23CSE111

OBJECT ORIENTED PROGRAMMING

LAB REPORT

Department of Computer Science Engineering

Amrita School of Computing

Amrita Vishwa Vidhyapeetam, Amaravati Campus

Name:M.AKSHAYA

Verified By **Roll No: 24218**

INDEX

	PROGRAM	REMARKS
	1. Downloading and installation of java.	
WEEK-1	2.Write a java program that prints name, roll no, section of a student.	
	1. Write a java program to find the simple interest where all the inputs are taken from the user.	
	2.Write a java program to find the fibonacci sequence of a given number.	
WEEK-2	3.Write a java program to find the area of rectangle.	
	4.write a java program to find the area of triangle.	
	5.write a java code to convert the temperature from celsius to fahrenheit and from fahrenheit to celsius.	

	PROGRAM	REMARKS
WEEK-3	1.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 methods named car1, car2, car3.	
	2.To create a class bankaccount with methods deposit() and withdrawal.	
WEEK-4	1.Write a java program with class named “book”. the class should contain various attributes such as “title of the book”, “author”, year of publications”, it should also contain a constructor with parameter which initializes “title of the book”, “author”, “year of publication”. create a method which displays the details of the book. title of the book (), author (), year of publication ().display the details of two book, by creating 2 objects.	

	PROGRAM	REMARKS
	2.TO create a java program with class named “myclass” with a static variable count of “int” type, in sized to “zero” and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class. define a contractor for “myclass” is created finally print the final values “count” and “pi” variables.	
WEEK-5	1.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.	

	PROGRAM	REMARKS
	2.A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed(should be in super class) 1. cars should have an additional property: no.of doors 2. Bikes should have a property indicating whether they have gears or not. 3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting. 4. Every class should have a constructor Question: 1. Which oops concept is used in the above program 2. If the company decides to add a new type of vehicle, Truck, how would you modify the program? a. Truck should include an additional property capacity (in tons) b. Create a showTruckdetails() method to display the truck's capacity. c. Write a constructor for Truck that initializes all properties 3. Implement the truck class and	

	PROGRAM	REMARKS
WEEK-6	1.Write a java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information a About Car.	
	2.A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification. 1. UG admission require min of 60% 2. PG admission require min of 70%	
	3.To create a Java Program with class named “my class” with a Static Variable Count int type and initialize to 0 and A Constant Variable "pi" of type double initialized to 3.1415 has attributes of that class. Now defi a Constructor for my class that increments the Count Variable each time an object of my class is created. Finaly Print the final values of count.	

	PROGRAM	REMARKS
	4.Write a Java Program and create a Shape class with a method calcArea(). That is overloaded for different shapes like square and rectangle. Create a sub class circle that overrides the calcArea() for a circle.	
WEEK-7	1.Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal.	
	2.Write a java program to create an abstract class Shape3D with abstract methods calculatevolume() and calculate surfacearea().Create the subclasses sphere and cube that extend the class Shape3D and implement the respective methods to calculate the volume ans surface area of each shape.	

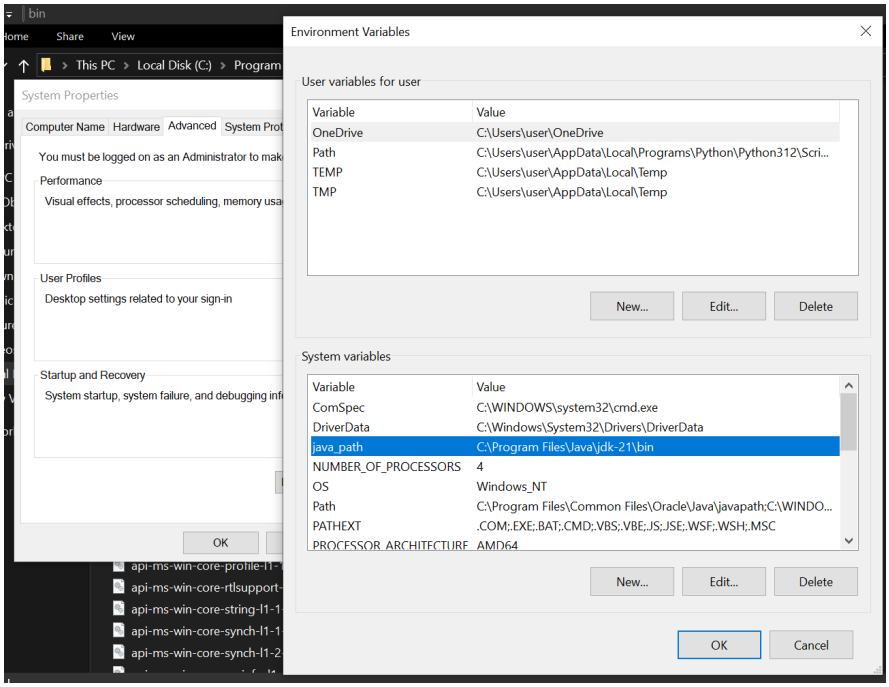
	PROGRAM	REMARKS
	3.Write a java program using abstract class to define a method for pattern printing. • Create an abstract class named PatternPrinter with an abstract method printPattern(int) and a concrete method to display the pattern title. • Implement two subclasses: 1. Starpattern- prints a right-angled triangle of star (*).	
WEEK-8	1.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	
	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.	

WEEK-1

1) Explain the process of Installing JDK (Java Development Kit)

Installing of JDK (Java Development Kit):

1. **Download JDK:**
    - Go to the Oracle JDK download page in your web browser and click on JDK-21 version which is Long term support (LTS) version.
    - Click on the download link for your operating system (Windows, macOS, or Linux).
2. **Install JDK:**
    - Once downloaded, run the installer.
    - Follow the instructions and keep clicking "Next" until it's done.
3. **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 **New** under **System Variables**:
        - **Set Variable name as:** java_home
        - **Variable value:** The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)
    - Find Path under **System Variables**, click **Edit**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)
    -



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.

```
Command Prompt
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>javac --version
javac 21.0.5

C:\Users\user>java --version
java 21.0.5 2024-10-15 LTS
Java(TM) SE Runtime Environment (build 21.0.5+9-LTS-239)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.5+9-LTS-239, mixed mode, sharing)

C:\Users\user>
```

Program-1:

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

Print helloworld:-

```
public class helloworld {
  public static void main(String[] args) {
   System.out.println("Hello World");
  }
}
```

Output:

```
m.akshaya@Akshaya-MacBook-Air MyProjects % javac Helloworld.java
m.akshaya@Akshaya-MacBook-Air MyProjects % java Helloworld
Hello World
m.akshaya@Akshaya-MacBook-Air MyProjects %
```

ERRORS:

None found

Program-2:

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

STUDENT DETAILS:

```
public class studentinformation {
  public static void main(String [] args) {
    System.out.println("NAME:M.AKSHAYA");
    System.out.println("Section:C");
   System.out.println("Roll no:AV.SC.U4CSE24218");
  }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects % javac Studentinformation.java
m.akshaya@Akshaya-MacBook-Air MyProjects % java Studentinformation
NAME:M.AKSHAYA
Section:C
Roll no:AV.SC.U4CSE24218
m.akshaya@Akshaya-MacBook-Air MyProjects %
```

IMPORTANT POINTS:

1. When printing the statements, everything should be inside double quotes.

ERROR TABLE:

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

WEEK-2:

Program-1:

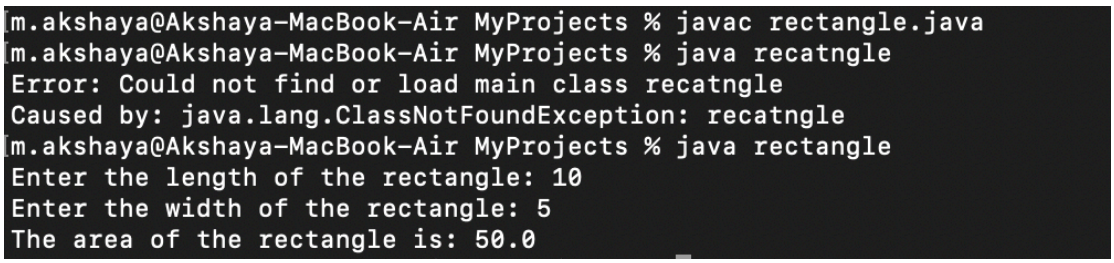
AIM: Write a java program to Calculate area of rectangle.

CODE:

```
 import.java.util.scanner;
Public class rectangle{
    Public static void main(String[]args){
        Scanner scanner=new Scanner(system.in);
        System.out.println(“enter length”);
        Double length=scanner.new double();
        System.out.println(“enter the width”)
       Double width=scanner.new double();
       Double area=length*width
       System.out.println(“area”+area);
       scanner.close();
```

```
  }
}
```

OUTPUT:



ERROR TABLE:

Code Error	**Code rectification**
1. While using for iteration, not giving the conditions correctly. 2. Declaring the data type as double instead of int. 3. Writing small	1. We should give iterative statements correctly. 2. We should give the data type as int for integers.

