
2. The play() method behaves differently based on the object type football, volleyball, basketball.

Each class encapsulates its own implementation of how the sport is played, hiding the details from the user.

3)Write a java program to implement a login system using intefaces.

CODE:

```
import java.util.Scanner;


interface LoginSystem {
    void login();
}


class UserLogin implements LoginSystem {
    String correctUsername = "amrita_student";
    String correctPassword = "amrita@123";


    public void login() {
        Scanner sc = new Scanner(System.in);


        System.out.print("Enter Amrita Portal Username: ");
        String username = sc.nextLine();
```

```
        System.out.print("Enter Password: ");
        String password = sc.nextLine();

        if (username.equals(correctUsername) && password.equals(correctPassword)) {
            System.out.println("Login successful! Welcome to Amrita University Portal, " +
username + "!");
        } else {
            System.out.println("Login failed! Incorrect Amrita credentials.");
        }
    }
}

public class Mainlab3 {
    public static void main(String[] args) {
        LoginSystem user = new UserLogin();
        user.login();
    }
}
```

ERROR:

Code Error	Code rectification
1. error: invalid method declaration; return type required	1. mention void return type.

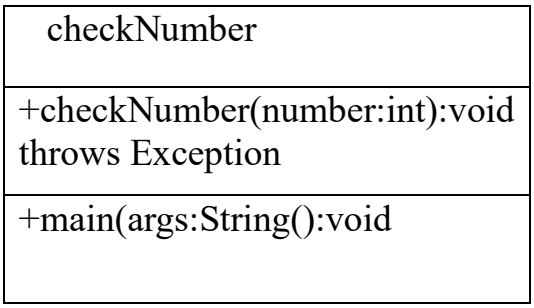
OUTPUT:

```
C:\Users\smoha\Java Prgms>javac  Mainlab3.java

C:\Users\smoha\Java Prgms>java  Mainlab3
Enter Amrita Portal Username: AV.SC.U4CSE24314
Enter Password: Mohana@104
Login failed! Incorrect Amrita credentials.
```

WEEK-09

1. **Write a java program to create a method that take integer as parameter and throws an example if the number is even.**

Class Diagram:



Code:-

```
public class checkNumber {

public static void checkNumber(int number) throws Exception {

if (number % 2 == 0) {
throw new Exception("Even number is not allowed: " + number);
} else {
System.out.println("Valid output number: " + number);
}}
public static void main(String[] args) {
try {
checkNumber(9);
} catch (Exception e) {
System.out.println("Exception caught: " + e.getMessage());
}}}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  checkNumber.java

C:\Users\smoha\Java Prgms>java  checkNumber
Valid output number: 9
```

Error Table:-

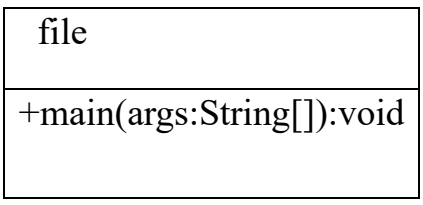
S.no	Expected Error	Error rectification
1	Setting the parameters inside the constructor	We cannot pass the values inside constructor without setting them first
2	}	Ending the class and main method is required

IMPORTANT PONTS:

- **Method Declaration**
  checkNumber(int number) is declared with throws Exception, meaning it can throw an exception that must be handled by the caller.
- 🔹 **Validation Logic**
  The method checks if the number is even (number % 2 == 0). If yes, it throws an exception.
- 🔹 **Use of throw**
  Uses throw new Exception(...) to manually throw an exception when the input is invalid (even number).
- 🔹 **Exception Message**
  The exception includes a custom message: "Even number is not allowed: " + number.

2.Write a java program to create a method that reads a file and throws an exception if the file is not found .

