**LAB MANUAL**



**ROLLNO:AV.SC.U4CSE24120**

**NAME:REVANTH G**

**SECTION: CSE-B**

**WEEK-1:**

**Aim:** How to install jdk and first program on

printing student details*.*

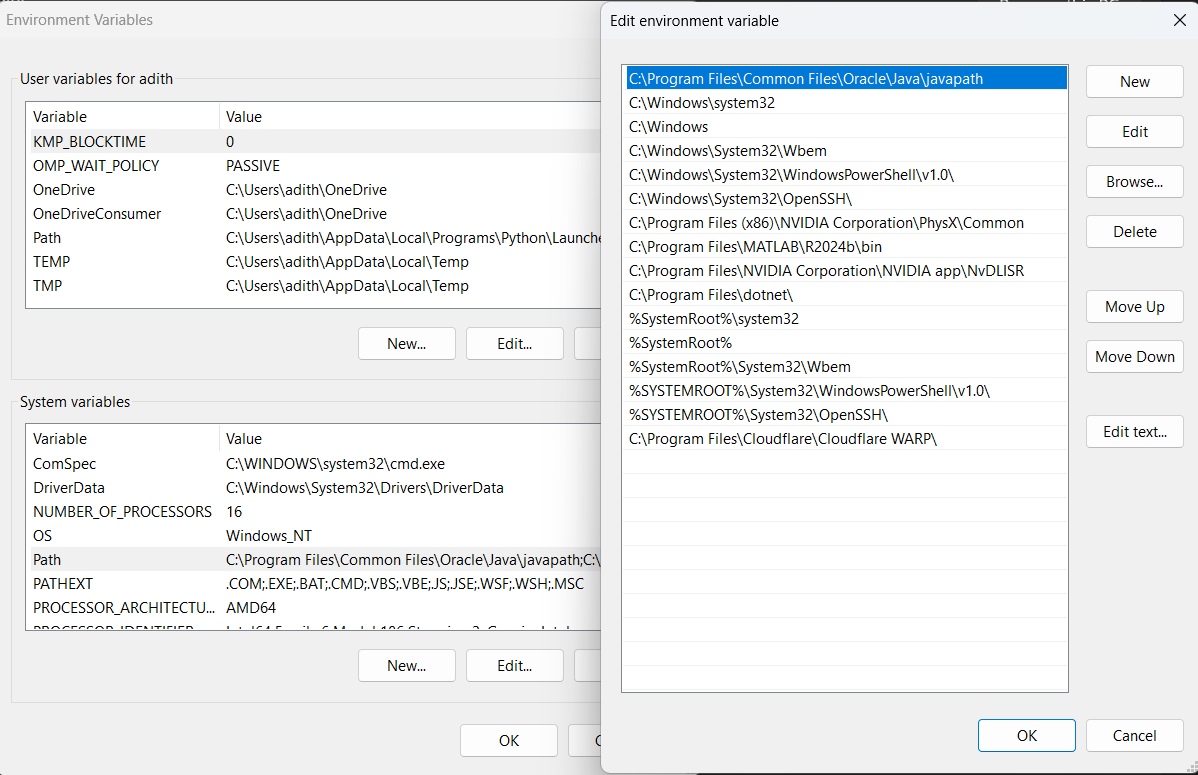
**Step-1:** Download JDK-21 from oracle website

**

**Step-2:**Install the JDK-21 with accepting terms and

conditions according to the respective windows.

**Step-3**:Setting up environmental variables.



\*Windows c -> C-drive -> program files ->Java -

>JDK-21->select bin

\*Select and open environmental variable in search

bar-> either select system variables or user

variables-> select path-> click edit->New-> paste

the bin-> finish the setup(apply the changes).

~for verifying the installed version

Open cmd-> type java --version

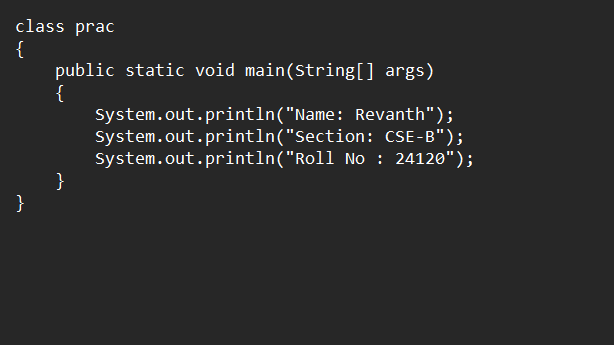
~command propt

Javac filename.java ->compiling.

Java filename.java ->displaying

**PROGRAM-1(Rectified):**

Step1: create new notepad folder and write the code of java program that prints name , roll number & section of a student.

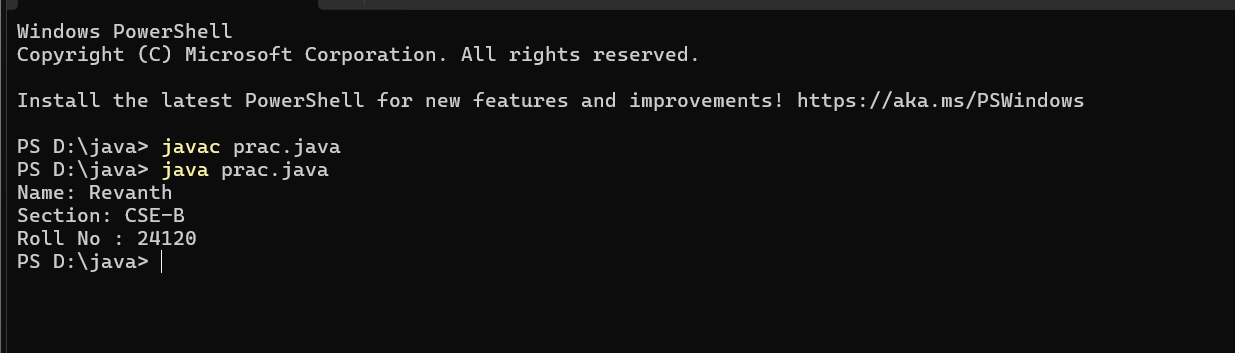
******

Step2: Save the file as StudentInfo.java and open the notepad file in command prompt and compile javac prac.java.