IMPORTANT POINTS:

1.Area of a rectangle is area = l*b, where
L = length of a side of the rectangle,
B= breadth of a side of the rectangle.
2.Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly

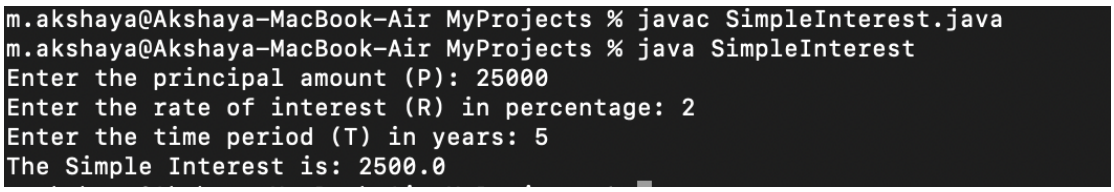
Program-2:

AIM: Write a java program to Calculate the simple interest by input given by the user.

CODE:

```
Import java.util.Scanner;
  public class SimpleInterest {
    public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the principal amount (P): ");
      double principal = scanner.nextDouble();
      System.out.print("Enter the rate of interest (R) in
percentage: ");
      double rate = scanner.nextDouble();
      System.out.print("Enter the time period (T) in years: ");
      double time = scanner.nextDouble();
     double simpleInterest = (principal * rate * time) / 100;
     System.out.println("The Simple Interest is: " +
simpleInterest);
     scanner.close();
   }
}
```

OUTPUT:



ERROR TABLE:

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

IMPORTANT POINTS:

1.Simple interest formula is: (p*t*r)/100, where:
P: Principal amount
R: Rate of interest
T: Time period
2.The data type double indicates the floating points in the integers.
3.The line “import java.util.Scanner” indicates:
Import: tells the java compiler that we want to use a specific class or package in your code.
Java.util : This is the package that contains utility classes for Java programming, including the “Scanner” class.
Scanner: this is the class that allows you to read input from the keyboard.

Program-3:

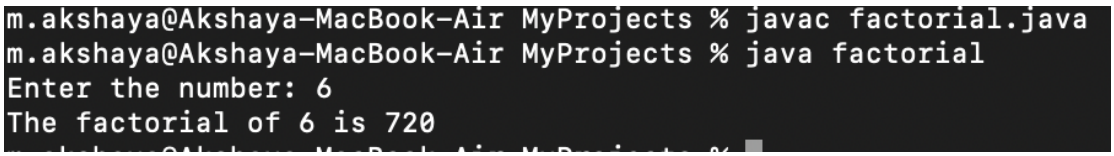
AIM: Write a java program to calculate the Factorial of N(given by the user).

CODE:

```
import java.util.Scanner;
 public class factorial {
  public static void main(String[] args) {
     Scanner scanner = new Scanner(System.in);
     System.out.print("Enter the number: ");
     int number = scanner.nextInt();
     long factorial = 1;
     if (number < 0) {
        System.out.println("No Factorial For Negative Numbers.");
     } else {
        for (int i = 1; i <= number; i++) {
           factorial *=i;
```

```
}
        System.out.println("The factorial of " + number + " is " +
factorial);
      }
     scanner.close();
  } }
```

OUTPUT:



ERROR TABLE:

Code Error	Code rectification
3. While using for iteration, not giving the conditions correctly. 4. Declaring the data type as double instead of int.	3. We should give iterative statements correctly. 4. We should give the data type as int for integers.

IMPORTANT POINTS:

1.While the for loop the data inside the parenthesis indicates the Initial expression
Test expression and
Update expression.
2.Here “factorial*=I” means factorial = factorial*I.
3.Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points.

Program-4:

AIM: Write a java program to find the Fibonacci series (all inputs taken from the user).

CODE:

```
public class FibonacciSeries {
   public static void main(String[] args) {
      int n = 10;
      int firstTerm = 0, secondTerm = 1;
      System.out.println("Fibonacci Series up to " + n + "
terms:");
         for (int i = 1; i <= n; ++i) {
         System.out.print(firstTerm + ", ");
         int nextTerm = firstTerm + secondTerm;
         firstTerm = secondTerm;
         secondTerm = nextTerm;
      } } }
```

OUTPUT:

ERROR TABLE:

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

IMPORTANT POINTS:

1.In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.

2.This process is repeated a certain number of times until the conditions are met.

Program-5:

AIM: Write a java program to find the area of triangle using herons formula.

CODE:

```
import java.util.Scanner;
public class triangle{
  public static void main(String[] args){
    Scanner scanner=new Scanner(System.in);
    System.out.println("enter length a:");
    Double a = scanner.nextDouble();
    System.out.println("enter length b:");
    Double b = scanner.nextDouble();
    System.out.println("enter length c:");
    Double c = scanner.nextDouble();
    Double s= (a+b+c/2);
    Double area= Math.sqrt(s*(s-a)*(s-b)*(s-c));
    System.out.println("area of the triangle by heron formula"+
area);
    scanner.close();
  }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects % javac triangle.java
m.akshaya@Akshaya-MacBook-Air MyProjects % java triangle
enter length a:
2
enter length b:
4
enter length c:
6
area of the triangle by heron formula30.740852297878796
```

ERROR TABLE:

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

IMPORTANT POINTS:

1.Here, we're finding the area of a triangle using heron's formula.
2.Heron's formula for finding a triangle is:
S = (a +b +c)/2
Where S is the semi-perimeter of the triangle.
Now the area formula is:
Area = sqrt(s*(s-a)*(s-b)*(s-c)).

Program-6:

I) AIM: Write a java program to Convert temperature celsius into farenheit.

CODE:

```
import java.util.Scanner;
  public class celsiustofarenheit {
    public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Enter temperature in Celsius: ");
       float celsius = scanner.nextFloat();
       float farenheit = (celsius * 9 / 5) + 32;
       System.out.println(celsius + "°C is equal to " + farenheit
+"°F");
       scanner.close();
    }
```

}

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects % javac celsiustofarenheit.java
m.akshaya@Akshaya-MacBook-Air MyProjects % java celsiustofarenheit
Enter temperature in Celsius: 49
49.0°C is equal to 120.2°F
```

ERROR TABLE:

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

PROGRAM-2:

AIM: Write a java program to Convert temperature Fahrenheit to celsius.

CODE:

```
import java.util.Scanner;
public class fahrenheittocelsius {
   public static void main(String[] args) {
      int C, F;
      Scanner num = new Scanner(System.in);
      System.out.println("Enter the Fahrenheit temperature: ");
      F = num.nextInt();
      C = (F - 32) * 5 / 9;
      System.out.println("Celsius is: " + C);
      num.close(); // Close the Scanner to avoid resource leak
   }}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air ~ % javac farenheinttocelsius.java
m.akshaya@Akshaya-MacBook-Air ~ % java farenheinttocelsius
Enter the Fahrenheit temperature:
54
Celsius is: 12
```

ERROR TABLE:

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

IMPORTANT POINTS:

1.The formula to convert a Fahrenheit to Celsius is
Celsius = (Fahrenheit-32)*5/9
2.The formula to convert a Celsius to Fahrenheit is
Fahrenheit = (Celsius*9/5)+32.
3.The line “Scanner input = new Scanner(System.in),” tends to create a new Scanner object named “input” that reads input from the standard input stream (System.in), like keyboard.

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 these methods named start(),stop(),service()
4.Create the objects named car, car1,car2

CODE:

```
public class Car {
   private String car_color;
   private String car_brand;
   private String fuel_type;
   private String mileage;

   public void start() {
      System.out.println("car is started");
   }

   public void stop() {
      System.out.println("car is stopped");
   }

   public void service() {
      System.out.println("car is for service");
   }

   public static void main(String args[]) {
      Car car = new Car();
      car.car_color = "white";
      car.car_brand = "audi";
      car.fuel_type = "petrol";
      car.mileage = "20";
      car.start();
      System.out.println("car_color: " + car.car_color + "
car_brand: " + car.car_brand + " fuel_type: " + car.fuel_type + "
mileage: " + car.mileage);
```

```
        Car car1 = new Car();
        car1.car_color = "white";
        car1.car_brand = "audi";
        car1.fuel_type = "petrol";
        car1.mileage = "20";
        car1.stop();
        System.out.println("car_color: " + car1.car_color + "
car_brand: " + car1.car_brand + " fuel_type: " + car1.fuel_type
```

Book

- Title: String
- Author: String
- Year of Publication: int

+ Book(title:String, author:String,Year of publication:int)
+ DisplayDetails():void