Class Diagram:-



Code:-

```
import java.io.BufferedReader;
 import java.io.FileReader;
 import java.io.IOException;
 public class file {
    public static void main(String[] args) {
       BufferedReader br = null; // Declare BufferedReader outside the try block
       try {
          br = new BufferedReader(new FileReader("E:/Amrita/example.txt"));
          String line; // Corrected 'string' to 'String'
          while ((line = br.readLine()) != null) {
             System.out.println(line); // Print the actual line instead of the string
"line"
          }
       } catch (IOException e) { // Catch IOException directly
          System.out.println("An error occurred while reading the file: " +
e.getMessage());
       } finally {
          if (br != null) {
             try {
                br.close();
             } catch (IOException e) {
                System.out.println("Error closing the BufferedReader: " +
e.getMessage());
             }}}}}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  file.java

C:\Users\smoha\Java Prgms>java  file
An error occurred while reading the file: E:\Amrita\example.txt
(The system cannot find the path specified)
```

Error Table:-

S.no	Expected Error	Error rectification
1	Setting the parameters inside the constructor	We cannot pass the values inside constructor without setting them first
2	}	Ending the class and main method is required

IMPORTANT PONTS:

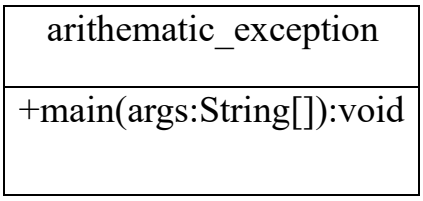
Method Declaration
checkNumber(int number) is declared with throws Exception, meaning it can throw an exception that must be handled by the caller.

Validation Logic
The method checks if the number is even (number % 2 == 0). If yes, it throws an exception.

Try-Catch Block
In the main method, a try-catch block is used to catch the exception thrown by checkNumber.

3.Write a java program to handle arithematic exception using try catch and finally .

Class Diagram:



Code:-

```
import java.util.Scanner;

public class arithematic_exception {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);

try {
System.out.println("Enter first number (numerator):");
int a = input.nextInt();

System.out.println("Enter second number (denominator):");
```

```
int b = input.nextInt();


int result = a / b;
System.out.println("Result of division: " + result);
} catch (ArithmeticException e) {
System.out.println("Error: Cannot divide by zero.");
} catch (Exception e) {
System.out.println("Error: " + e.getMessage());
} finally {
input.close();
}
}
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac   ArithmeticExceptionExample.java

C:\Users\smoha\Java Prgms>java   ArithmeticExceptionExample
Enter first number (numerator):
8
Enter second number (denominator):
2
Result of division: 4
```

Error Table:-

S.no	Expected Error	Error rectification
1	Setting the parameters inside the constructor	We cannot pass the values inside constructor without setting them first
2	}	Ending the class and main method is required

IMPORTANT PONTS:

- **Class Purpose**
  The program demonstrates handling of arithmetic exceptions during division.

- **User Input with Scanner**
  Scanner is used to take two integer inputs from the user: numerator and denominator.

- **ArithmeticException Handling**
  If the denominator is zero, ArithmeticException is caught and a specific error message is shown.

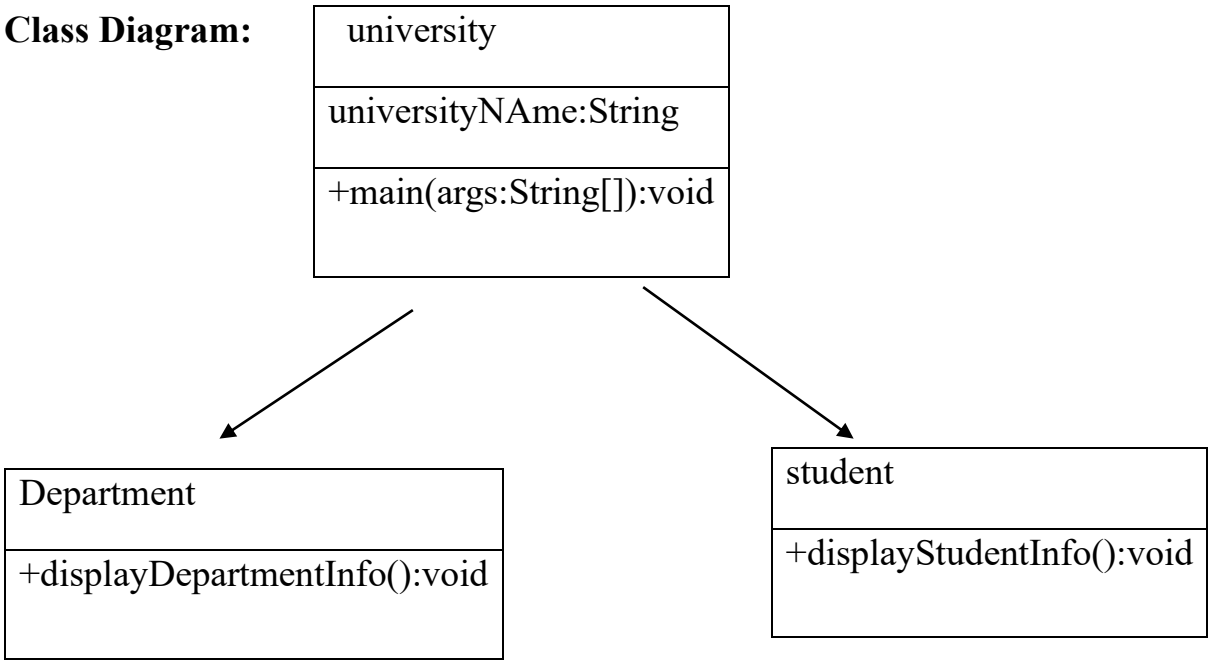
- **Generic Exception Catch**
  A general Exception block is included to catch any unexpected errors (like invalid input).

