**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**NAME :G. Manideep**

**Verified By Roll No: AV.SC.U4CSE24108**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Experiment Name** | **Page No.** | **Remarks** | **Signature** |
|  | **WEEK-1** |  |  |  |
| 1 | Installation Process of JDK | 3 |  |  |
| 2 | Simple Java Program for printing basic details of student | 5 |  |  |
|  | **WEEK-2** |  |  |  |
| 1 | Write a Java Program to find the factorial of a number | 7 |  |  |
| 2 | Write a Java Program to find the Fibonacci Series of given length | 9 |  |  |
| 3 | Write a Java Program to find the temperature from Celsius to Fahrenheit | 11 |  |  |
| 4 | Write a Java Program to find the Simple Interest | 12 |  |  |
| 5 | Write a Java Program to find the area of triangle using heron’s formula | 13 |  |  |
| 6 | Write a Java Program to find the area of rectangle | 15 |  |  |
|  | WEEK-3 |  |  |  |
| 1 | Write a java program with the following instructions:  a. Create class with name car.  b. Create 4 attributes named car color, car brand, fuel type, mileage.  c. Create 3 methods named start(), stop(), service().  d. Create 3 objects C1, C2, C3.  e. Create a constructor with parameters with car color, car brand, fuel type, mileage. | 18 |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Create a class named bank account with methods deposit and withdraw. Where the deposit method should accepts a parameter and when this method is called the deposited amount should be added to current balance. In addition to that when a withdraw method is called it has to verify whether the withdraw amount is less than the current balance. If not display message saying that there are insufficient funds. Use the constructor to display the details of the customer (Name, Account number, IFSC code, Branch). Also create two customer objects C1, C2. | 21 |  |  |
|  | WEEK-4 |  |  |  |
| 1 | Write a Java program with a class named book. The class should contain various attributes such as title, author, year of publication, and price. It should also contain a constructor with parameters that include title, author, year of publication, and price. Create a method that displays details of the book(Display the details of 3 books that is create 3 objects and display their details) | 26 |  |  |
| 2 | Create a java program with a Class named "my class" with a static variable 'count' of int type static and initialized to zero and a constant variable 'pi' of type double initialized to 3.1415 as attributes of that class now define a constructor of my class that increments the count variable each time an object of my class is created and finally prints the final values of count and pi variables. | 30 |  |  |
|  | **WEEK-5** |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output. | 33 |  |  |
| 2 | 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.  i. Cars should have an additional property: number of doors, Seating capacity.  ii. Bikes should have a property indicating whether they have gears or not.  iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.  iv. Each class should have a constructor.  Questions:  1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.  2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?  a. Truck should include and additional property capacity (in tons).  b. Create a showTruck() 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 subclasses. Finally display the details. | 35 |  |  |
|  | **WEEK-6** |  |  |  |
| 1 | Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and colour using the constructor | 39 |  |  |
| 2 | 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.  i) UG admissions require a minimum of 60%  ii) PG admissions require a minimum of 70% | 41 |  |  |
| 3 | 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.  i) Add two integers  ii) Add two doubles  iii) Add three integer | 44 |  |  |
| 4 | 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. | 46 |  |  |
|  |  |  |  |  |
|  | **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 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. |  |  |  |
|  |  |  |  |  |
| 3 | write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method print pattern (int n) and a concrete method to display the pattern tittle. Implement two sub  class :  1.star pattern -prints a right-angled triangle of stars(\*)  2.number pattern-prints a right angled triangle of increasing numbers. In the main () method,create objects of both subclasses and print the pattern for a given number of rows. |  |  |  |
|  |  |  |  |  |
|  | WEEk-8 |  |  |  |
| 1 | Write a java program to create an interface shape with getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the shape interface implement the getPerimeter() method for each of the three classes. |  |  |  |
|  |  |  |  |  |
| 2 | Write a java program to create an interface playable with a method play that takes no arguments and returns play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements the playable interface and override the play method to play the respective sports. |  |  |  |
|  |  |  |  |  |
| 3 | Write a java program to implement a login system using interfaces |  |  |  |
|  |  |  |  |  |
|  | **WEEK-8** |  |  |  |
| **1** | Write a java program to create an interface shape with getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the shape interface implement the getPerimeter() method for each of the three classes. |  |  |  |
|  |  |  |  |  |
| **2** | Write a java program to create an interface playable with a method play that takes no arguments and returns play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements the playable interface and override the play method to play the respective sports. |  |  |  |
|  |  |  |  |  |
| 3 | Write a java program to implement a login system using interfaces |  |  |  |
|  |  |  |  |  |
|  | WEEK-9 |  |  |  |
| 1 | write a java program to create a method that takes a integer as a parameter and throws an exception if the public class even number |  |  |  |
|  |  |  |  |  |
| 2 | 2)write a java program to create a method that reads a file and throws an exception if the file is not found  imp points  buffered reader |  |  |  |
|  |  |  |  |  |
| 3 | write a java program to handle arthematic expression try catch and finally |  |  |  |
|  |  |  |  |  |
| 4 | write a java program to simulate a university system using inner classes  create an outer class named "university" with a variable universityName  inside it two non static inner classes:  1.department with variable like deptName and deptCode and a method to display department details  2.student with variables like student name and roll number and a method to display  3.create an object for each class and call their methods to display their details along with universityname  ex: public class university{  string universityname="amrita Vishwa Vidyapeetham" |  |  |  |
|  |  |  |  |  |

**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.

A screenshot of a web page

Description automatically generated

1. **Set Environment Variables (Windows):**
   * Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then  
     click on **Environment Variables**.
   * Click **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.