```
+ " mileage: " + car1.mileage);
        Car car2 = new Car();
        car2.car_color = "white";
        car2.car_brand = "audi";
        car2.fuel_type = "petrol";
        car2.mileage = "20";
        car2.service();
        System.out.println("car_color: " + car2.car_color + "
car_brand: " + car2.car_brand + " fuel_type: " + car2.fuel_type
+ " mileage: " + car2.mileage);
    }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects % javac car.java
m.akshaya@Akshaya-MacBook-Air MyProjects % java car
car is started
car_color: white car_brand: audi fuel_type: petrol mileage: 20
car is stopped
car_color: white car_brand: audi fuel_type: petrol mileage: 20
car is for service
car_color: white car_brand: audi fuel_type: petrol mileage: 20
```

2.AIM: To create a class BankAccount with methods deposit() and withdraw() . create two subclasses savings account and checking account override the withdraw () method in each subclass to impose different withdrawal limits and fees.

PROGRAM:

```
import java.util.Scanner;

public class BankAccount {
   private double balance;
      public BankAccount(double initialBalance) {
      this.balance = initialBalance;
   }
  public void deposit(double amount) {
      balance += amount;
      System.out.println("Deposited: " + amount);
   }
  public void withdraw(double amount) {
      if (balance >= amount) {
         balance -= amount;
         System.out.println("Withdrawn: " + amount);
      } else {
         System.out.println("Insufficient balance.");
      }
```

```
    }
    public double getBalance() {
        return balance;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter initial balance: ");
        double initialBalance = scanner.nextDouble();
        BankAccount account = new BankAccount(initialBalance);

        System.out.print("Enter deposit amount: ");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);

        System.out.print("Enter withdrawal amount: ");
        double withdrawalAmount = scanner.nextDouble();
        account.withdraw(withdrawalAmount);

        System.out.println("Current balance: " +
account.getBalance());

        scanner.close();
    }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac BankAccount.java
m.akshaya@Akshaya-MacBook-Air MyProjects4 % java BankAccount
Enter initial balance: 100000
Enter deposit amount: 50,000
Deposited: 50000.0
Enter withdrawal amount: 90,000
Withdrawn: 90000.0
Current balance: 60000.0
```

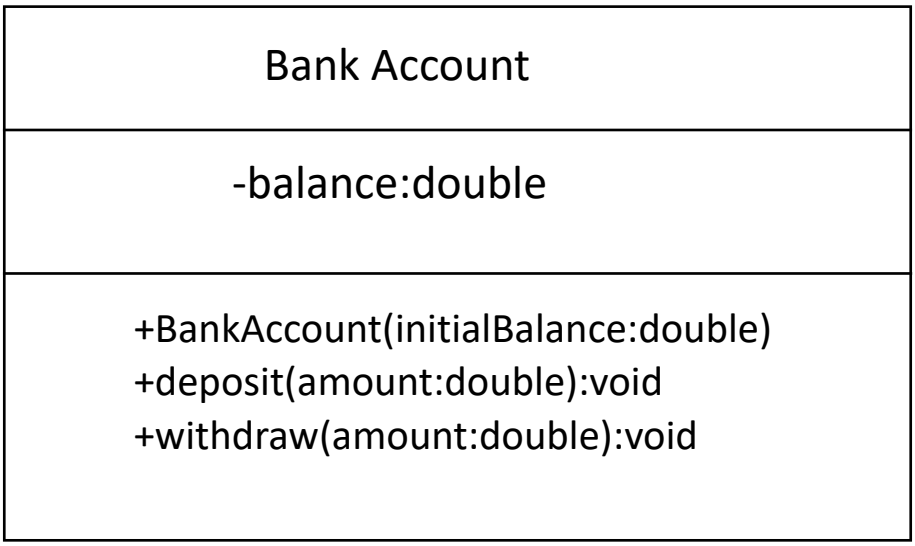
ERROR TABLE:

Code Error	Code rectification
1. Not putting the semi-colon; after calling the function. 2. After Withdrawal,deposit not giving the parenthesis ( ).	1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ).

IMPORTANT POINTS:

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

CLASS DIAGRAM:



WEEK -4

PROGRAM – 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 and display the details of two books.

CODE:

```
 public class Book{
  String title;
String author;
int year_of_publication;
 public Book (String title, String author, int year_of_publication)
{
 this.title = title;
 this.author = author;
 this.year_of_publication = year_of_publication;
}
public void displayDetails() {
System.out.println(this.title);
System.out.println(this.author);
System.out.println(this. year_of_publication);
System.out.println();
  }
}
public class Main {
 public static void main(String[]args) {
 Book book1 =new Book ( "the first frost"," Sang Yan", 1997);
Book book2 =new Book ("hidden love", " Sang xi" , 2007);
System.out.println("Book1 details: ");
book1.displayDetails();
System.out.println("Book2 details: ");
book2.displayDetails();
 }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air ~ % javac Main.java
m.akshaya@Akshaya-MacBook-Air ~ % java Main
Book1 details:
Title: The First Frost
Author: Sang Yan
Year of Publication: 1997

Book2 details:
Title: Hidden Love
Author: Sang Xi
Year of Publication: 2007
```

ERROR TABLE:

Code Error	Code rectification
1. Not defining the function in a file. 2. Two public class files should not be saved in the same file.	1. To call the method we must define a function in a file. 2. Two public class files should be saved in different files.

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air ~ % javac Book.java
Book.java:17: error: class Main is public, should be declared in a file named Ma
in.java
public class Main {
       ^
1 error
```

IMPORTANT POINTS:

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.
2. While defining a method we should also define a function to call that method.

CLASS DIAGRAM:

PROGRAM – 2:

AIM: Create a java Program with class named myclass with 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, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

CODE:

```
 public class myclass {
   Static int count = 0;
   Final double pi=3.14;
public myclass () {
count++;
}
public static void main (String[]args) {
 My class obj1 =new myclass();
My class obj2 =new myclass();
My class obj3 =new myclass();

 System.out.println("count:" + count);
 System.out.println("value of pi:" + obj1.pi);
System.out.println("value of pi:" + obj2.pi);
System.out.println("value of pi:" + obj3.pi);
System.out.println("M.Akshaya");
  }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air ~ % javac myclass.java
m.akshaya@Akshaya-MacBook-Air ~ % java myclass
count:3
value of pi:3.14
value of pi:3.14
value of pi:3.14
M.Akshaya
m.akshaya@Akshaya-MacBook-Air ~ %
```

ERROR TABLE:

Code Error	Code rectification
1. Not Putting the semi-colon after calling a function, 2. Not giving the indentation properly.	1. Put the semi-colon after calling a function. 2. All the indentation must be correct to run the code correct.

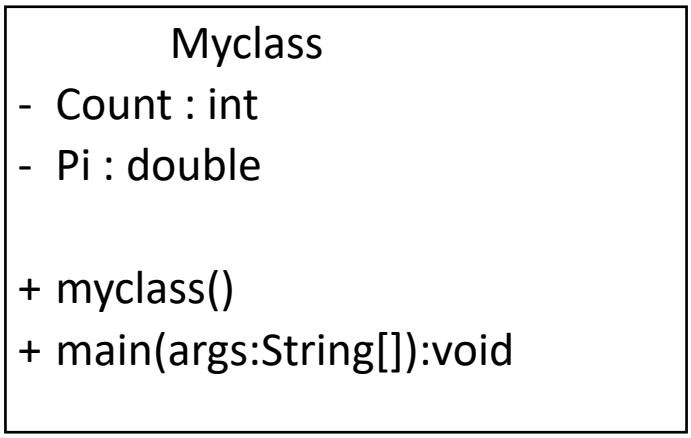
NEGATIVE CASE:

```
11 errors
m.akshaya@Akshaya-MacBook-Air ~ % javac myclass.java
myclass.java:2: error: <identifier> expected
   Static int count = 0;
         ^
myclass.java:3: error: <identifier> expected
   Final double pi=3.14;
        ^
2 errors
```

IMPORTANT POINTS:

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

CLASS DIAGRAM:



WEEK-5

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.

CODE :

```
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;
   }
```

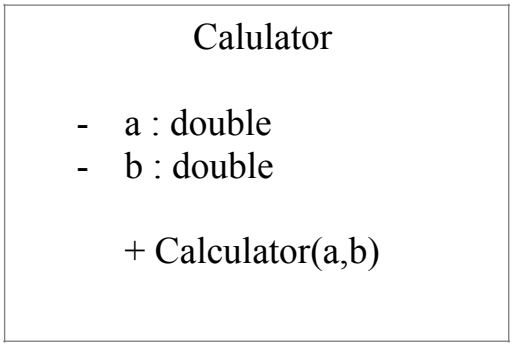
```
}
class Multiplication extends Subtraction {
    public Multiplication(double a, double b) {
        super(a, b);
    }
    public double multiply() {
        return a * b;
    }
}
class Division extends Multiplication {
    public Division(double a, double b) {
        super(a, b);
    }
    public double divide() {
        if (b != 0) {
            return a / b;
        } else {
            System.out.println("Error");
            return Double.NaN;
        }
    }
}
class Final extends Division {
    public Final(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());
    }
}
import java.util.Scanner;
public class allcalculator {
```