Step3: So ,we are getting 3 errors which is in the printing statement. Go to notepad and correct the mistake in the printing statement.

Step4: Now save the java file and open the command window and type javac prac .java and compile it.

**Output:**



**WEEK-2:**

**PROGRAM-1:**

**Aim:** Write a java program for SI

**

**Output:**

******

**ERROR TABLE*:***

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

**PROGRAM-2:**

**Aim:**Write a program in java for area of rectangle.

**

*IMPORTANT POINTS:*

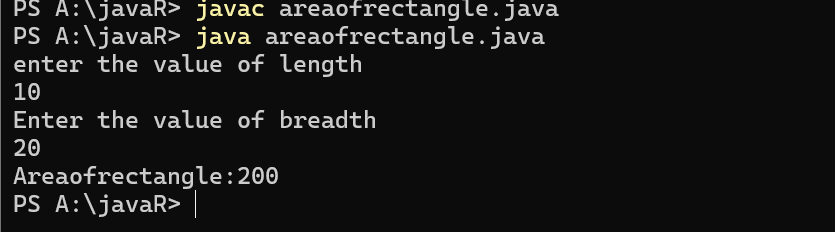
1. *Area of a rectangle is area = l\*b, where*

*L = length of a side of the rectangle,*

*B= breadth of a side of the rectangle.*

1. *Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.*

**Output:**

******

**ERROR TABLE:**

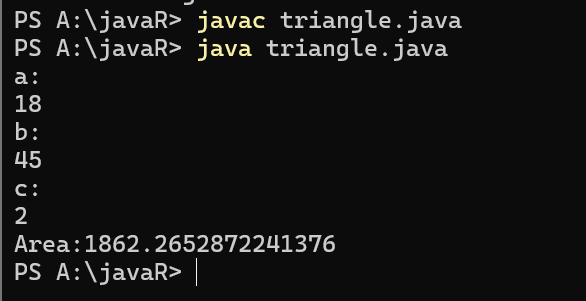
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type as double instead of int. | 1.We should give iterative statements correctly.  2.We should give the data type as int for integers. |

**PROGRAM-3:**

**Aim:**Write a program in java for area of triangle using heron’s formula.



**Output:**

******

ERROR TABLE:

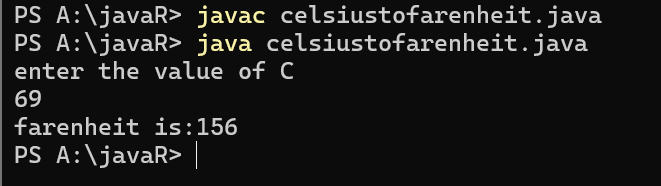
|  |  |
| --- | --- |
| **Code Error** | Code rectification |
| 1.While printing the variable not giving + sign.  2.Not closing the scanner. | 1.We should give correct indentation.  2.Closing the scanner is must. |

**PROGRAM-4(a):**

**Aim:**Write a program in java for converting temperature from celsius to fahrenite.



OUTPUT:

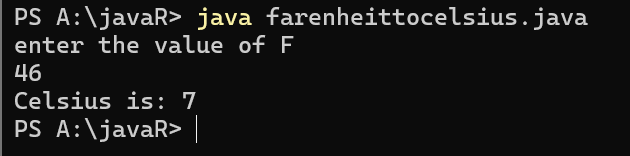


**PROGRAM-4(b):**

**Aim:**Write a program in java for converting temperature from fahrenite to celsius.

******

**Output:**

******

**ERROR TABLE:**

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

**PROGRAM-5:**

**Aim:**Write a program in java for factorial of a number.

******

IMPORTANT POINTS:

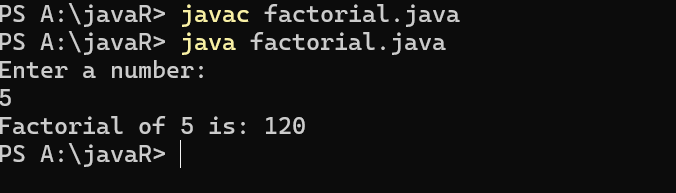
1. While the for loop the data inside the parenthesis indicates the Initial expression

Test expression and

Update expression.