4. Write a java program to simulate a university system using inner classes

- ✓ Create an outer class namedd University with a variable UniversityName
- ✓ Inside it defgine two non-static in classes

1. Department-With variable like deptName and deptCode and a method to display department details.
2. Student-Variable like stdName and stdCode and a method to display Student details.
3. Create an object for each class and call their methods to display their details and with the university name.

Class Diagram:



Code:-

```
public class university {

    String universityName = "Amrita University";

    class Department {

        String deptName = "computer science";
        int deptcode = 101;

        void displayDepartmentInfo() {
            System.out.println("department" + deptName);
            System.out.println("department" + deptcode);
        }
    }

    class student {

        String stdname = "Sai Krishna";
        int stdcode = 18977;

        void displayStudentInfo() {
            System.out.println("department" + stdname);
            System.out.println("department" + stdcode);

        }
    }

    public static void main(String[] args) {
        university uni = new university();
        System.err.println("University" + uni.universityName);
        System.err.println("Department__Info");
        university.Department dept = uni.new Department();
        dept.displayDepartmentInfo();

        System.out.println("***** Student Info *****");
        university.student stdent = uni.new student();
        stdent.displayStudentInfo();
    }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  university.java

C:\Users\smoha\Java Prgms>java  university
UniversityAmrita University
Department__Info
departmentcomputer science
department101
***** Student Info *****
departmentS Mohana Maheswari
department18777
```

Error Table:

S.no	Expected Error	Error rectification
1	Setting the parameters inside the constructor	We cannot pass the values inside constructor without setting them first
2	}	Ending the class and main method is required

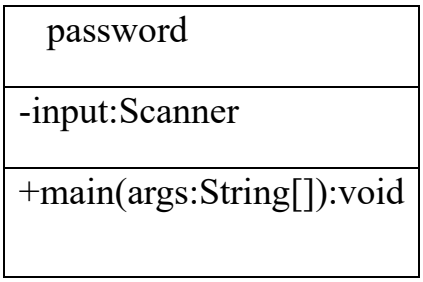
IMPORTANT PONTS:

- Outer Class Definition
  The class university represents a university and contains inner classes for departments and students.
- **Member Variable in Outer Class**
  It has a String variable universityName initialized with "Amrita University".
- **Non-Static Inner Classes**
  Two non-static inner classes are defined: Department and student, each with its own attributes and methods.
- **Department Inner Class**
  Department has deptName and deptcode, and a method displayDepartmentInfo() to print them.
- **Student Inner Class**
  student has stdname and stdcode, and a method displayStudentInfo() to print them.

WEEK-10

1. Write a java program to generate a password for a student using his/her intials and age. The password displayed should be the string consists of first character of first name, middle name last name with age.

Class Diagram:



Code:-

```
import java.lang.String;
import java.util.Scanner;
public class password {
   public static void main(String[] args) {

     Scanner input=new Scanner(System.in);

     System.out.println("enter the first name");
     String FN=input.next();
     System.out.println("enter the last name");
     String LN=input.next();
     System.out.println("enter the age");
     int AGE=input.nextInt();
     String initials=FN.substring(0,1)+LN.substring(0,1)+AGE;
     String PIN=initials.toLowerCase();
    System.out.println(initials+":- is the password created");
}
    }
```

Output:

```
C:\Users\smoha\Java Prgms>javac  password.java

C:\Users\smoha\Java Prgms>java  password
enter the first name
Mohana
enter the last name
Maheswari
enter the age
18
MM18:- is the password created
```