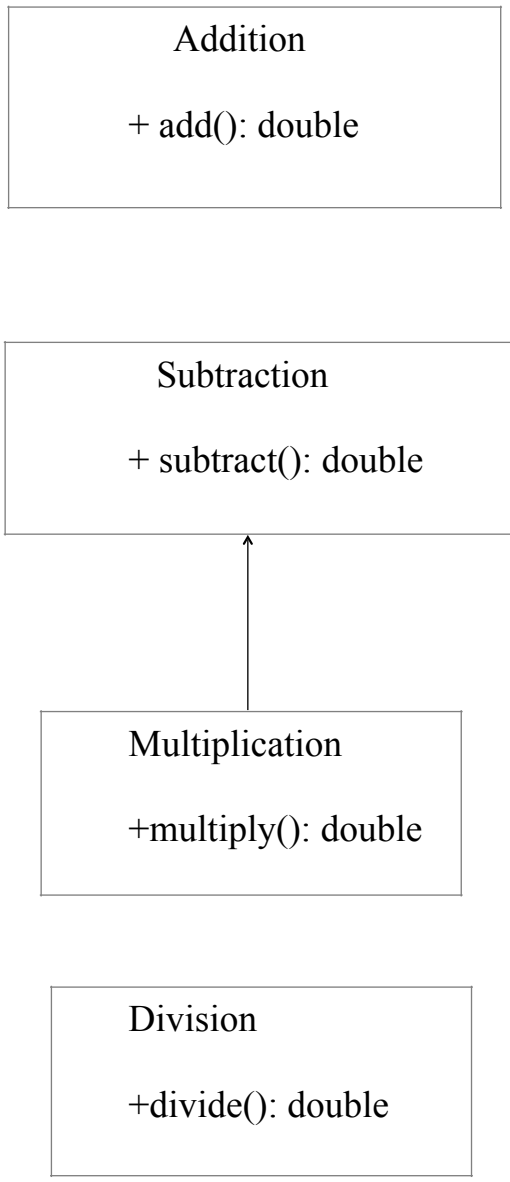
```
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a number: ");
        double a = input.nextDouble();
        System.out.println("Enter b number: ");
        double b = input.nextDouble();
        Final calc = new Final( a,  b);
        calc.displayResults();
        System.out.println("Akshaya");
        input.close();
    }
}
```

OUTPUT:

```
Enter a number:
65
Enter b number:
24
Addition: 89.0
Subtraction: 41.0
Multiplication: 1560.0
Division: 2.7083333333333335
Akshaya
```

CLASS DIAGRAM:





ERROR TABLE:

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.

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air ~ % javac allcalculator.java
allcalculator.java:56: error: class, interface, enum, or record expected
   import java.util.Scanner;
   ^
1 error
```

IMPORTANT POINTS:

1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. in java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.
4. to import a whole package, end the sentence with an asterisk sign(*).

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.

CODE:

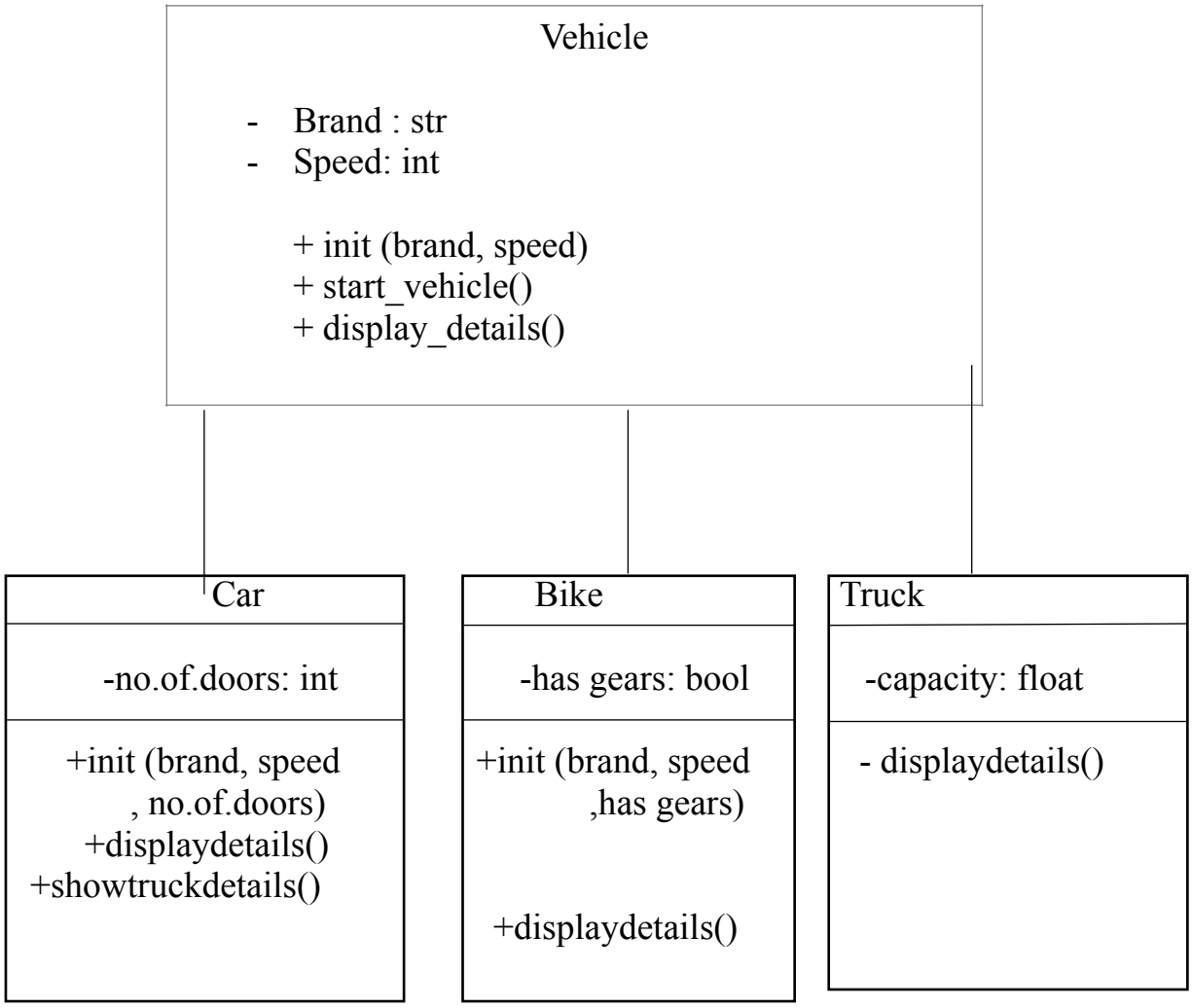
```
public class vehicle {
   public String brand;
   public int speed;
   public vehicle(String brand, int speed) {
      this.brand = brand;
      this.speed = speed;
   }
   public void start() {
      System.out.println(brand + " is starting");
   }
   public void showDetails() {
      System.out.println("Brand: " + brand);
      System.out.println("Speed: " + speed + " km/h");
   }
}
class Car extends vehicle {
   private int noOfDoors;
   public Car(String brand, int speed, int noOfDoors) {
      super(brand, speed);
```

```
        this.noOfDoors = noOfDoors;
    }
    public void showDetails() {
        super.showDetails();
        System.out.println("Number of Doors: " + noOfDoors);
    }
}
class Bike extends vehicle {
    private boolean hasGears;
    public Bike(String brand, int speed, boolean hasGears) {
        super(brand, speed);
        this.hasGears = hasGears;
    }
    public void showDetails() {
        super.showDetails();
        System.out.println("Has Gears: " + (hasGears ? "Yes" :
"No"));
    }
}
class Truck extends vehicle {
    private int capacity;
    public Truck(String brand, int speed, int capacity) {
        super(brand, speed);
        this.capacity = capacity;
    }
    public void showTruck() {
        super.showDetails();
        System.out.println("Capacity: " + capacity + " tons");
    }
}
```

OUTPUT:

```
Car Details
Toyota is starting
Brand: Toyota
Speed: 150 km/h
Number of Doors: 4
Bike Details
Yamaha is starting
Brand: Yamaha
Speed: 120 km/h
Has Gears: Yes
Truck Details
Volvo is starting
Brand: Volvo
Speed: 90 km/h
Capacity: 10 tons
Akshaya
```

CLASS DIAGRAM:

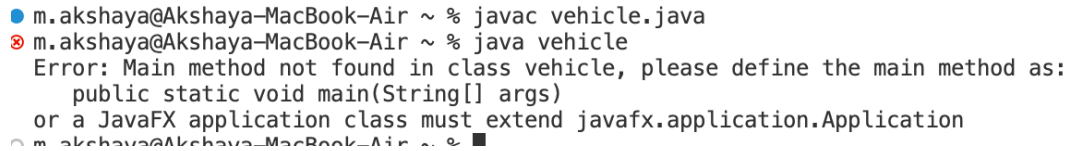


ERROR TABLE:

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.

NEGATIVE CASE:



IMPORTANT POINTS:

1. a constructor helps in initializing an object that doesn't exist.
2. a double method can represent more decimal point numbers than float method.
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:

The oops concepts used in the above program are: Inheritance, encapsulation, polymorphism, abstraction.To add a new vehicle type truck we need to create a truck class that will:

- Include an additional property capacity (in tons).
- Implement a showtruckdetials() method to display the truck's capacity

WEEK-6:

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).

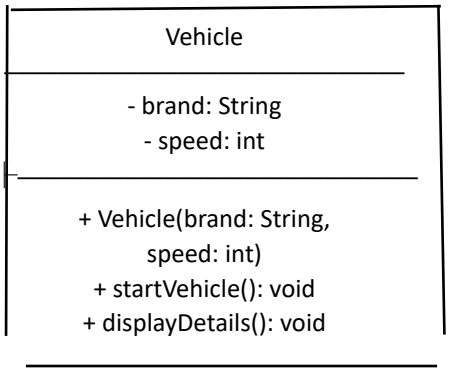
CODE:

```
class vehicle {
   public String car_model;
   public String color;
   public String fuel_type;
   Public vehicle (String car_model, String color, String
fuel_type) {
      this.car_model = car_model;
      this.color = color;
      this.fuel_type = fuel_type;
   }
   void displayDetails() {
      System.out.println("car_model: " + car_model);
      System.out.println("car_color: " + color);
      System.out.println("car_fuel_type: " + fuel_type);
      System.out.println("I have an Porche");
   }}
class Car extends vehicle {
   public Car (String car_model, String color, String fuel_type) {
      super(car_model,color,fuel_type);
   }
   void displayDetails() {
      super.displayDetails();
      System.out.println("I have a ferrari ");
   }}
public class Truck {
   public static void mian(String[]args) {
      vehicle v = new vehicle("Porche","royal blue","diesel");
      v.displayDetails();
      Car c = new Car ("ferrari","red","petrol");
      c.displayDetails();
   }}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProject % java Truck
car_model: Porche
car_color: royal blue
car_fuel_type: diesel
I have an Porche
car_model: ferrari
car_color: red
car_fuel_type: petrol
I have an Porche
I have a ferrari
```

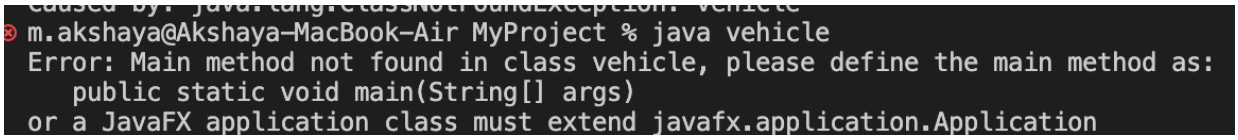
CLASS DIAGRAM:



ERROR TABLE:

Error	Error 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 correctlypasses Benz" to super(car_model, color, fuel_type);

NEGATIVE CASE:



IMPORTANT POINTS:

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

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

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.

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: Create a calculator class with overloaded methods to perform addition of:
1. Add two integers
2. Add two doubles
3. Add three integers

CODE:

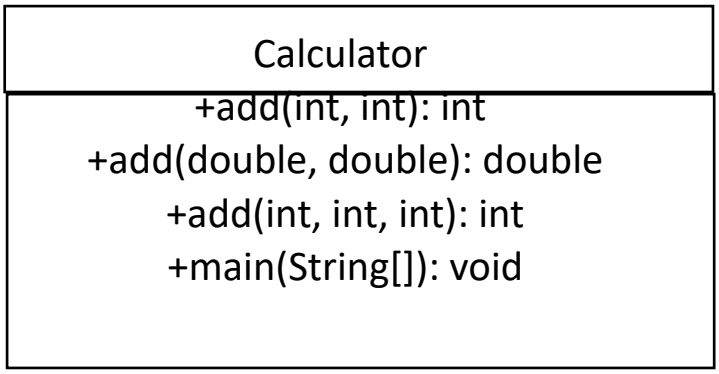
```
public class calculator {
   public int add(int a,int b){
      return a+b;
   }
```

```
    public double add(double a, double b){
        return a+b;
    }
    public int add(int a ,int b,int c){
        return a+b+c;
    }
    public static void main (String[] args) {
        calculator calc = new calculator();
        System.out.println("sum: " + calc.add(20,50));
        System.out.println("sum of 3.5 and 6.2:" +
calc.add(3.5,6.2));
        System.out.println("sum of 20,30 and 20: " +
calc.add(20,30,20));
    }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac calculator.java
m.akshaya@Akshaya-MacBook-Air MyProjects4 % java calculator
sum: 70
sum of 3.5 and 6.2:9.7
sum of 20,30 and 20: 70
```

CLASS DIAGRAM:



ERROR TABLE:

Code Errors	CODE RECTIFICATIONS
1.Method parameters missing spaces (e.g., "int a,int b" should be "int a, int b") 2. Inconsistent indentation in method bodies (some lines not properly aligned)	1.Add proper spacing between Parameters: -old:add(int a,intb) -New:add(int a,int b) 2.Fix indentation: consist 4 space of indentation

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air ~ % cd MyProjects4
m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac calculator.java
calculator.java:13: error: ')' or ',' expected
        System.out.println("sum: " + calc add(20,50));
                                         ^
calculator.java:13: error: ';' expected
        System.out.println("sum: " + calc add(20,50));
                                                ^
```

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-3:

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.

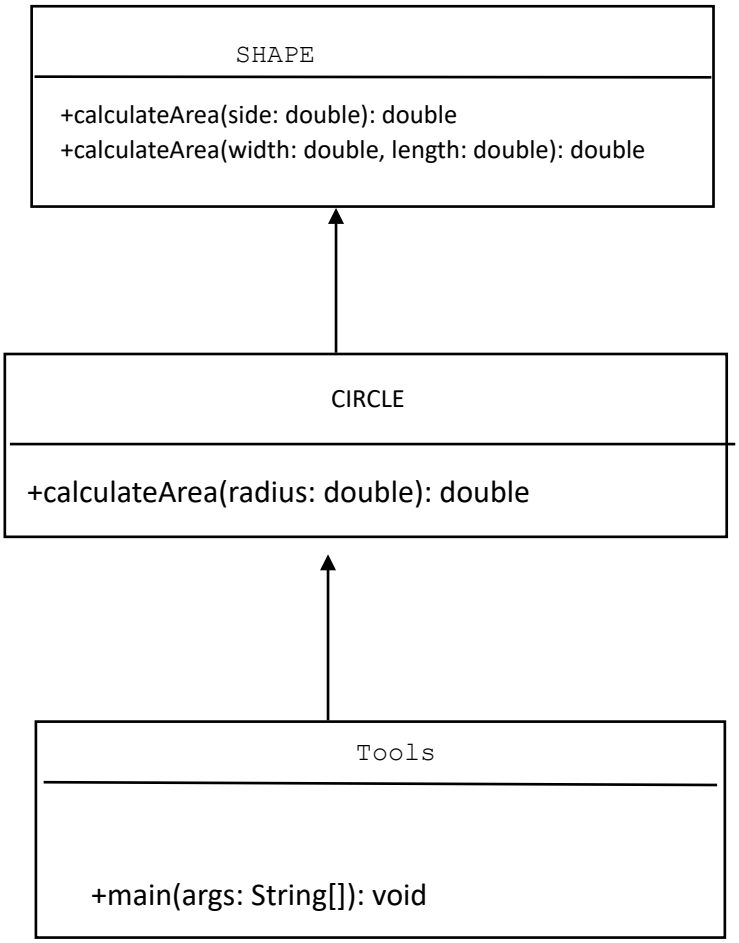
CODE:

```
class shape {
   double calculateArea(double side) {
      return side * side ;
   }
   double calculateArea(double width , double length) {
      return width * length ;
   }
}
class Circle extends shape {
   double calculateArea(double radius) {
      return 3.14 * radius * radius ;
   }
}
public class Tools {
   public static void main(String[] args) {
      shape s = new shape();
      Circle c = new Circle();
      System.out.println("Area of square (side 6): " +
s.calculateArea(6));
      System.out.println("Area of rectangle (6x5): " +
s.calculateArea(6,5));
      System.out.println("Area of circle (radius 4): " +
s.calculateArea(4));
   }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac Tools.java
m.akshaya@Akshaya-MacBook-Air MyProjects4 % java Tools
Area of square (side 6): 36.0
Area of rectangle (6x5): 30.0
Area of circle (radius 4): 16.0
```

Class Diagram:



ERROR TABLE:

Error	Error Rectification
1. **Method calls in** main **are missing an** **1.** **object reference** (e.g., calculateArea(4) instead of s.calculateArea(4)). **2.Circle class method does not override theparent class method properly.**	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.

Negative case:

```
m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac Tools.java
Tools.java:15: error: <identifier> expected
    public staic void main(String[] args) {
                ^
1 error
```

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)).

PROGRAM-4:

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.