1. Here “factorial\*=I” means factorial = factorial\*I.
2. Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points.

OUTPUT:



ERROR TABLE:

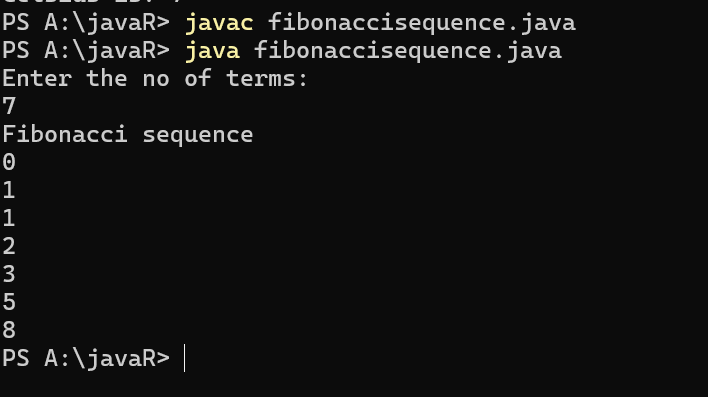
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type as double instead of int. | 1.We should give iterative statements correctly.  2.We should give the data type as int for integers. |

**PROGRAM-6:**

**Aim:**Write a program in java for fibonacci series.



OUTPUT:



ERROR TABLE:

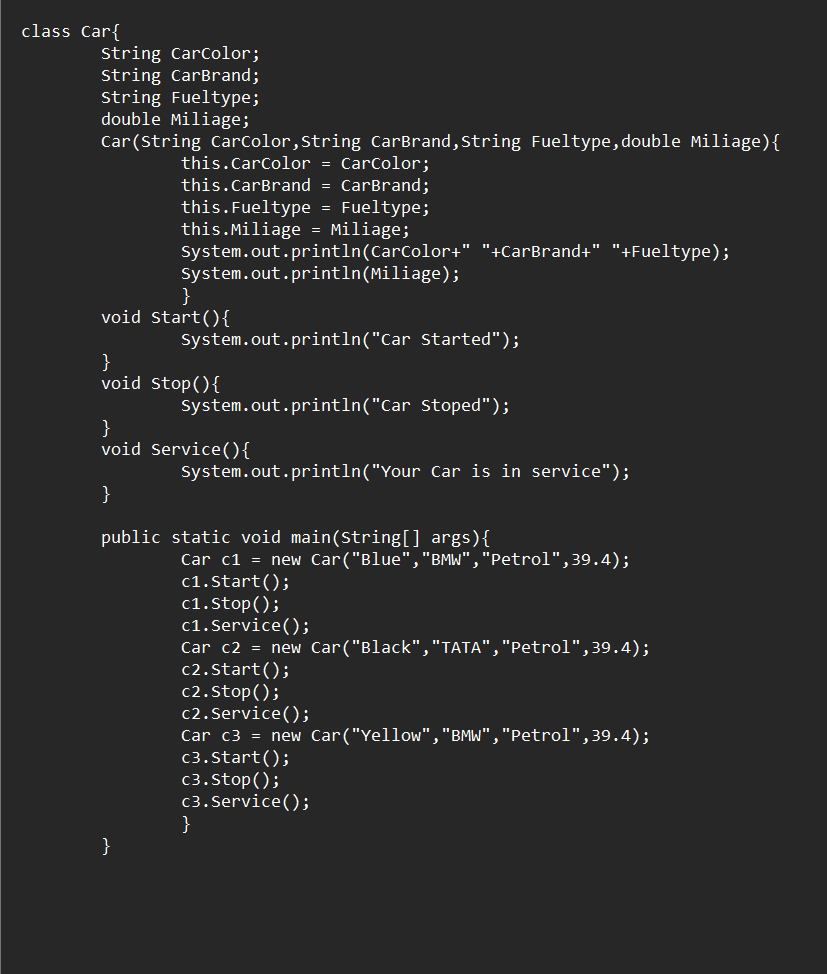
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not giving parenthesis after closing the input. | 1.Should not give space between next and Double.  2.We must put parenthesis after closing the input. |

**Week 3**

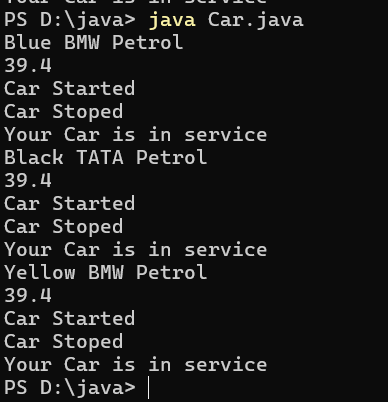
**Program 1:**

**Aim :** Write a java program with the following instructions

1. Create a class with name “Car”
2. Create 4 attributes named CarColor,CarBrand,Fueltype,Miliage
3. Create 3 methods named Start,Stop,Service
4. Create 3 objects named c1,c2,c3
5. Create a constructor with parameters CarColor,CarBrand,Fueltype,Miliage



**Output :**

****

IMPORTANT POINTS:

1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function are called objects, we can give any objects

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.I have kept ‘,’ between variables in print statement | 1.Insted of ‘,’ we should use ‘+’ between  variables in print statement |