Error Table:-

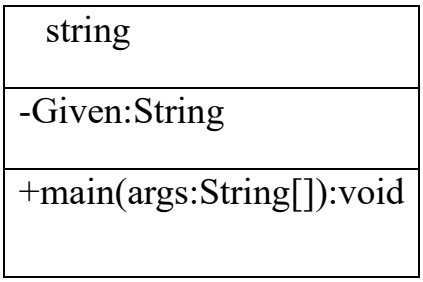
S.no	Expected Error	Error rectification
1	}	Ending the class and main method is required

IMPORTANT PONTS:

- Scanner for Input
  The program uses Scanner to take user input for first name, last name, and age.
- String Manipulation
  Uses substring(0,1) to extract the first character from both the first and last names.
- Password Creation Logic
  Concatenates the initials of the names with the age to form a password.
- Case Formatting
  Converts the generated password to lowercase using toLowerCase() and stores it in PIN.

2. Design and implement a java program that will do the following questions to this string "Welcome! You are practicing Strings concept".
i) Convert all the alphabets to capital letters and print out the result
ii) Convert all alphabets to lower-case letters and print out the result
iii) print out the length of the string
iv) Print out the index of the concept.

Class Diagram:



Code:-

```
import java.lang.String;
public class string{
   public static void main(String[] args) {
      String Given="welcome! You are practicing strings concept";
      System.out.println("Converting into upper case letters :
"+Given.toUpperCase());
      System.out.println("Converting into lower case letters :
"+Given.toLowerCase());
      System.out.println("Resulting the length of the string : "+Given.length());
      System.out.println("Finding the index of the given String
:"+Given.indexOf("concept"));
   }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  string.java

C:\Users\smoha\Java Prgms>java  string
Converting into upper case letters : WELCOME! YOU ARE PRACTICING STRINGS CONCEPT
Converting into lower case letters : welcome! you are practicing strings concept
Resulting the length of the string : 43
Finding the index of the given String :36
```

Error:

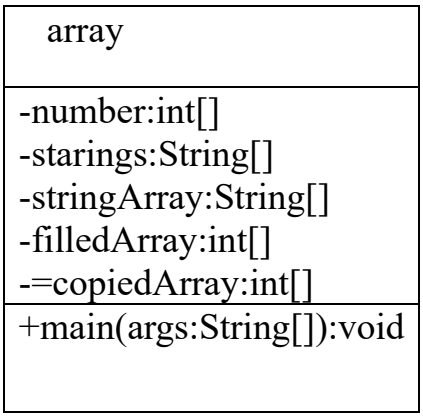
Code Error	error rectification
1. error: ';' expected	1. we must end line with semicolon.

IMPORTANT PONTS:

- String Declaration
  A String variable named Given is initialized with a sentence for manipulation.
- Uppercase Conversion
  toUpperCase() is used to convert the entire string to uppercase letters.
- Lowercase Conversion
  toLowerCase() converts the entire string to lowercase letters.
- Finding String Length
  length() method returns the number of characters in the string.
- Finding Substring Index
  indexOf("concept") finds and returns the starting index of the word "concept".
-

3.Implement a java program using the below array methods
i) Sorting the elements (numbers and strings) of an array
ii) Convert the array elements into string
iii) Fill the part of an array
iv) Copy the elements of one array into another.

Class Diagram:



Code:-

```
import java.util.Arrays;
public class array {
   public static void main(String[] args) {
      // 1. Sorting the elements (numbers)
      int[] numbers = {5, 3, 8, 1, 2};
      System.out.println("Original numbers array: " +
Arrays.toString(numbers));
      Arrays.sort(numbers);
      System.out.println("Sorted numbers array: " + Arrays.toString(numbers));
      // 1. Sorting the elements (strings)
      String[] strings = {"Banana", "Apple", "Orange", "Mango"};
      System.out.println("Original strings array: " + Arrays.toString(strings));
      Arrays.sort(strings);
      System.out.println("Sorted strings array: " + Arrays.toString(strings));
      // 2. Convert the array elements into strings
      String[] stringArray = Arrays.stream(numbers)
                           .mapToObj(String::valueOf)
                           .toArray(String[]::new);
      System.out.println("Converted numbers array to strings: " +
Arrays.toString(stringArray));
      // 3. Fill part of an array
      int[] filledArray = new int[10];
      Arrays.fill(filledArray, 0, 5, 7); // Fill first 5 elements with 7
      System.out.println("Array after filling part of it: " +
Arrays.toString(filledArray));
      // 4. Copy the elements of one array into another
      int[] copiedArray = new int[numbers.length];
      System.arraycopy(numbers, 0, copiedArray, 0, numbers.length);
      System.out.println("Copied array: " + Arrays.toString(copiedArray));
      // Close the scanner
   }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac   array.java

C:\Users\smoha\Java Prgms>java   array
Original numbers array: [5, 3, 8, 1, 2]
Sorted numbers array: [1, 2, 3, 5, 8]
Original strings array: [Banana, Apple, Orange, Mango]
Sorted strings array: [Apple, Banana, Mango, Orange]
Converted numbers array to strings: [1, 2, 3, 5, 8]
Array after filling part of it: [7, 7, 7, 7, 7, 0, 0, 0, 0, 0]
Copied array: [1, 2, 3, 5, 8]
```

Error:

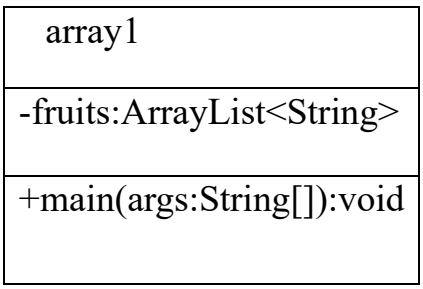
Code Error	**error rectification**
1. error: ';' expected	1. we must end line with semicolon.

IMPORTANT PONTS:

- Array Sorting (Integers)
  Arrays.sort(numbers) sorts the integer array in ascending order.
- Array Sorting (Strings)
  Arrays.sort(strings) arranges string elements alphabetically (lexicographically).
- Array to String Conversion
  Uses Arrays.stream(numbers).mapToObj(String::valueOf) to convert an integer array to a string array.
- Using Arrays.toString()
  Arrays.toString(array) is used throughout to print array contents in readable format.
- Demonstrates Java Utility Methods
  Showcases various utility methods provided by the java.util.Arrays class.

4.Implement a java program using the below Array list
i) Insert an element at particular index in the array list
ii) Modify an element in the array list
iii)access an element from the array list
iv) remove an element from the array list.

Class Diagram:



Code:-

```
import java.util.ArrayList;
public class array1{
   public static void main(String[] args) {
       ArrayList<String> fruits = new ArrayList<>();
      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Orange");
      System.out.println("Original ArrayList: " + fruits);
      fruits.add(1, "Mango");
      System.out.println("After inserting 'Mango' at index 1: " + fruits);
      fruits.set(2, "Grapes");
      System.out.println("After modifying element at index 2: " + fruits);
      String fruitAtIndex3 = fruits.get(3);
      System.out.println("Element at index 3: " + fruitAtIndex3);
      fruits.remove("Banana");
      System.out.println("After removing 'Banana': " + fruits);
      fruits.remove(0);
      System.out.println("After removing element at index 0: " + fruits);
   }
}
```

Output:

```
C:\Users\smoha\Java Prgms>javac  array1.java

C:\Users\smoha\Java Prgms>java  array1
Original ArrayList: [Apple, Banana, Orange]
After inserting 'Mango' at index 1: [Apple, Mango, Banana, Orange]
After modifying element at index 2: [Apple, Mango, Grapes, Orange]
Element at index 3: Orange
After removing 'Banana': [Apple, Mango, Grapes, Orange]
After removing element at index 0: [Mango, Grapes, Orange]
```

Error:

Code Error	**error rectification**
1. error: ';' expected	1. we must end line with semicolon.

IMPORTANT PONTS:

- ArrayListDeclaration
  An ArrayList<String> named fruits is created to store a list of fruit names.
- **AddingElements**
  fruits.add(...) is used to add elements to the ArrayList; "Apple", "Banana", "Orange" are added initially.
- **InsertingatIndex**
  fruits.add(1, "Mango") inserts "Mango" at index 1, shifting other elements.
- **Dynamic List Behavior**
  The ArrayList resizes automatically as elements are added or removed.
- **No User Input**
  All operations are hardcoded; no input is taken from the user.
- **Demonstrates Core List Operations**
  The program demonstrates key ArrayList methods: add(), set(), get(), and remove().