**2) Simple Java Program for printing Name, Class, Roll No, of a Student**

Write your code in Notepad and execute in cmd prompt

**CODE:**

class Main

{

public static void main(String[] args)

{

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

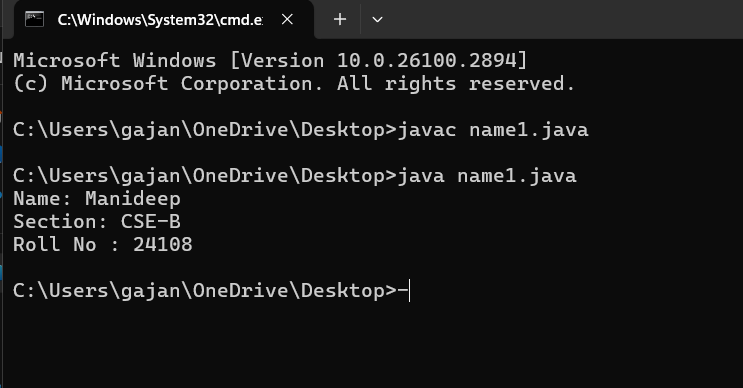
System.out.println("Section: CSE-B");

System.out.println("Roll No : 24108");

}

}

**Output:**



**WEEK-2**

1. **Simple Java Program for finding simple interest by taking input from User ?**

**Code:**



**Output:**

**A computer screen shot of a program

AI-generated content may be incorrect.**

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

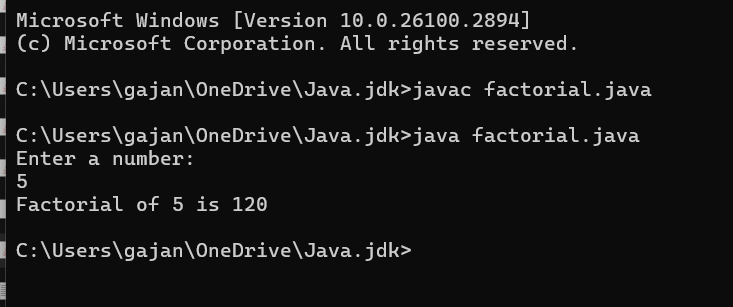
**2.Write a simple program to calculate factorial of a number and read the input from user ?**

**code:**

A computer screen shot of a code

AI-generated content may be incorrect.

**Output:**



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

**3.Write a program to to calculate the fibonacii sequence and take the input from user ?**

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Output:**

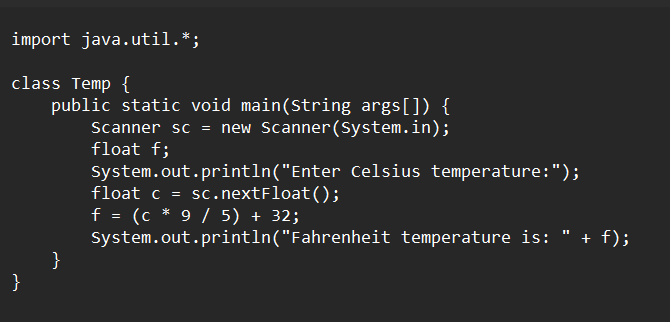
A computer screen shot of a program

AI-generated content may be incorrect.

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

**4).Write a java program to convert temperature from Fahrenheit to celsius and take the input from user ?**

**Code:**

****

**Output:**

A computer screen with white text

AI-generated content may be incorrect.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Syntax error** | **Missing ”** | **“ is added** |
| **2** | **Missing import error** | **Util package missing** | **Util package added** |

**5).Write a simple program to find the area of rectangle and take the input from user ?**

**Code:**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**Output:**

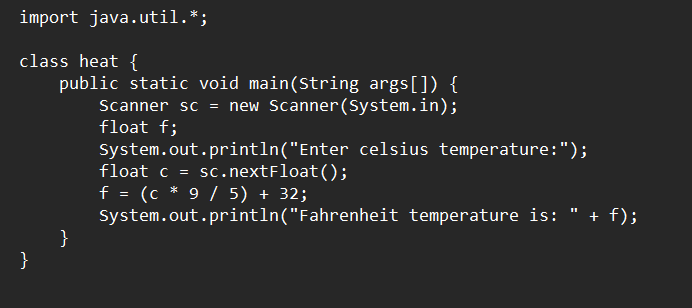
A computer screen with white text

AI-generated content may be incorrect.

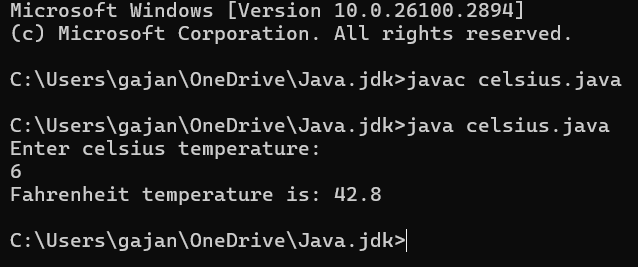
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Semi colon missing** | **Semi colon added** |