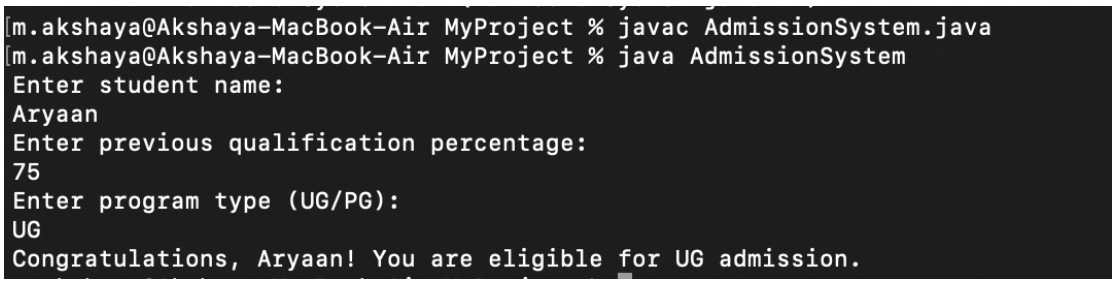
1. UG admission recquire a minimum of 60%.
2. PG admission recquire a minimum of 70%.

CODE:

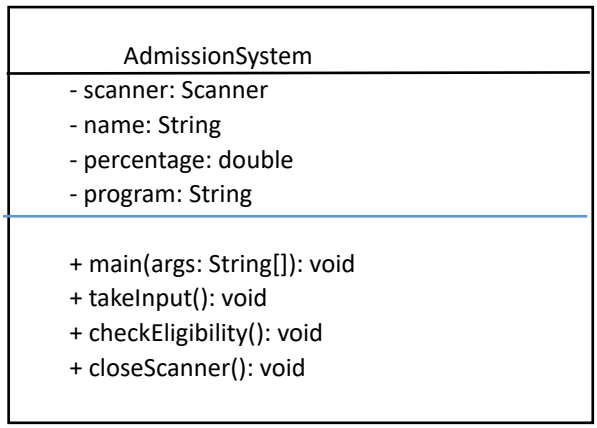
```
import java.util.Scanner;

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

      System.out.println("Enter student name:");
      String name = scanner.nextLine();
```

```
        System.out.println("Enter previous qualification
percentage:");
        double percentage = scanner.nextDouble();

        System.out.println("Enter program type (UG/PG):");
        String program = scanner.next().toUpperCase();

        if (program.equals("UG")) {
            if (percentage >= 60) {
                System.out.println("Congratulations, " + name + "! You
are eligible for UG admission.");
            } else {
                System.out.println("Sorry, " + name + ". You do not
meet the UG admission criteria.");
            }
        } else if (program.equals("PG")) {
            if (percentage >= 70) {
                System.out.println("Congratulations, " + name + "! You
are eligible for PG admission.");
            } else {
                System.out.println("Sorry, " + name + ". You do not
meet the PG admission criteria.");
            }
        } else {
            System.out.println("Invalid program type entered. Please
enter UG or PG.");
        }

        scanner.close();
    }
}
```

OUTPUT:



CLASS DIAGRAM:



Error Table:

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

NEGATIVE CASE:

```
C:\Users\svehy\OneDrive\Documents\MATLAB\javascriptusinghtml>JAVAC ADMISSIONSYAYTEM
error: Class names, 'ADMISSIONSYAYTEM', are only accepted if annotation processing is explicitly requested
1 error

C:\Users\svehy\OneDrive\Documents\MATLAB\javascriptusinghtml>
```

Important Points:

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

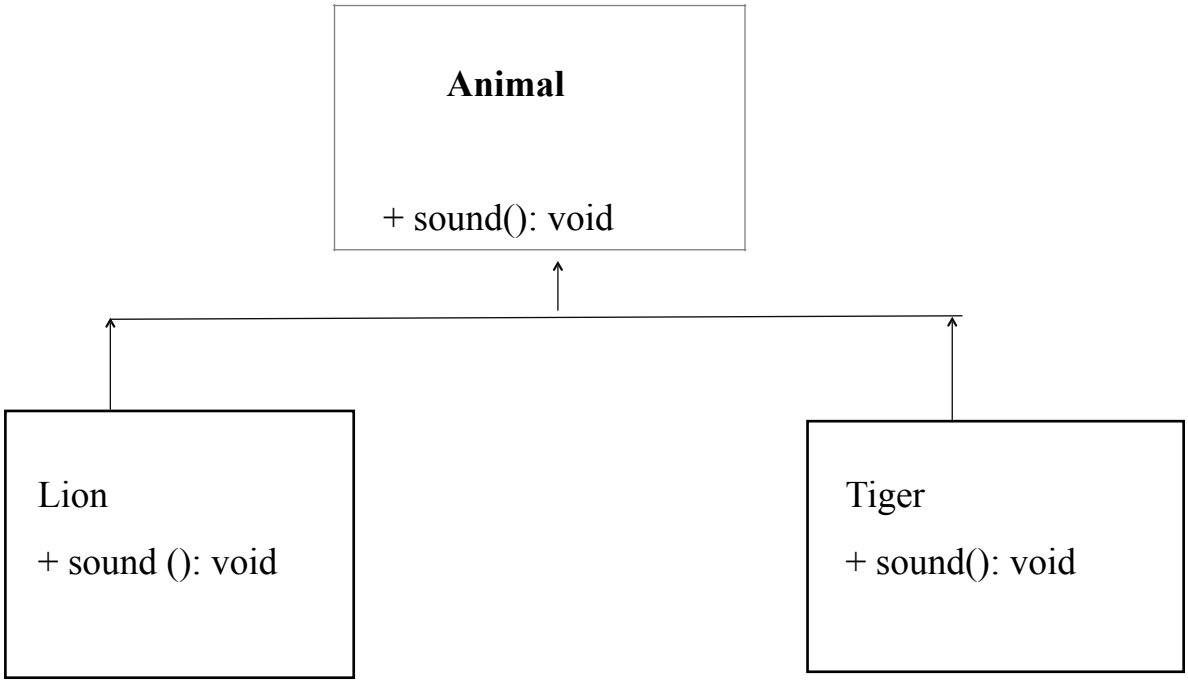
Decision Making with Conditions: Uses if-else statements to check eligibility criteria.
String Handling: Converts program input to uppercase (toUpperCase()) to handle case variations.
Closing Scanner: Properly closes scanner using scanner.close(); to prevent resource leaks.

WEEK-7

PROGRAM-1:

AIM: Write a java program to create an abstract class animal with an abstract method called sound().Create subclasses lion and tiger that extends the animal class and implement the sound() method to make specific sound for each animal.

CLASS DIAGRAM:



CODE:

```
abstract class Animal{
  abstract void sound();
}
class Lion extends Animal {
   void sound() {
      System.out.println("lion says : Roarrr!!");
   }
```

```
}
class Tiger extends Animal {
   void sound() {
      System.out.println("tiger says : Grrr!!");
   }
}
class AnimalTest {
   public static void main (String[] args){
       Lion narasimha = new Lion();
       narasimha.sound();
       Tiger manyampuli = new Tiger();
       manyampuli.sound();
   }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProject % java AnimalTest
lion says : Roarrr!!
tiger says : Grrr!!
```

ERROR TABLE:

ERROR	ERROR RECTIFICATION
1. Error while printing the variables. 2. Incorrect declaration of integer.	1. Give the plus sign while printing. 2. Give input.nextInt(), where I should be capital.

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air MyProject % javac AnimalTest.java
m.akshaya@Akshaya-MacBook-Air MyProject % java Animaltest
Error: Could not find or load main class Animaltest
Caused by: java.lang.NoClassDefFoundError: Animaltest (wrong name: AnimalTest)
```

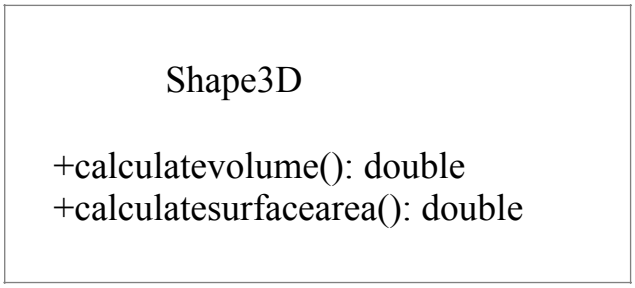
IMPORTANT POINTS:

1. We override the methods in the superclass.
2. Here we are using the heirarchial inheritance.

PROGRAM-2:

AIM: Write a java program to create an abstract class Shape3D with abstract methods calculatevolume() and calculate surfacearea().Create the subclasses sphere and cube that extend the class Shape3D and implement the respective methods to calculate the volume ans surface area of each shape.

CLASS DIAGRAM:



CODE:

```
abstract class Shapes3D{
public void calculateVolume(){
}
public void calculateSurfaceArea(){
}
}
class Sphere extends Shapes3D {
public void calculateVolume(double r){
double vol=1.33*3.14*r*r*r;
```