**Class diagram:**

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

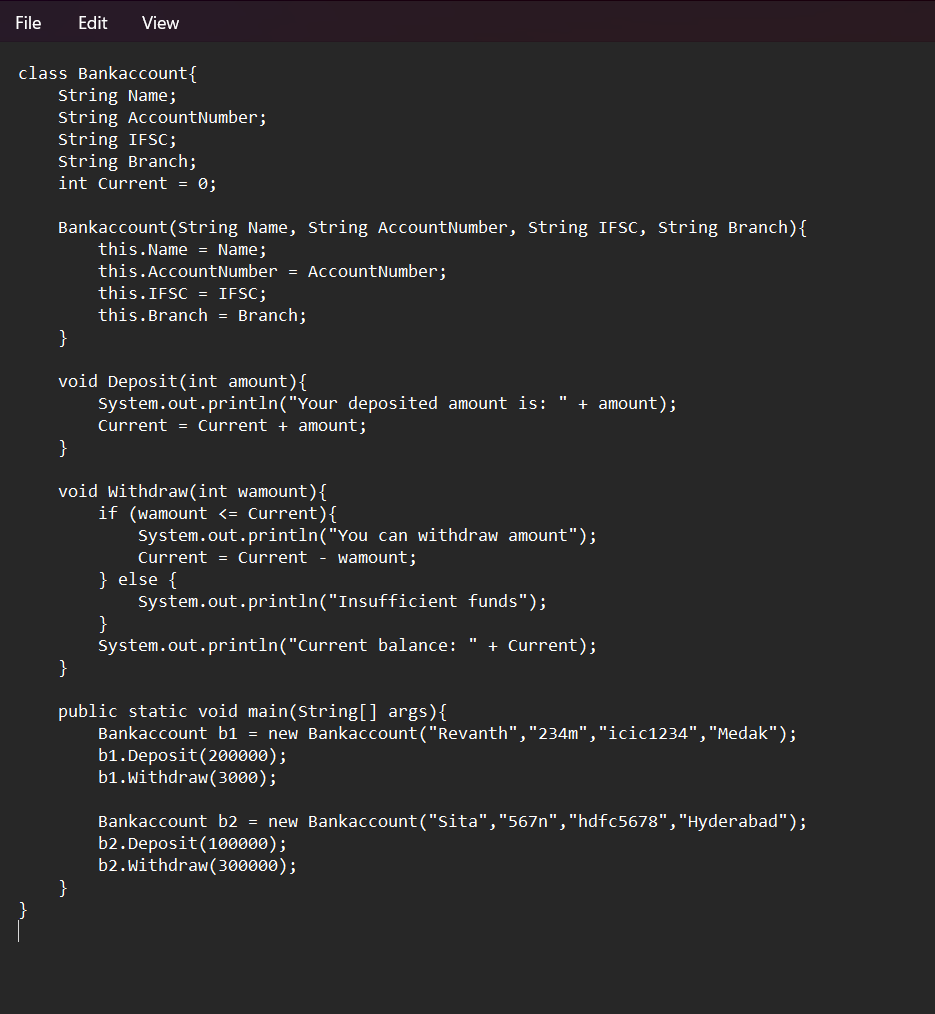
**Program 2:**

**Aim:** Create a class named bank account with methods Deposit,withdraw,were the deposit method should accepts a parameter and when this method is called the deposited amount should added to current balance, in addition to that when a withdraw method is called it has to verify where current balance, if current balance is less then withdraw amount , then “There are insufficient funds” message should display.

🡪Use the constructor to display the details of the customer,(Name,AccountNumber,IFSE,Branch)

🡪Also Create two customer objects

**Program:**

****

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. The condition checking in the withdrawal amount should be <= | 1. Change the condition to correct form |

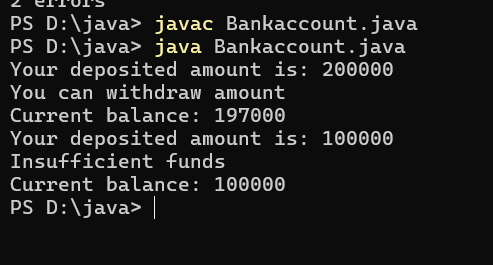
IMPORTANT POINTS:

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:**

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

**Output:**

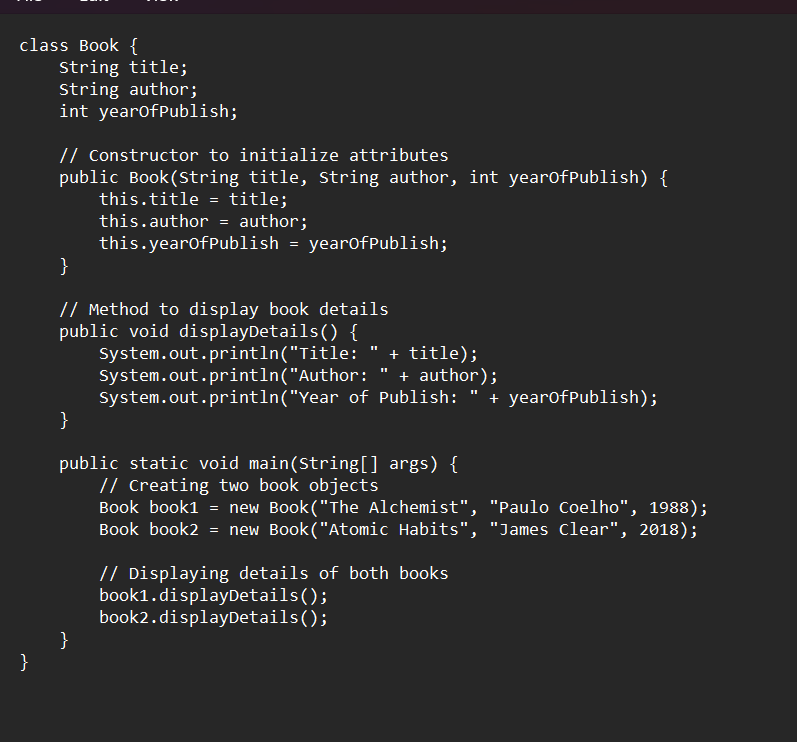


**Week 4:**

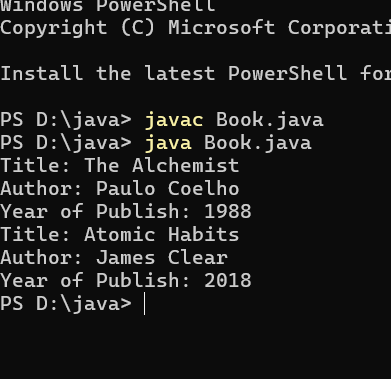
**Program 1**

**Aim:** Write a java program with class named ‘Book’ the class should contain various attributes such as title ,author, year of publish it should also contain a constructor with parameters which inisalizes title, author, year of publish .Create a method which display’s the details of the books ,Display the detailes of two books i.e create two objects and display their detailes.