**6).Write a java program to convert temperature from Celsius to Fahrenheit**

**Code**



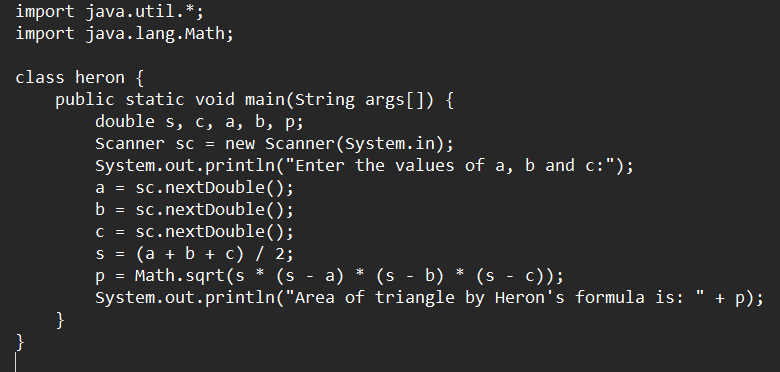
**Output:**

****

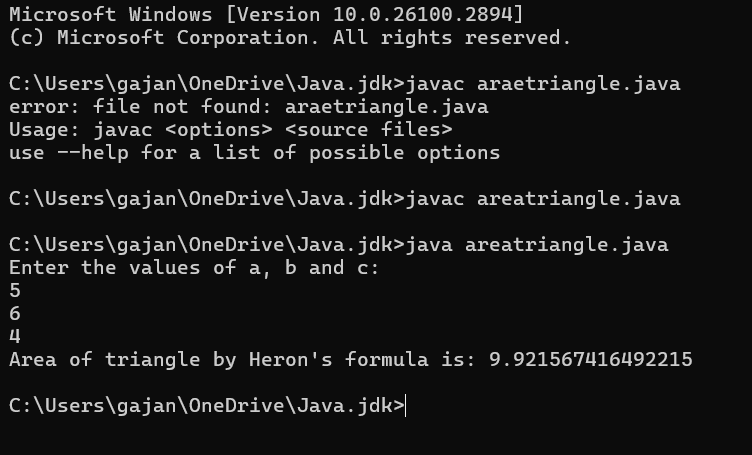
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Runtime error** | **Incorrect path selection** | **Correct path added** |
| **2** | **Logical error** | **Incorrect logic** | **Correct logic** |

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

**Code:**



**Output:**

****

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

**WEEK-3**

**Aim:**

**1) create a java program with following instructions**

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

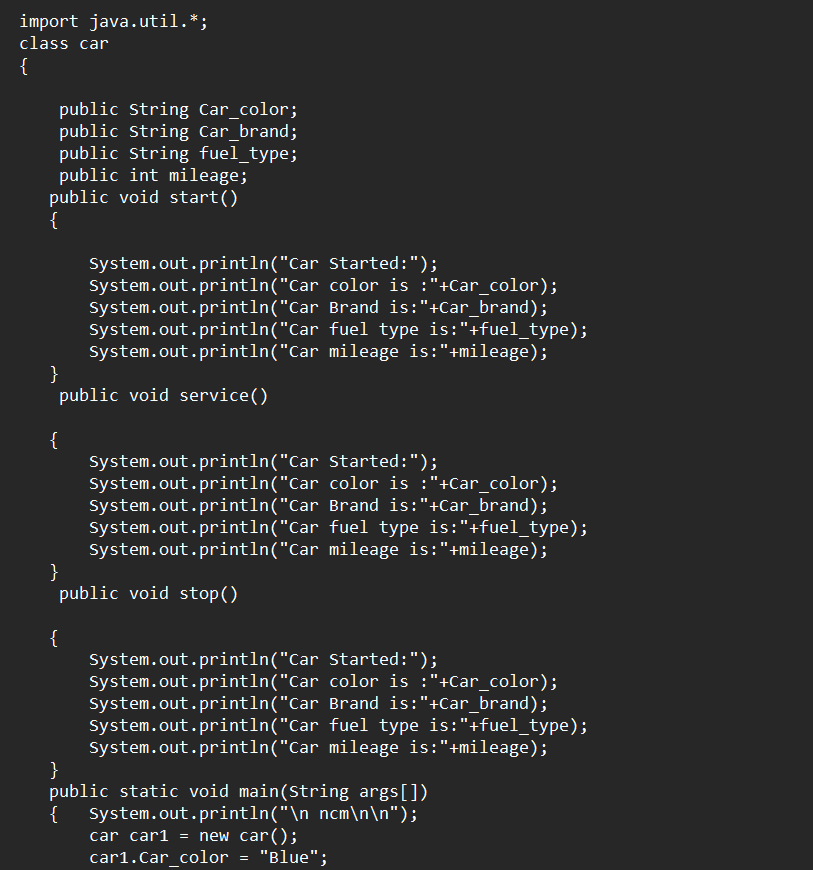
**b. Create four attributes named car\_color ,Car\_brand,fuel\_type,mileage**

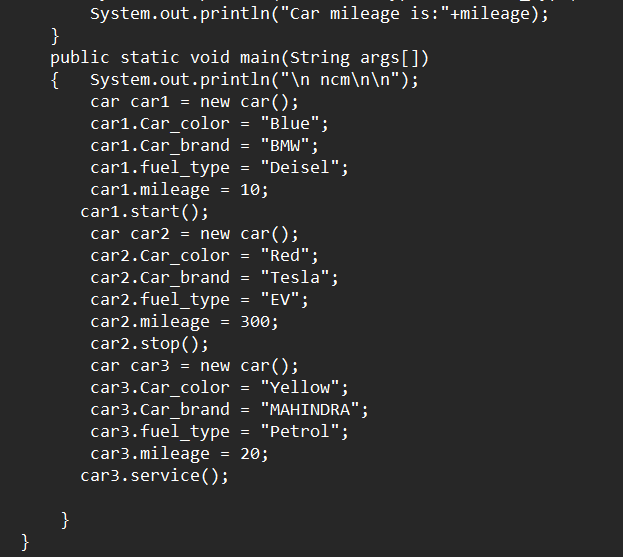
**c. Create three methods named start(), stop(). Service()**

**d. Create three objects named car1,car2 and car3**

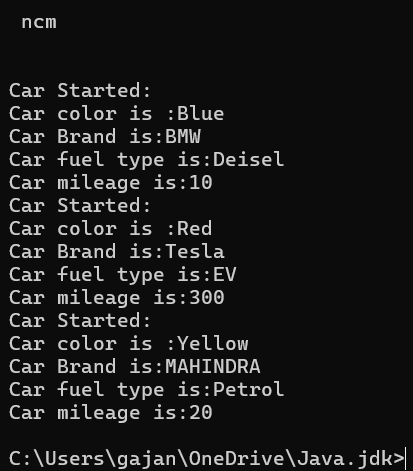
**2). To create a class bankAccount with methods deposit() and withdrawl ?**

**Code:**

****

****

**Output:**

****

**Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | **Name error** | **Undeclared variable** | **Variable declared** |
| **2** | **Run time error** | **Incorrect path decloration** | **Correct path declared** |
| **3** | **Syntax error** | **Semi colon missed** | **Semi colon added** |
| **4** | **Syntax error** | **Missing “** | **Added “** |
| **5** | **Syntax error** | **Int missing** | **Int added** |