```
System.out.println("The volume of sphere is: "+vol);
}
public void calculateSurfaceArea(double r){
double area=4*3.14*r*r;
System.out.println("The surface area of sphere is: "+area);
}
}
class Cube extends Shapes3D {
public void calculateVolume(double side){
double vol=side*side*side;
System.out.println("The volume of the cube is: "+vol);
}
public void calculateSurfaceArea(double side){
double area=6*side*side;
System.out.println("The surface area of cube is: "+area);
}
}
class Shapes{
public static void main(String[] args){
Sphere s=new Sphere();
s.calculateVolume(6);
s.calculateSurfaceArea(6);
Cube c=new Cube();
c.calculateVolume(4);
c.calculateSurfaceArea(4);
System.out.println("M.Akshaya';24218;CSE-C");
}
}
```

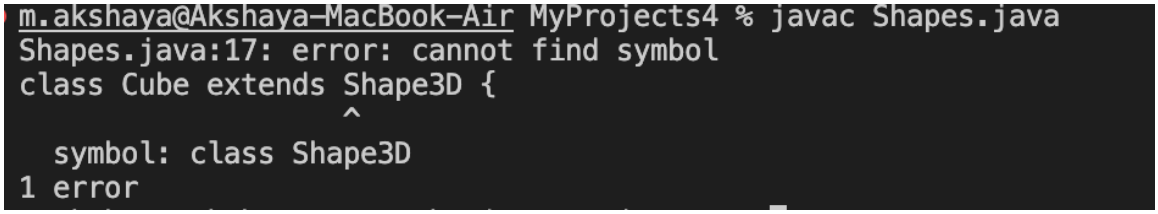
OUTPUT:

```
[m.akshaya@Akshaya-MacBook-Air MyProjects4 % javac Shapes.java
[m.akshaya@Akshaya-MacBook-Air MyProjects4 % java Shapes
The volume of sphere is: 902.0592000000001
The surface area of sphere is: 452.15999999999997
The volume of the cube is: 64.0
The surface area of cube is: 96.0
M.Akshaya';24218;CSE-C
```

ERROR TABLE:

ERROR	ERROR RECTIFICATION
1. Wrong datatype entered. 2. Object not defined.	1. Enter the correct datatype i.e double instead of int. 2. Enter the correct object and if not create new one.

NEGATIVE CASE:



IMPORTANT POINTS:

1. Here we used the abstract to declare an abstract class.
2. Abstract classes and methods help us to declare the methods without declaring the return type in them.

To get the values, we declared a constructor for each subclass and initialized values for them

PROGRAM-3:

AIM: Write a java program using abstract class to define a method for pattern printing.

- Create an abstract class named PatternPrinter with an abstract method printPattern(int) and a concrete method to display the pattern title.
- Implement two subclasses:

1. Starpattern- prints a right-angled triangle of star (*).
2. Numberpattern - prints a right-angled triangle of increasing numbers.

• In the main () method, create objects of both

Example Output for n = 5:

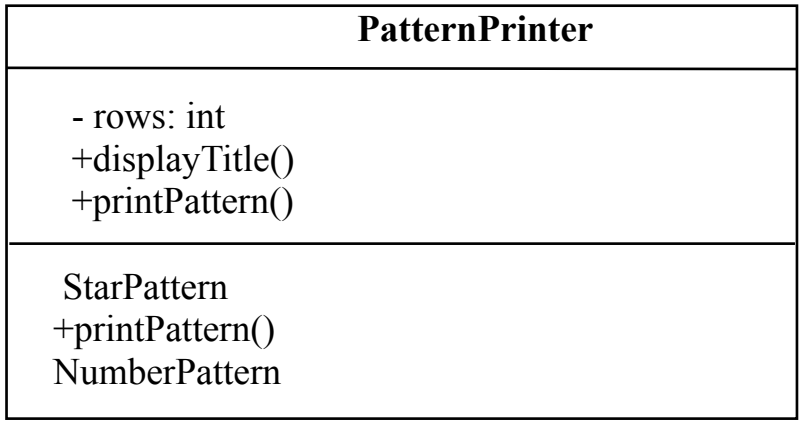
Star Pattern

```
*
* *
* * *
* * * *
* * * * *
```

Number Pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

CLASS DIAGRAM:



CODE:

```
abstract class PatternPrinter {
 int rows;

 PatternPrinter(int rows) {
```

```
 this.rows = rows;
 }

 abstract void printPattern();

 void displayTitle(String title) {
 System.out.println("\n" + title);
 }
}

class StarPattern extends PatternPrinter {
 StarPattern(int rows) {
 super(rows);
 }

 void printPattern() {
 for (int i = 1; i <= rows; i++) {
 for (int j = 1; j <= i; j++) {
 System.out.print("* ");
 }
 System.out.println();
 }
  }
}

class NumberPattern extends PatternPrinter {
 NumberPattern(int rows) {
 super(rows);
 }

 void printPattern() {
 for (int i = 1; i <= rows; i++) {
 for (int j = 1; j <= i; j++) {
 System.out.print(j + " ");
 }
 }
 }
}
```

```
public class PatternPrinter2 {
 public static void main(String[] args) {
 System.out.println("Name:M.Akshaya,Section:cse-c,Roll
NO:AV.SC.U4CSE24218");
int numberOfRows = 5;

 PatternPrinter star = new StarPattern(numberOfRows);
 star.displayTitle("Star Pattern");
 star.printPattern();

PatternPrinter number = new
NumberPattern(numberOfRows);
 number.displayTitle("Number Pattern");
 number.printPattern();
 }
}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProject % javac PatternPrinter2.java
m.akshaya@Akshaya-MacBook-Air MyProject % java PatternPrinter2
Name:M.Akshaya,Section:cse-c,Roll NO:AV.SC.U4CSE24218

Star Pattern
*
* *
* * *
* * * *
* * * * *

Number Pattern
1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 %
```

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air MyProject % javac PatternPrinter2.java
PatternPrinter2.java:43: error: class Patternprinter2 is public, should be declared in a file named Patternprinter2.java
public class Patternprinter2 {
       ^
1 error
```

ERROR TABLE:

CODE ERROR:	ERROR RECTIFICATION
1) Class name and file name should match 2) Subclass doesn't override abstract method	1) Save file as main.java 2)implement printpattern()in all subclasses

IMPORTANT POINTS:

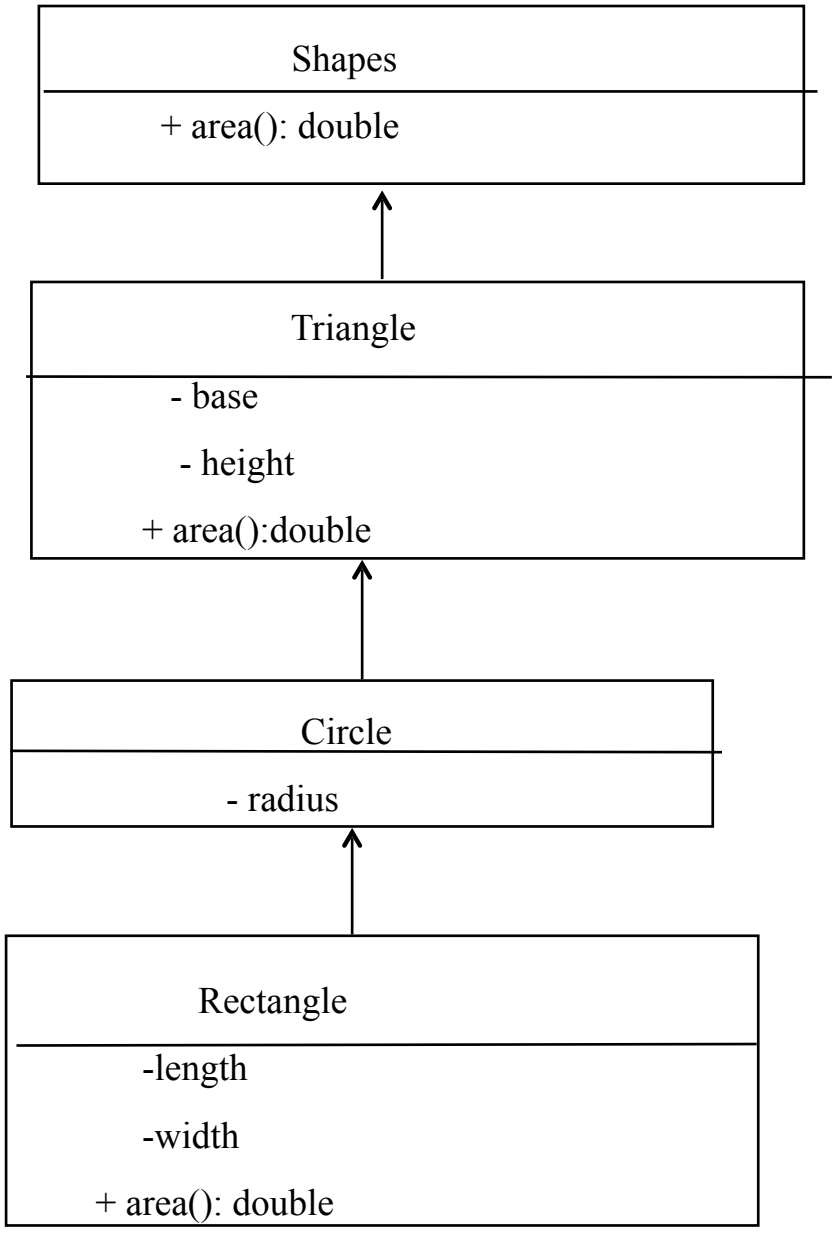
- Use abstract classes to enforce a common structure for pattern printing.
- PatternPrinter is the abstract class defining the common template.
- Subclasses (StarPattern, NumberPattern) provide specific implementations.
- displayTitle() is a concrete method shared by all subclasses.