**Program:**



**Output:**

****

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. |

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:

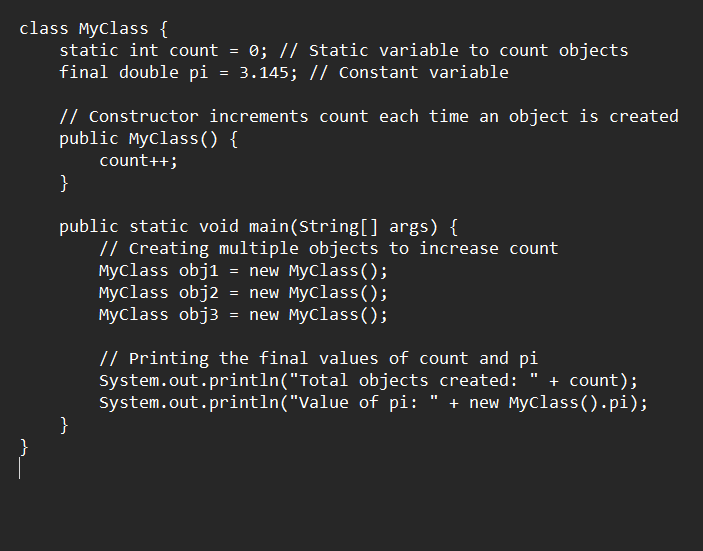
|  |
| --- |
| Book   * Title: String * Author: String * Year of publication: int   + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

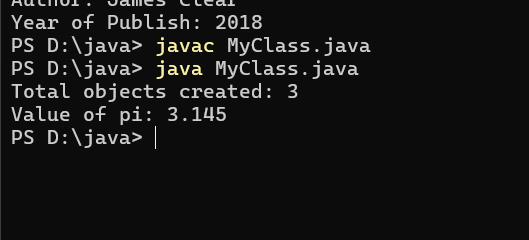
**Program 2:**

**Aim:** Create a java program with class named “MyClass”, with a static variable count of int datatype and inisalized to zero and a constant variable ‘pi’

of type “double” insilaize to 3.145 as attributes of the class, Now define a count variable each time an object of “MyClass” is created.Finally print the final values of count and ‘pi’ variables.

**Program:**

**Output:**

****

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. |

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:

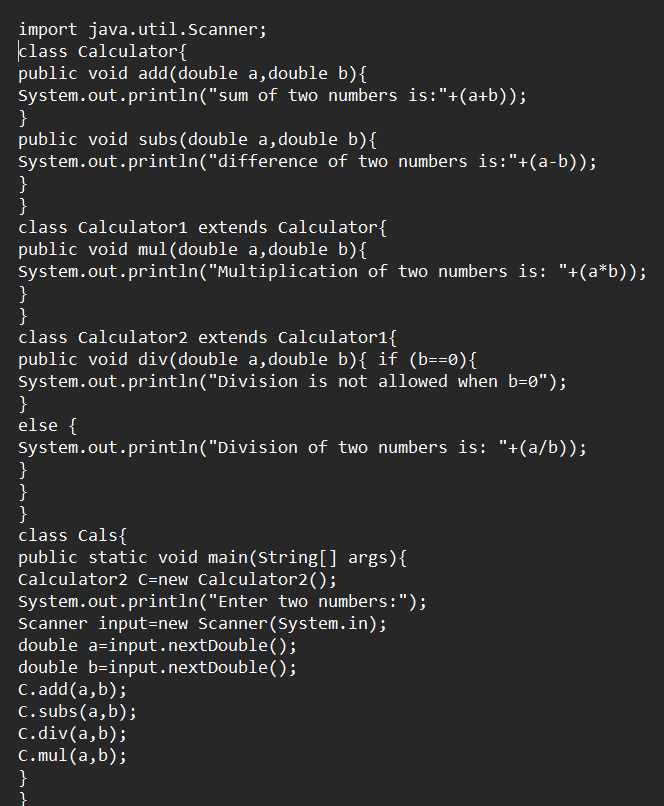
|  |
| --- |
| Myclass   * Count: int * Pi: double   + myclass( )  + main(args: String[]): void |

**Week-5**

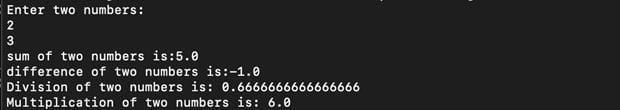
**Program 1:**

**Aim:** Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output.

**Program:**



**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |

|  |  |  |
| --- | --- | --- |
| **1.** | **Semi colon (;)** | **Givethe semi colon (;) in each line where it is**  **required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