**Important points:**

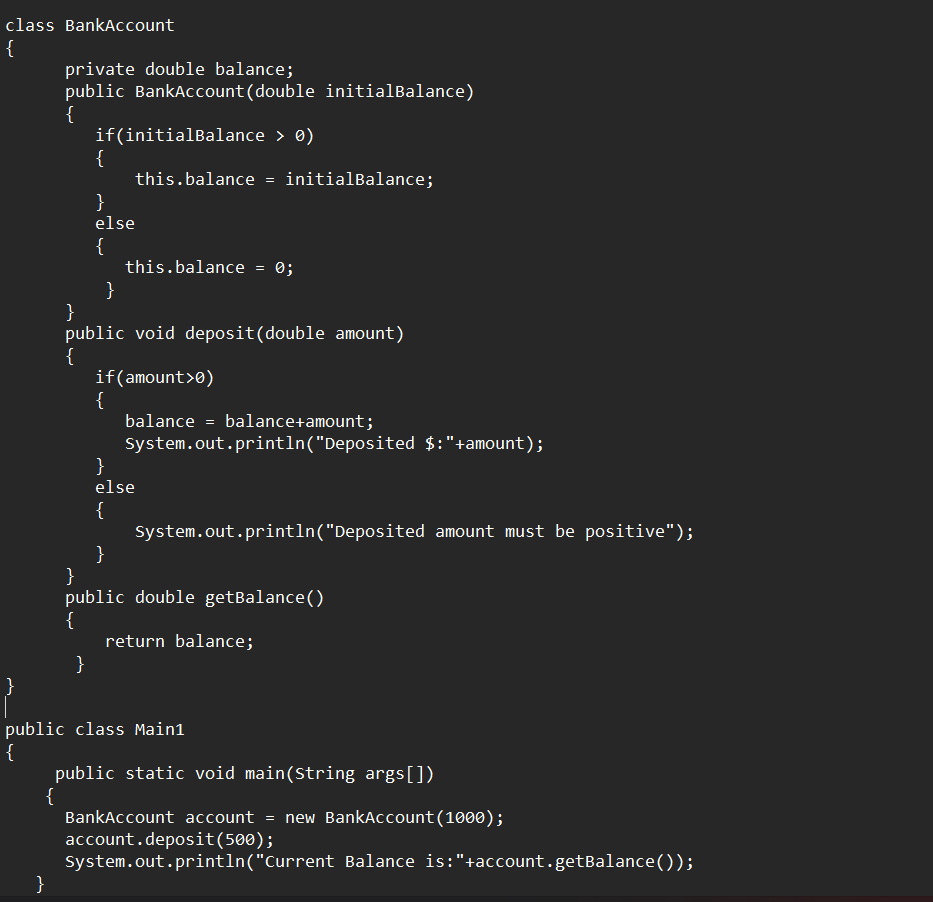
* Variable name mismatch: The variable car\_Color in the code should be car\_color
* Incorrect variable name: car1.car\_color is used when the actual variable is car1.car\_Color, which will cause an error due to case sensitivity.
* Missing Semicolon: Forgetting to add a semicolon at the end of a statement will cause a compilation error.

**CLASS DIAGRAM-**

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

**2).To create a class bankAccount with methods deposit() and withdrawl ?**

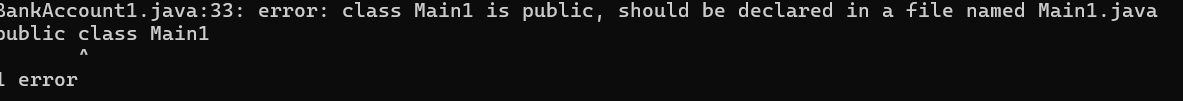
**code:**



**Output:**



**Errors:**



**Important points:**

* The balance should be a float or double to handle decimal values correctly, but it's declared as an int.
* Incorrect deposit method signature: The method DEPOSIT () has an incorrect return type int(), while it should be void since it doesn't need to return any value.
* Fixed the return type of deposit: Changed from int to void, as the method does not need to return anything

**CLASS DIAGRAM-**

A close-up of a bank account

AI-generated content may be incorrect.

**WEEK-4**

**1. Write a Java program with a class named Book**

**a) a class should contain various attributes such as title, author, and year of publication.**

**b) it should also contain a constructor with parameters which initialize the title, author, and year of publication.**

**c)create a method which displays the details of the book title, author, year of publication**

**Display the details of two books.**

A screen shot of a computer program

AI-generated content may be incorrect.**CODE:**

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. |  |  |
| 2. |  |  |

**Important points:**

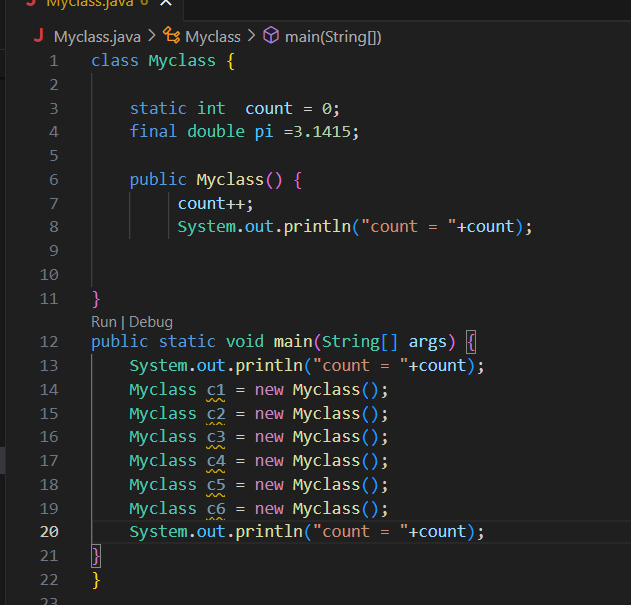
* While defining two classes for a code, we must be sure that we save both the classes in separate files.
* While defining a method we should also define a function to call that method.