WEEK-8

PROGRAM-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:

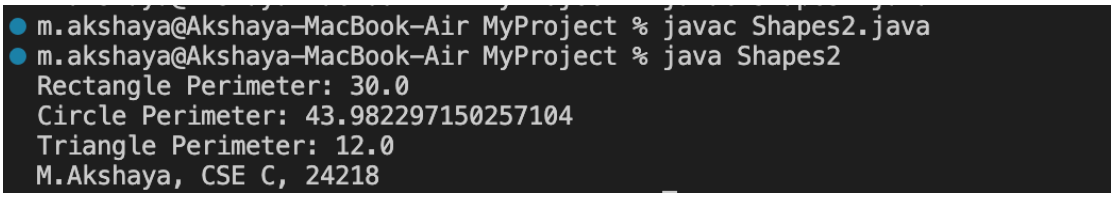


CODE:

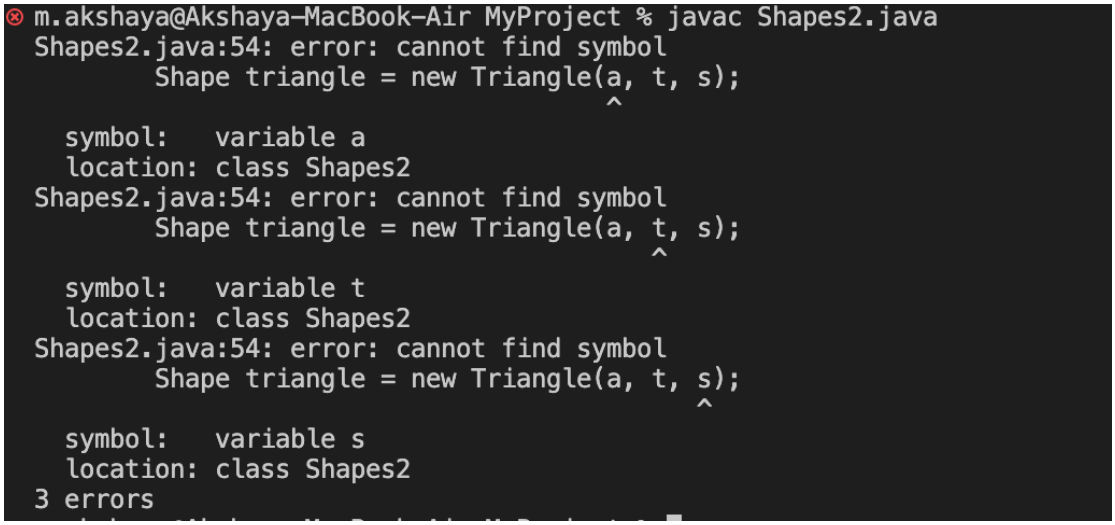
```
interface Shape {
   double getPerimeter();
}
class Rectangle implements Shape {
   private double length;
   private double width;
   public Rectangle(double length, double width) {
      this.length = length;
      this.width = width;
   }
   @Override
   public double getPerimeter() {
      return 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 sideA;
   private double sideB;
   private double sideC;
   public Triangle(double sideA, double sideB, double sideC) {
      this.sideA = sideA;
      this.sideB = sideB;
      this.sideC = sideC;
}
 public double getPerimeter() {
      return sideA + sideB + sideC;
```

```
   }}
public class shape2 {
   public static void main(String[] args) {
      Shape rectangle = new Rectangle(10, 5);
      Shape circle = new Circle(7);
      Shape triangle = new Triangle(3, 4, 5);
      System.out.println("Rectangle Perimeter: " +
rectangle.getPerimeter());
      System.out.println("Circle Perimeter: " +
circle.getPerimeter());
      System.out.println("Triangle Perimeter: " +
triangle.getPerimeter());
      System.out.println("M.Akshaya, CSE C, 24218");
   }
}
```

OUTPUT:



NEGATIVE CASE:



ERROR TABLE:

CODE ERROR:	ERROR RECTIFICATION
1)Class name "Shapes" is inconsistently used (should be consistent capitalization)	1)Change to consistent capitalization (either all "Shapes" or all "Shapes")
2)Base class method area() returns 0 by default - better to make it abstract	2)Consider making Shapes abstract with abstract area() method

IMPORTANT POINTS:

Inheritance Hierarchy:

The traingle, Circle and Rectangle classes all inherit from the base Shapes class (note: class name is misspelled as "Shapes" in some places and "Shapes" in others).

Polymorphism: Each subclass overrides the area() method to provide its own implementation, demonstrating polymorphic behaviour.

Encapsulation: All shape classes properly encapsulate their attributes (base, height, radius, length, width) as private fields.

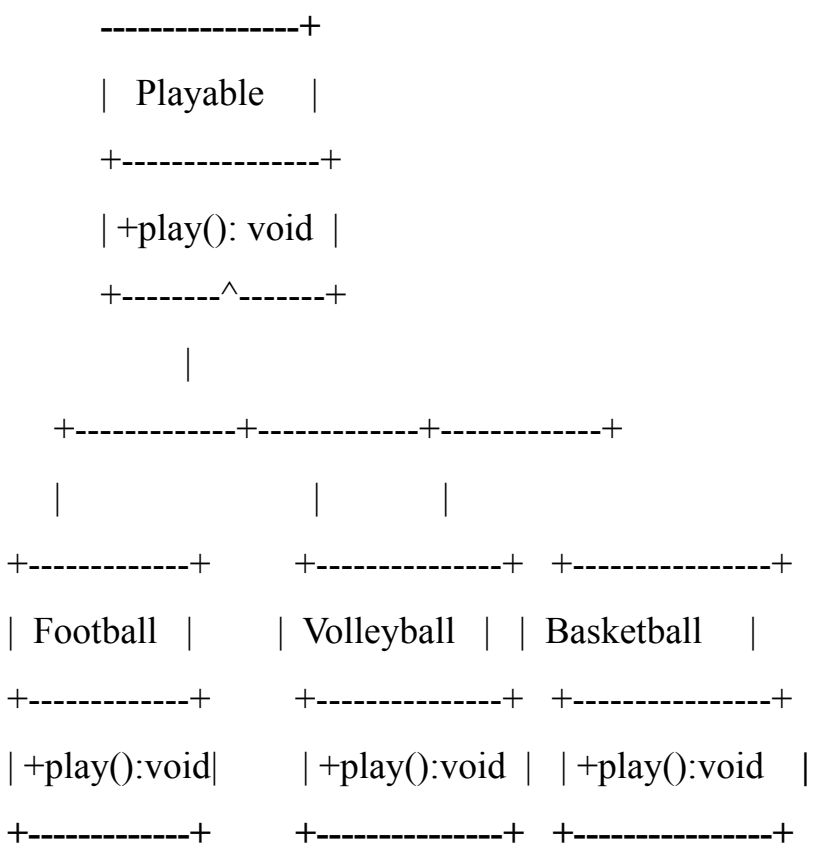
Method Overriding: The area() method is overridden in each subclass with the appropriate calculation formula for that shape.

Main Class: The Shape2 class demonstrates the use of these shapes by creating instances and calling their area() methods.

PROGRAM-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 {
   abstract void play();
}class Football implements Playable {
 public void play() {
System.out.println("some people play football in the Ground.");
```

```
    }}
class Volleyball implements Playable {
    public void play() {
        System.out.println("some people play volleyball in the
Stadium.");
    }}
class Basketball implements Playable {
    public void play() {
        System.out.println("some people play basket ball in the
Rain.");
    }}
public class play2 {
    public static void main(String[]args) {
        System.out.println(“M.Akshaya");
        System.out.println("AV.SC.U4CSE24218");
        System.out.println("CSE-C");
        Playable f = new Football();
        Playable v = new Volleyball();
        Playable b = new Basketball();
        f.play();
        v.play();
        b.play();
    }}
```

OUTPUT:

```
m.akshaya@Akshaya-MacBook-Air MyProject % javac play2.java
m.akshaya@Akshaya-MacBook-Air MyProject % java play2
M.Akshaya
AV.SC.U4CSE24218
CSE-C
some people play football in the ground
some people play volleyball in the Stadium
some people play basketball in the rain
```

NEGATIVE CASE:

```
m.akshaya@Akshaya-MacBook-Air MyProject % javac play2.java
play2.java:26: error: cannot find symbol
        playable b = new Basketball();
        ^
  symbol:   class playable
  location: class play2
1 error
```

ERROR TABLE:

Code Error	**Code rectification**
1. Declaring an abstract class instead of interface class. 2. Not declaring public in each class.	1. Declare an interface class instead of abstract class. 2. Declare public infront of each class.

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