**Class Diagram:**

+add(doublea,doubleb): void

+subs(doublea,double b): void

calculator

+multiplication

(double a,double b): void

**Calculator1**

+div(doublea,double b): void

**Calculator2**

**Program 2:**

**Aim:** A vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each vehicle such as brand and speed**.**

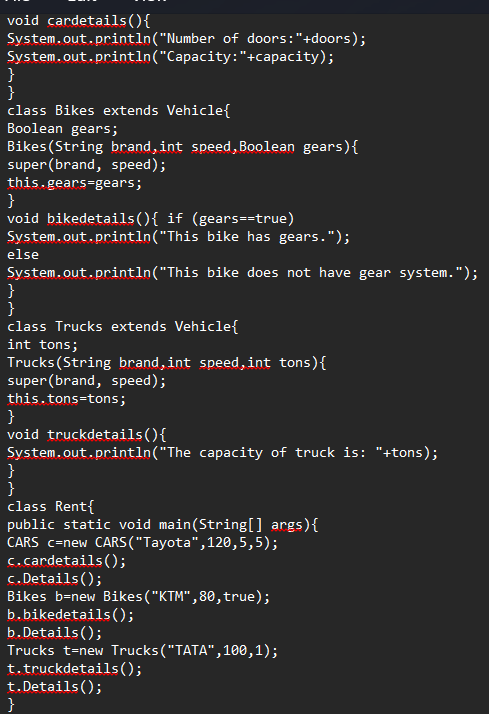
**Questions:**

1. Which OOPS concepts used in the above program? Explain why itis useful in this scenario.
2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?

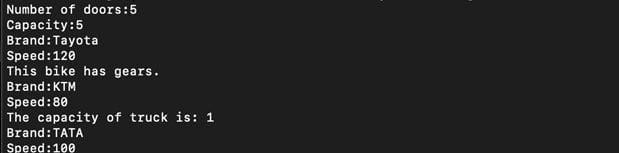
Truck should include and additionalpropertycapacity (in tons).

* + - 1. CreateashowTruck()methodtodisplaythetruck’s capacity.
      2. Write a constructor for truck that initializes all properties.
      3. Implementthetruck classandupdatethemainmethodto createaTruckobjectand alsocreateanobjectforcarand bikesubclasses. Finallydisplay the details

**Program:**



**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| **1.** | **Semi colon (;)** | **Give the semi colon (;) in each line where it is required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| **brand: string speed: string** |
| **+Vehicle(String brand, int speed)**  **+Details(): void** |

|  |
| --- |
| **CARS** |
| **doors: int** |

|  |
| --- |
| **capacity: int** |
| **+ CARS (String brand, int speed, int doors, int capacity)**  **+cardetails(): void** |

|  |
| --- |
| **Bikes** |
| **gears: Boolean** |
| **+ Bikes(String brand, int speed, Boolean gears)**  **+bikedetails(): void** |

****

|  |
| --- |
| **Trucks** |
| **tons: int** |
| **+ Trucks(String brand,int speed,int tons)**  **+truckdetails(): void** |

**Important points:**

Multi-inheritance: It is one of the types of the inheritance where subclass 2 inherits subclass1 and subclass1 inherits superclass.

Here Vehicle is the super class or parent class and remaining cars, bikes, trucks are the subclasses or child classes

**Week 6:**

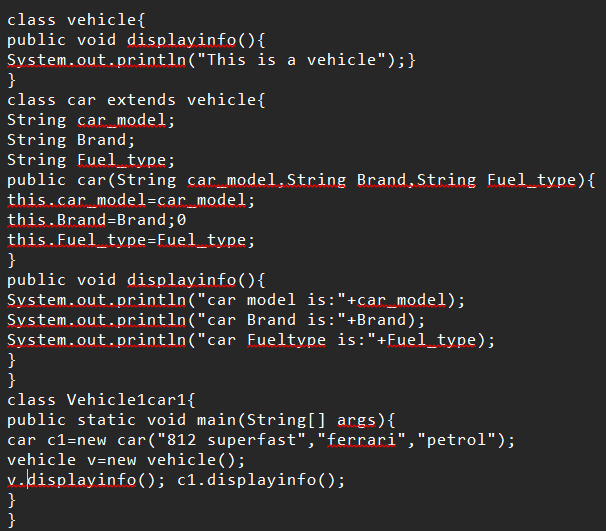
**Program 1:**

**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 a car, model, fuel type, and colour using the constructor

**Syntax:**

**Super class extends subclass**

Here extends is the main key word which represents the extending relation from parent class to child class.

**Program:**

Output:



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| 1. | Semi colon (;) | Give the semi colon (;) in each line where it is required |
| 2. | Syntax Error | Giving Capital ‘S’ in printing statements (System.out.println) |

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| **+displayinfo():void** |

|  |
| --- |
| **Car\_model:String Brand:String Fuel\_type:String** |
| **+ car(String car\_model,String Brand,String Fuel\_type)**  **+displayinfo(): void** |

**Important points:**

In order to do this, we have to use inheritance concept. Here we used the multi-inheritance concept**.**

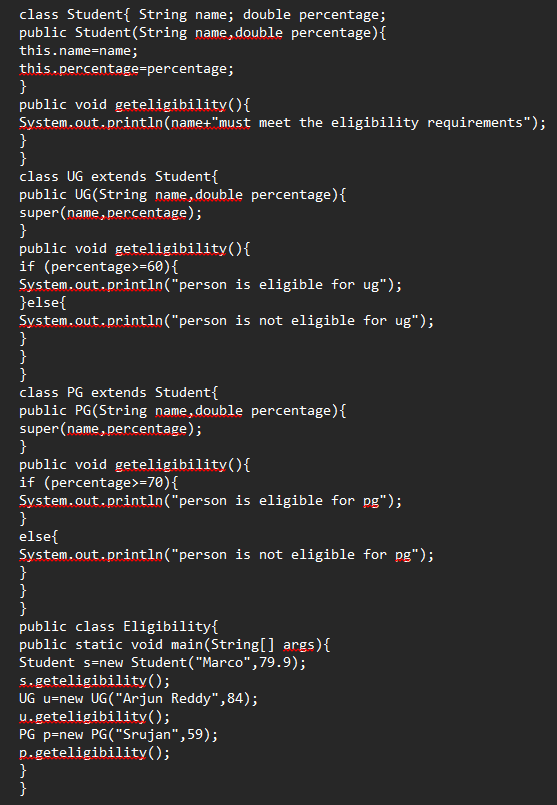
**Program 2:**

**Aim:** Create a Java program for the scenario.

A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student's percentage in their previous qualification.