**CLASS DIAGRAM-**

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

**2). Create a Java program with a class named ‘MyClass’ with a static variable count of int type, initialized to zero and a constant variable ‘pi’ or type double initialized to 3.14 as attributes of the class. Now define a constructor for “MyClass” that increments the count variable each time an object of MyClass is created. Finally, print the final values of the count and pi variables. Create three objects and a constructor.**

**CODE:**



**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| **1.** |  |  |
| **2.** | **error: can't find primary (String []) method in class: SimpleInterestCalculator** | **Should close the string brackets []** |

**Important points:**

* **We must declare the initial value of the variable before declaring the final one.**
* **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**

**Task-1**

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

**Important points:**

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

Syntax:

**Super class extends subclass**

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

**Program :**

import java.util.Scanner;

import jdk.jshell.execution.Util;

public class Calci {

    public void addition(int a , int b){

        System.out.println("addition :"+(a+b));

    }

    public  void diff(int a , int b){

        System.out.println("Substraction :"+(a - b));

    }

} //end

class  Calci2 extends Calci{

    public void multi(int a , int b){

        System.out.println("multiplication :"+(a\*b));

    }

} // end

class Calci3 extends Calci2{

    public void divid(int a , int b){

if(b == 0){

        System.out.println("b should not be zero");     }

    else{

        System.out.println("Division :" +(a/b));

    }

    }

}//end

class Multilevel {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        Calci c1 = new Calci();

        c1.addition(80, 92);

        c1.diff(80, 92);

        Calci2 c2 = new Calci2();

        c2.multi(8, 9);

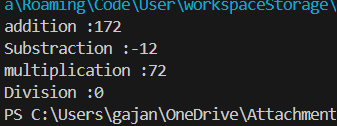
        Calci3 c3 = new Calci3();

        c3.divid(80, 92);

    }

}

**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 :**

|  |
| --- |
| **Ultimate** |
| +division (int a int b): void |

|  |
| --- |
| **Simple** |
| +addition (int a int b): void  +subtraction (int a int b): void |

|  |
| --- |
| **Advanced** |
| +multiplication (int a int b): void |

**Task-2**

**Aim:**

Task-2: 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.

i. Cars should have an additional property: number of doors, Seating capacity.

ii. Bikes should have a property indicating whether they have gears or not.

iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.

iv. Each class should have a constructor.

Questions:

1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.

2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?

a. Truck should include and additional property capacity (in tons).

b. Create a showTruck() 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 subclasses. Finally display the details.

**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.

**Program :**

class Vehicle{

        String brand;

        int speed;

        Vehicle(String brand,int speed){

            this.brand=brand;

            this.speed=speed;

        }

        void Details(){

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

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

        }

    }//End of super class

    class CARS extends Vehicle{

        int doors;

        int capacity;

        public CARS(String brand,int speed,int doors,int capacity){

            super(brand, speed);

            this.doors=doors;

            this.capacity=capacity;

        }

        void cardetails(){

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

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

        }

    }//End of car sub-class

    class Bikes extends Vehicle{

        Boolean gears;

        Bikes(String brand,int speed,Boolean gears){

            super(brand, speed);

            this.gears=gears;

        }

        void bikedetails(){

            if (gears==true)

            System.out.println("This bike has gears.");

            else

            System.out.println("This bike does not have gear system.");

        }

    }//End of bike sub-class

    class Trucks extends Vehicle{

        int tons;

        Trucks(String brand,int speed,int tons){

            super(brand, speed);

            this.tons=tons;

        }

        void truckdetails(){

            System.out.println("The capacity of truck is: "+tons);

        }

    }//End of truck sub-class

    class Rent{

        public static void main(String[] args){

            CARS c=new CARS("Mahindra",180,5,5);

            c.cardetails();

            c.Details();

            Bikes b=new Bikes("BMW",80,true);

            b.bikedetails();

            b.Details();

            Trucks t=new Trucks("TATA",100,1);

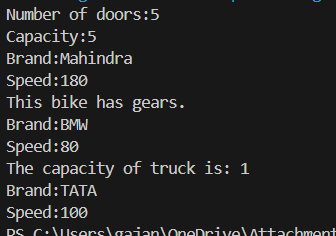
            t.truckdetails();

            t.Details();

        }

    }

**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 |

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

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

**WEEK-6**

1) **Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and color using the constructor**

- Write your code in VS CODE and execute

**- Important Points:**

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**CODE:**class Vehicle {

String Brand;

String model;

String fuel;

String color;

int capacity;

Vehicle(String Brand, String model, String fuel, int capacity, String color) {

this.Brand = Brand;

this.model = model;

this.fuel = fuel;

this.capacity = capacity;

this.color = color;

}

void displayInfo(String Brand, String model, String fuel, int capacity, String color) {

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

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

System.out.println("Model: " + model);

System.out.println("Fuel: " + fuel);

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

System.out.println("Color: " + color);

}

}

class Car extends Vehicle {

Car(String Brand, String model, String fuel, int capacity, String color) {

super(Brand, model, fuel, capacity, color);

}

void displayInfo() {

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

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

System.out.println("Model: " + model);

System.out.println("Fuel: " + fuel);

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

System.out.println("Color: " + color);

}

}

class Week6\_1 {

public static void main(String[] args) {

// Creating an instance of Car

Car car1 = new Car("BMW", "X5", "Petrol", 6, "Red");

car1.displayInfo(); // Display car details

}

}

**OUTPUT:**

A screen shot of a black background

AI-generated content may be incorrect.

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**2) 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.**

**i) UG admissions require a minimum of 60%**

**ii) PG admissions require a minimum of 70%**

- Write your code in VS CODE and execute

**- Important Points:**

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**CODE:**

class College{

String name;

int percentage;

void geteligibility(String name,int percentage){

this.name=name;

this.percentage=percentage;

}

}

class UG extends College{

void geteligibility(String name,int percentage){

if (percentage>=60){

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

}

else{

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

}

}

}

class PG extends College{

void geteligibility(String name,int percentage){

if (percentage>=70){

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

}

else{

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

}

}

}

class week6\_2{

public static void main(String[] args){

UG ug=new UG();

ug.geteligibility("Person-1",40);

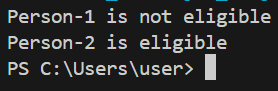
PG pg=new PG();

pg.geteligibility("Person-2",80);

}

}

**OUTPUT:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**3) 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.**

**i) Add two integers**

**ii) Add two doubles**

**iii) Add three integers**

- Write your code in VS CODE and execute

**- Important Points:**

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**CODE:**

class Calc{

public int add(int a,int b){

return a+b;

}

public double add(double a,double b){

return a+b;

}

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

return a+b+c;

}

}

class week6\_3{

public static void main(String[] args){

Calc C1=new Calc();

System.out.println("Sum of 6 and 9 is: "+C1.add(6,9));

System.out.println("Sum of 7.6 and 8.6 is: "+C1.add(7.6,8.6));

System.out.println("Sum of 2,4 and 6 is: "+C1.add(2,4,6));

}

}

**OUTPUT:**

A black background with white text

AI-generated content may be incorrect.

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**4) 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.**

- Write your code in VS CODE and execute

**- Important Points:**

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**CODE:**class Shape {

double calculateArea(double side) {

return side \* side;

}

double calculateArea(double width, double height) {

return width \* height;

}

}

class Circle extends Shape {

double calculateArea(double radius) {

return 3.14 \* radius \* radius;

}

}

class Week6\_4 {

public static void main(String[] args) {

Shape S1 = new Shape();

System.out.println("Area of square: " + S1.calculateArea(5));

System.out.println("Area of rectangle: " + S1.calculateArea(2, 5));

Circle C1 = new Circle();

System.out.println("Area of circle: " + C1.calculateArea(3));

}

}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**WEEK-7**

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

**Important points :**

1. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2. An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

3. Here we used hierarchy inheritance concept means every sub class extends super class.

**Program:**

abstract class animal{

    abstract void sound();

}

class lion extends animal{

    void sound(){

        System.out.println("lion roars");

}

}

class tiger extends animal{

    void sound(){

        System.out.println("tiger growls");

    }

}

class ABanimal{

    public static void main(String[] args) {

        System.out.println("NAME : Manideep");

        System.out.println("ROLL NO : 24108");

        System.out.println("SEC: CSE-B");

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

        lion l = new lion();

        l.sound();

        tiger t = new tiger();

        t.sound();

    }

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO |  |  |
| 1 | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| 2 | Method | We need to provide return type to the method. |
| 3 | Data type | As per need provide data type 4 Abstract method |
| 4 | Abstract method | Implementation in subclass only |

1. 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.

**Program:**

abstract class Shape3D {

    public abstract double calculateVolume();

    public abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    private double radius;

    public Sphere(double radius) {

        this.radius = radius;

    }

    public double calculateVolume() {

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

    }

    public double calculateSurfaceArea() {

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

    }

}

class Cube extends Shape3D {

    private double side;

    public Cube(double side) {

        this.side = side;

    }

    public double calculateVolume() {

        return Math.pow(side, 3);

    }

    public double calculateSurfaceArea() {

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

    }}

public class  Shapees{

    public static void main(String[] args) {

        System.out.println("MAnideep");

        System.out.println( "24108");

        System.out.println( "Section B");

        Shape3D sphere = new Sphere(5);

        Shape3D cube = new Cube(4);

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

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

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

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

    }

}

**Output:**

**A computer screen shot of numbers and letters

AI-generated content may be incorrect.**

**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| 2 | Data type | As per need provide data type 3 Abstract method Implementation in subclass only |
| 3 | Abstract method | Implementation in subclass only |

**Important points :**

1. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2. An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

3. Understanding the calling of constructor.

1. **Aim :** **write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method print pattern (int n) and a concrete method to display the pattern tittle. Implement two sub**

**class :**

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

**2.number pattern-prints a right angled triangle of increasing numbers. In the main () method,create objects of both subclasses and print the pattern for a given number of rows.**

**Program:**

abstract class PatternPrinter{

    public abstract void printPattern(int n);

    public void printTitle(String title) {

        System.out.println(title);

    }

}

class StarPattern extends PatternPrinter {

    public void printPattern(int n){

        for (int i=1;i<=n;i++) {

            for (int j=1;j<=i;j++) {

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

            }

            System.out.println();

        }

    }

}

class NumberPattern extends PatternPrinter {

    public void printPattern(int n) {

        for (int i=1;i<n;i++) {

            for (int j=1;j<=i;j++) {

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

            }

            System.out.println();

        }

    }

}

public class pattern {

    public static void main(String[] args) {

System.out.println("Manideep");

System.out.println("24108");

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

        int rows=5;

        PatternPrinter starPattern = new StarPattern();

        PatternPrinter numberPattern = new NumberPattern();

        starPattern.printTitle("Star Pattern:");

        starPattern.printPattern(rows);

        System.out.println();

        numberPattern.printTitle("Number Pattern:");

        numberPattern.printPattern(rows);

    }

}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| 2 | Data type | As per need provide data type |
| 3 | Syntax | in for Initializing value and condition should be correct |
| 4 | overridding | Same method name |

**Important points :**

1. Here we used nested for loop concept the block of code is executed until the condition is false.

2. Here the logic very important .

3. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

**WEEK-8**

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

**Program:**

interface Shapee {

    float pi = 3.14f;

    int getperimeter();

}

class Rectangle implements Shapee {

    int length;

    int breadth;

    public Rectangle(int length, int breadth) {

        this.length = length;

        this.breadth = breadth;

    }

    public int getperimeter() {

        return 2 \* (length + breadth);

    }

}

class Circle implements Shapee {

    private int radius;

    public Circle(int radius) {

    this.radius = radius;   }

    public int getperimeter() {

        return (int) (2 \* pi \* radius);