1. UG admissions require aminimum of 60%
2. PG admissions require aminimum of 70%

**Program:**



**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| **1.** | **Semi colon (;)** | **Give the semi colon (;) in each line where it is required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

**Important points:**

Super keyword is used take the method,variable,constructor from the super class.

**Class diagram:**

**Student**

name: String

percentage:double

+Student(String name,double percentage)

+geteligibility():void

+UG(String name,double

UG

**Program-3:**

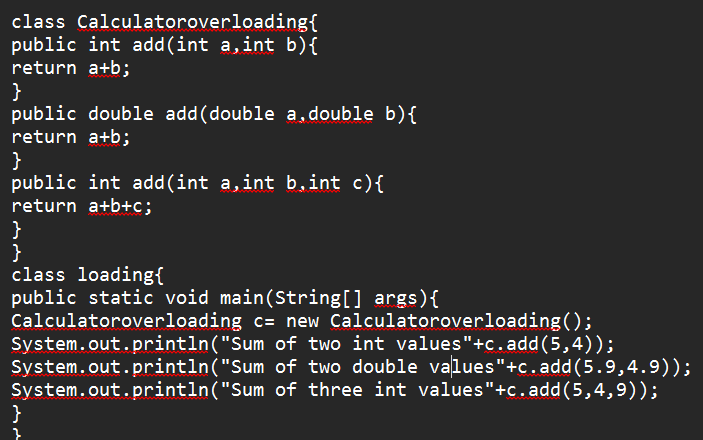
**Aim:** Write a Java Program to create a Calculator class with overloaded methods to perform addition: Take the integer values a and b from the user.

* 1. Addtwointegers
  2. Addtwodoubles
  3. Addthreeintegers

**Important points:**

We should carefully pass the double and integer and different types of input to an constructor when creating an object to access the different constructors based on the parameter.

**Program:**

****

**Output:**

****

**Class diagram:**

|  |
| --- |
| **Calculatoroverloading** |
| **+ add(int a,int b):int**  **+add(double a,double b):double**  **+ add(int a,int b,int c):int** |

**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| **1.** | **Semi colon (;)** | **Give the semi colon (;) in each line where it is**  **required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

**Program 4:**

**Aim:** Write a Java Program to create a shape class with a method calculateArea() that is overloaded for different shapes(e.g., Square, Rectangle ). Then create a subclass Circle that overrides the

calculateArea() method for a circle.

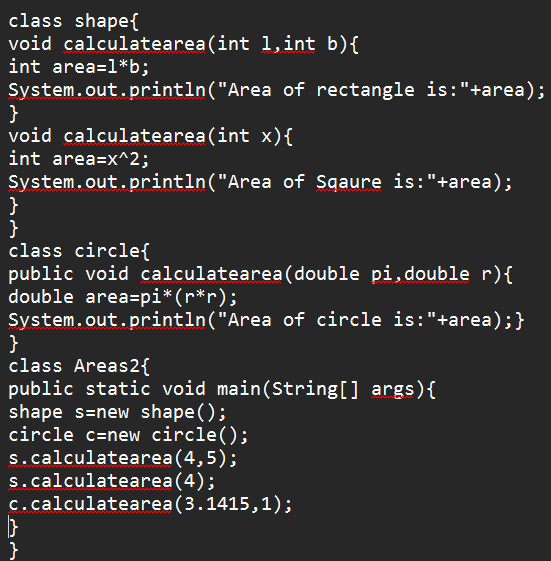
**Important points:**

In this program we use both method overloading and overriding to calculate area of different shapes**.**

**Class Diagram:**

|  |
| --- |
| **shape** |
| **+calculatearea(int l,intb):void**  **+calculatearea(int x):void** |

|  |
| --- |
| **circle** |
| **+void calculatearea(double pi,double r):void** |

**PROGRAM:**

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| **1.** | **Semi colon (;)** | **Give the semi colon (;) in each line where it is**  **required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

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

**Program:**

abstract class Animal{

    abstract void sound();

}

class Tiger extends Animal{

    void sound(){

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

        }

    }

class Lion extends Animal{

    void sound(){

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

        }

    }

class lab7\_1{

    public static void main(String[] args){

        System.out.println("G REVANTH / CSE B/ av.sc.u4cse24120");

        Animal a1=new Tiger();

        Animal a2=new Lion();

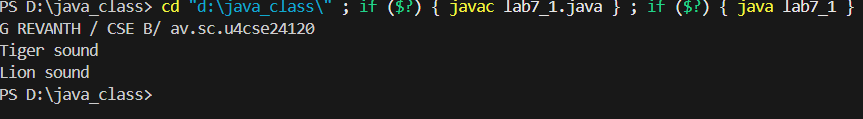
        a1.sound();

        a2.sound();