    }}

class Triangle implements Shapee {

    private int side1;

    private int side2;

    private int side3;

    public Triangle(int side1, int side2, int side3) {

        this.side1 = side1;

        this.side2 = side2;

        this.side3 = side3;

    }

    public int getperimeter() {

        return (side1 + side2 + side3);

    }}

class InShape {

    public static void main(String[] args) {

    System.out.println("Manideep");

    System.out.println("24108");

    System.out.println("CSE-B");

        Rectangle r = new Rectangle(5, 6);

        System.out.println("Rectangle perimeter: " + r.getperimeter());

        Circle c = new Circle(7);

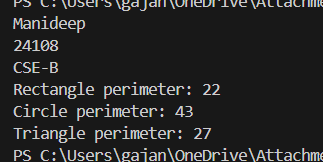
        System.out.println("Circle perimeter: " + c.getperimeter());

        Triangle t = new Triangle(8, 9, 10);

        System.out.println("Triangle perimeter: " + t.getperimeter());

    }}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Method not implemented | |  | | --- | |  |  |  | | --- | | Ensure all classes implement the getperimeter() method from the interface. | |
| 2 | |  | | --- | | Incorrect method name | | |  | | --- | | Use the exact method name getperimeter() as declared in the interface. | |
| 3 | Missing cast in Circle calculation | Cast the result to int: (int)(2 \* pi \* radius). |
| 4 | Access modifier confusion | |  | | --- | |  |  |  | | --- | | Clearly define public, private, or default where appropriate. | |

**Important Points:**

1. Interfaces can have constants: pi in Shapee is implicitly public static final.
2. All classes implement the same interface, so they must define the getperimeter() method.
3. Casting is done in Circle: (int)(2 \* pi \* radius) — this truncates the result to an integer.
4. Access Modifiers: Notice mixed use of public/default (e.g., length is package-private in Rectangle)

**2.Write a java program to create an interface playable with a method play that takes no arguments and returns play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements the playable interface and override the play method to play the respective sports.**

**Program:**

public interface playable {

    void play();

}

class Football implements playable {

    public void play() {

        System.out.println("Football is being played...");

}}

class Vollyball implements playable {

    public void play() {

        System.out.println("Vollyball is being played...");

    }}

class Basketball implements playable {

    public void play() {

        System.out.println("Basketball is being played...");

    }}

class played {

    public static void main(String[] args) {

  System.out.println("My name is : Manideep");

  System.out.println("My roll.no : 108");

  System.out.println("My class-sec : CSE-B");

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

        Football fot = new Football();

        fot.play();

        Vollyball voly = new Vollyball();

        voly.play();

        Basketball bask = new Basketball();

        bask.play();

}}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | |  | | --- | | Interface not fully implemented | | |  | | --- | | Implement the play() method in all classes. |   . |
| 2 | |  | | --- | | Class/interface name mismatch |  |  | | --- | |  | | Make sure class and interface names match exactly (e.g., intplayable). |
| 3 | |  | | --- | | Method name spelling error |  |  | | --- | |  | | Use correct syntax: System.out.println("...");. |
| 4 | |  | | --- | | Object reference typo |  |  | | --- | | . | | Maintain consistent spelling in class names and object references |

**Important Points:**

1. Polymorphism is possible with intplayable but not used here (all objects are called separately).
2. Each class provides its own implementation of play().
3. Could be enhanced by using an array or list of intplayable objects and a loop.

**3.Write a java program to implement a login system using interfaces**

**Program:**

interface loginSystem{

    public boolean  login(String username,String password);

}

class Universitylogin implements loginSystem {

    public boolean login(String username, String password) {

        if(username == "ashish1" && password == "cse1111") {

            System.out.println("Successfully logged in..");

            return true;

    }

    else {

        System.out.println("Invalid username or password");

        return false;

    }

}}

class login{

    public static void main(String[] args) {

System.out.println("Manideep");

System.out.println("24108");

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

        Universitylogin ul = new Universitylogin();

        ul.login("ashish1", "cse1111");

        ul.login("ashih","ashish01");

    }}

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | |  | | --- | | String comparison using == |  |  | | --- | | Use | | username.equals("ashish1") instead of username == "ashish1". |
| 2 | |  | | --- | | Missing return statement |  |  | | --- | |  | | Ensure login() returns true or false as required. |
| 3 | |  | | --- | | Poor class naming convention |  |  | | --- | |  | | Rename class login to Login (PascalCase) to follow Java conventions. |
| 4 | |  | | --- | | Incorrect method call in main |  |  | | --- | |  | | Call ul.login("username", "password") correctly with matching arguments. |

**Important Points:**

1. Use .equals() for string comparison in Java to compare values.
2. Interface method is correctly implemented by Universitylogin.
3. The method prints a success or failure message based on input.
4. Proper boolean return allows flexible control in further development.

**WEEK-9**

**1)write a java program to create a method that takes a integer as a parameter and throws an exception if the public class even number**

**Program:**

public class Exceptionnumber {

    public static void checknum(int num) throws Exception {

        if (num%2 == 0){

            System.out.println("number cannont be even" +"--"+  num+"--"  + "so try using another number");

        }

        else{

            System.out.println("given number is checked"+num+"successfully exicuted");

        }}

    public static void main(String[] args) {

        Exceptionnumber en = new Exceptionnumber();

        int num = 5;

        try {

            checknum(5);

        }

        catch (Exception e) {

            System.out.println("number cannot be cheked becuse it is an even"+ num +"number"+e.getMessage());

        }

        }}

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**Error:**

**Class Diagram:**

|  |
| --- |
| EvenNumberExceptionDemo |
| + checkOddNumber(int number) : void  + main(String[] args) : void |

|  |
| --- |
| EvenNumberException  (extends Exception) |
| + EvenNumberException(String msg) |

**ERROR TABLE:**

|  |  |
| --- | --- |
| CODE ERROR  1)Unhandled exception type EvenNumberException  2) Syntax Error  3) Compilation Error | ERROR RECTIFICATION  1) If you call checkOddNumber() without using try-catch or without declaring throws.  2)If missing curly braces {} or wrong method syntax  3)If constructor of EvenNumberException is missing or incorrectly defined. |

**IMPORTANT POINTS:**

1) Created a custom exception by extending the Exception class.