        }

}

**Output:**



**Class Diagram:**

**CLASS DIAGRAM:**

|  |
| --- |
| **Animal**  **+ sound(): void** |

|  |
| --- |
| **Lion**  **+ sound (): void** |

|  |
| --- |
| **Tiger**    **+ sound(): void** |

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code 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. |

**AIM:**Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

**CODE:**

abstract class Shape3D {

abstract double volume();

abstract double surfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

System.out.println("Sphere created with radius: " + radius);

}

@Override

double volume() {

return (4.0/3.0) \* 3.14 \* radius \* radius \* radius;

}

@Override

double surfaceArea() {

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

}

}

class Cylinder extends Shape3D {

double radius;

double height;

Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

System.out.println("Cylinder created with radius: " + radius + " and height: " + height);

}

@Override

double volume() {

return 3.14 \* radius \* radius \* height;

}

@Override

double surfaceArea() {

return (2 \* 3.14 \* radius \* height) + (2 \* 3.14 \* radius \* radius);

}

}

class ShapeTest {

public static void main(String[] args) {

System.out.println("G Revanth , CSE B, av.sc.u4cse24120");

Sphere s1 = new Sphere(5.0);

System.out.println("Sphere Volume: " + s1.volume());

System.out.println("Sphere Surface Area: " + s1.surfaceArea());

Cylinder c1 = new Cylinder(3.0, 7.0);

System.out.println("Cylinder Volume: " + c1.volume());

System.out.println("Cylinder Surface Area: " + c1.surfaceArea());

}

}

**Output:**

**CLASS DIAGRAM:**

|  |
| --- |
| **Shape3D**  **+calculatevolume(): double**  **+calculatesurfacearea(): double** |

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code 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. |

**Program 3:**

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

Create an abstract class named PatternPrinter with:an abstract method printPattern(int n)and a concrete method to display the pattern title

Implement two subclasses:

1. StarPattern – prints a right-angled triangle of stars

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.

Example Output for n = 5:

Star Pattern

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Number Pattern

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

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 + " ");

}

System.out.println();

}

}

}

public class Teststar {

public static void main(String[] args) {

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:



CLASS DAIGRAM:

PatternPrinter

- rows: int

+displayTitle()

+printPattern()

StarPattern

+printPattern()

NumberPattern

+printPattern()

**ERROR TABLE:**

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

Code:

class Shapes {

    public double area() {

        return 0;

    }

}

class Triangle extends Shapes {

    private double base;

    private double height;

    public Triangle(double base, double height) {

        this.base = base;

        this.height = height;

    }

    @Override

    public double area() {

        return 0.5 \* base \* height;

    }

}

class Circle extends Shapes {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    @Override

    public double area() {

        return Math.PI \* radius \* radius;

    }

}

class Rectangle extends Shapes {

    private double length;

    private double width;

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    @Override

    public double area() {

        return length \* width;

    }

}

public class ShapeArea {

    public static void main(String[] args) {

        Shapes triangle = new Triangle(2, 5);

        Shapes circle = new Circle(4);

        Shapes rectangle = new Rectangle(6, 9);

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

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

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

    }

}

CLASS DIAGRAM:

|  |
| --- |
| Shapes |
| +area(): double |

|  |
| --- |
| Triangle |
| -base  -height |
| +area():double |

|  |
| --- |
| Circle |
| -radius |

|  |
| --- |
| Rectangle |
| -length  -width |
| +area():double |

Error table:

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

***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 ShapeArea 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.

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 void 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 TestSports {

    public static void main(String[] args) {

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

        football.play();

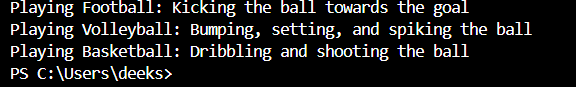
        volleyball.play();

        basketball.play();

    }

}

Output:



Class diagram:

|  |
| --- |
| Interface:playable |
| +play():void |

|  |
| --- |
| FootBall |
| +play():void |

|  |
| --- |
| VolleyBall |
| +play():void |

|  |
| --- |
| BasketBall |
| +play():void |

**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

**PROGRAM-3:**

**AIM:**Write a java program to implements login System using interfaces.

**CODE:**

interface LoginSystem {

    boolean login(String id, String pass);

}

class University\_portal implements LoginSystem {

    @Override

    public boolean login(String id, String pass) {

        if (id.equals("Student123") && pass.equals("pass02")) {

            System.out.println("Login successful");

            return true;

        } else {

            System.out.println("Invalid credentials");

            return false;

        }

    }

    public static void main(String[] args) {

        University\_portal p1 = new University\_portal();

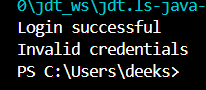
        p1.login("Student123", "pass02");

        p1.login("Student123", "wrongpass");

    }

}

Output:



CLASS DAIGRAM:

interface :Login system

+ login(String,String): boolean

University\_portal

+ login(StringString): boolean

ERROR TABLE:

|  |  |  |
| --- | --- | --- |
| CODE ERROR   1. Saving file with the interface name is error 2. Removing public will leads to error  |  | | --- | |  | | ERROR RECTIFICATION   1. save with implement name. 2. add public static void main   (String[]args) |

**IMPORTANT POINTS:**

***Interface Implementation:***

University\_potral correctly implements Login System interface

Uses Override annotation for the login() method

***Authentication Logic***:

Hardcoded credentials: id="Student123", password="pass02"

Returns boolean and prints appropriate message.

***Main Method***:

Demonstrates both successful and failed login attempts

Includes student information print statement

***Polymorphism***:

Could create Login System Portal=new University\_portal();

Demonstrates interface-based programming