2) Used throw keyword to manually throw the custom exception if the number is even.

3)Handled the exception using a try-catch block inside main() method.

4) Demonstrates user-defined exception handling.

5) Shows clear separation of concerns: checking number and exception message.

**2)write a java program to create a method that reads a file and throws an exception if the file is not found**

**imp points buffered reader**

**program:**

import java.io.\*;

public class File {

    public static void main(String[] args) {

        System.out.println("Manideep,24108,CSE-B");

        try {

            BufferedReader reader = new BufferedReader(

                new FileReader("C:/Users/deeks/OneDrive/Attachments/Documents/Desktop/hahaha.txt")

            );

            String line;

            while ((line = reader.readLine()) != null) {

                System.out.println(line);

            }

            reader.close();

        } catch (FileNotFoundException e) {

            System.out.println("File not found: " + e.getMessage());

        } catch (IOException e) {

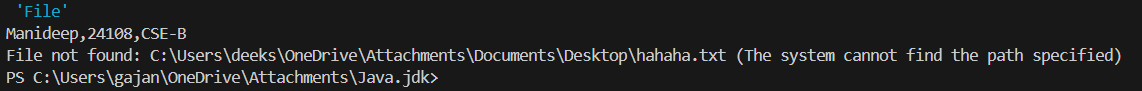
            System.out.println("Error reading file: " + e.getMessage());

        }

    }

}

**Output:**

****

**Errors:**

**CLASS DIAGRAM:**

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

|  |
| --- |
| Uses: |
| - BufferedReader  - FileReader  - FileNotFoundException  - IOException |

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error | Error Rectification |
| 1. FileNotFoundException 2. IOException 3. Syntax Error | 1. Occurs if the specified file path is wrong or file does not exist. 2. Occurs while reading file if an input/output error happens. 3. If missing semicolon ;, wrong try-catch block syntax. |

**IMPORTANT POINTS:**

1) Used BufferedReader and FileReader to read text files.

2) FileNotFoundException occurs if the file is missing.

3) IOException occurs for input/output errors during file reading.

4) try-catch block is used for proper exception handling.

5) Always close the reader after reading the file (reader.close()).

**3)write a java program to handle arthematic expression try catch and finally**

**Program:**

import java.util.Scanner;

class Arthematic {

    public static void main(String[] args) {

        try {

            Scanner sc = new Scanner(System.in);

            System.err.println("enter your first number :");

            int a = sc.nextInt();

            System. out. println("enter your secind number :");

            int b = sc.nextInt();

            int result = a/b;

            System.out.println("Result: " + result);;

            System.out.println("Manideep,24108,CSE-b");

        } catch (Exception e) {

            System.out.println(" Error: Division by zero or invalid input."+e.getMessage());

        }

        finally{

            System.out.println(" program is beaing exicuted");

        }}}

**Output:**

A black screen with white text

AI-generated content may be incorrect.

**Errors:**

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

|  |
| --- |
| Uses: |
| - BufferedReader  - FileReader  - FileNotFoundException  - IOException |

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error | Error Rectification |
| 1)FileNotFoundException  2)IOException  3)Syntax Error | 1)Occurs if the specified file path is wrong or file does not exist.  2)Occurs while reading file if an input/output error happens.  3)If missing semicolon ;, wrong try-catch block syntax. |

**IMPORTANT POINTS:**

1) Used BufferedReader and FileReader to read text files.

2) FileNotFoundException occurs if the file is missing.

3) IOException occurs for input/output errors during file reading.

4) try-catch block is used for proper exception handling.

5) Always close the reader after reading the file (reader.close()).

**4)write a java program to simulate a university system using inner classes**

**create an outer class named "university" with a variable universityName**

**inside it two non static inner classes:**

**1.department with variable like deptName and deptCode and a method to display department details**

**2.student with variables like student name and roll number and a method to display**

**3.create an object for each class and call their methods to display their details along with universityname**

**ex: public class university{**

**string universityname="amrita Vishwa Vidyapeetham"**

**program:**

class University {

    String universityName = "Amrita vishwa vidyapitam";

    class Department{

        String deptName = "CSE";

        int deptcode = 1012;

}

        class Student{

            String studentName = "Mani";

            int rollNum = 1420;

        }

        public static void main(String[] args) {

            University un = new University();

            System.err.println(un.universityName);

            Department dp = un.new Department();

            System.err.println(dp.deptName);

            System.err.println(dp.deptcode);

            Student st = un.new Student();

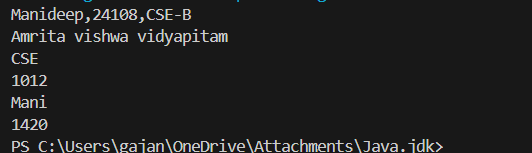
            System.err.println(st.studentName);

            System.err.println(st.rollNum);

        }

}

**Output:**

****

**CLASS DIAGRAM:**

|  |
| --- |
| University |
| -universityName: String  + University(String name) |

**Inner Classes:**

**- Department**

**- deptName: String**

**- deptCode: String**

**+ displayDepartment(): void**

**- Student**

**- studentName: String**

**- rollNumber: int**

**+ displayStudent(): void**

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

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error | Error Rectification |
| 1. Syntax Error 2. Compilation Error 3. Runtime Error | 1. Wrong object creation for inner class 2. Accessing outer class members wrongly 3. NullPointerException if outer object missing |

**IMPORTANT POINTS:**

1. **Demonstrates inner class usage.**
2. **Inner classes access outer class members easily.**
3. **Separate objects for Department and Student.**

**Good example of encapsulation.**