**23CSE111**

**Object-Oriented Programming**

**LAB MANUAL**



**Department of Computer Science**

**Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati**

**Campus**

Verified By: Name: T. Lasya Lakshmi

Roll No: AV.SC.U4CSE24330

**Week – 1**

* **Aim:** To download and Install Java Compiler in our laptops.

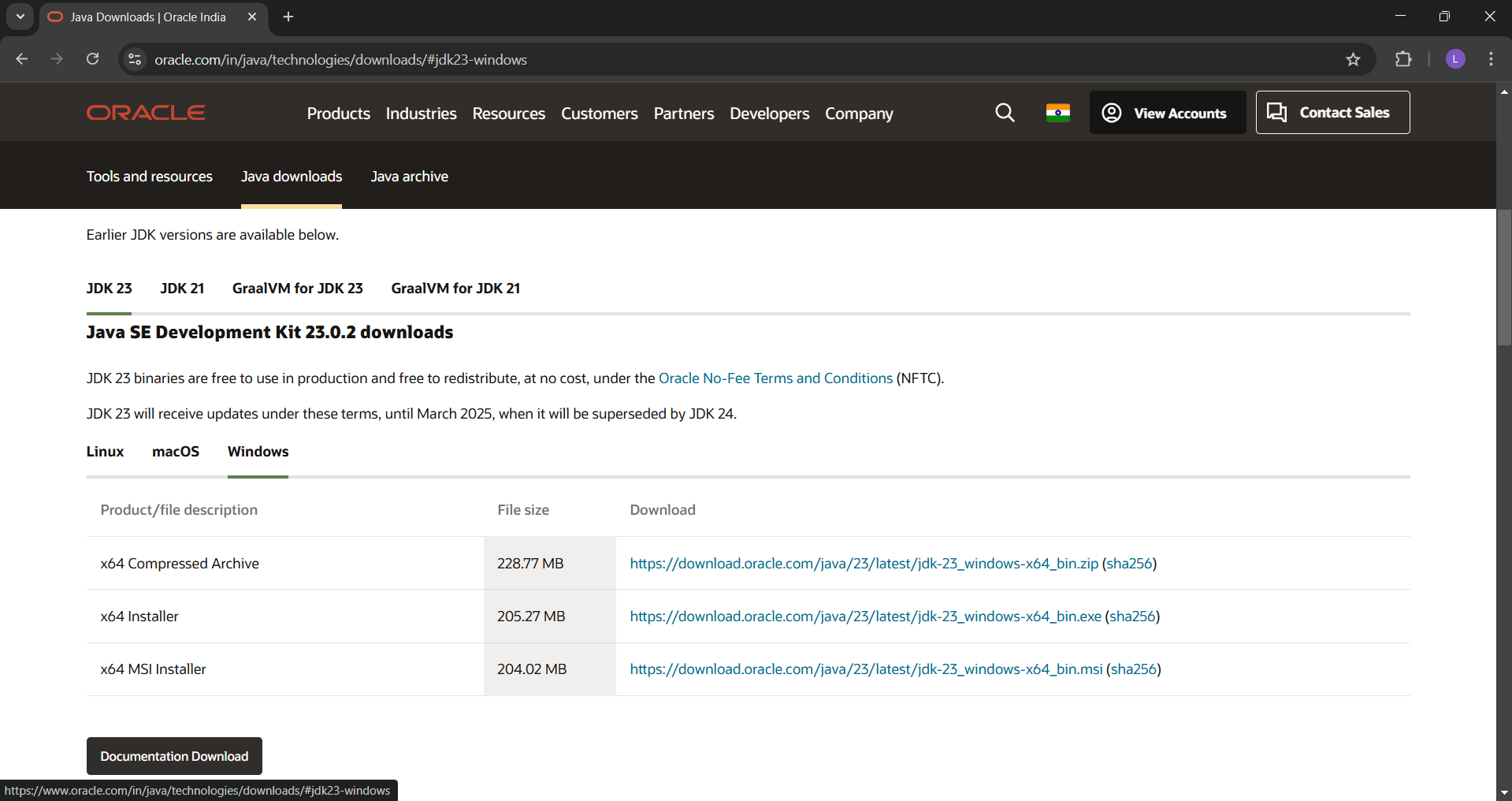
**Procedure:**

**Follow the steps below to install Java on Windows:**

1. **Download JDK(Java Development Kit)**
2. **Run the Installer**
3. **Configure Environment Variables**
4. **Verify Installation**

**1.Download JDK:**

**Go to the official oracle website to download the JDK. Choose x64 MSI Installer on the windows tab and click on download link.**

****

**2. Run the Installer:**

**Now, go to your downloads folder and run the installer you just downloaded.**

**3. Configure Environment Variables:**

**After installation, you will need to tell your system where to find Java. This is done by setting environment variables.**

**Locate JDK Path:**

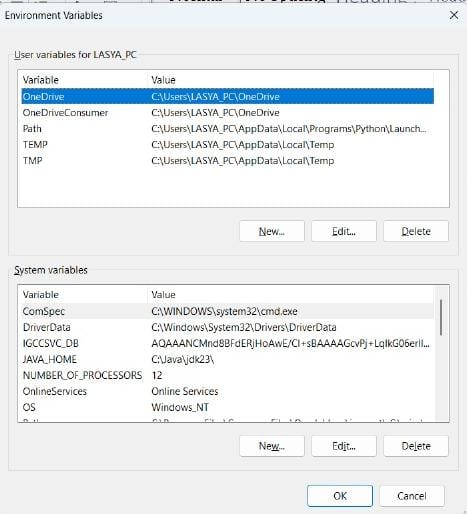
**Navigate through your file explorer to reach the JDK installation directory. Normally, it is located at -**

**C:\Program Files\Java\jdk-23\bin**

**Copy this path.**

**Access Environment Variables:**

**Search environment variable on the terminal. In system properties, click on environment variables. You will be prompted to the screen below.**



**Update the Path Variable:**

**Find the Path variable in the System variables section and click on edit. Then, click New and paste your JDK bin path.**

**Finally, click Ok to close each window.**

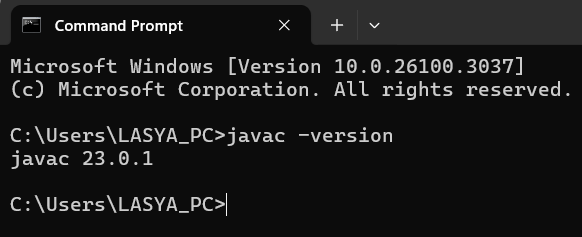
**Set JAVA\_HOME Variable:**

**Back in the environment variables window, under the system variables section, click New to create a new variable.**

**Now, name the variable  and set its value to the path of the JDK folder directory. Close all the dialogues with the Ok button.**

1. **Verify Installation:**

**After the installation, you can verify whether Java is installed by using the following command in the command prompt.**

****

**If Java is installed successfully, it will print the version information; otherwise, it will produce an error message indicating that the command is not recognized.**

* **Aim:** To print basic details of a student.

**CODE:**

import java.io.\*;

public class Lasya{

public static void main(String args[]){

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

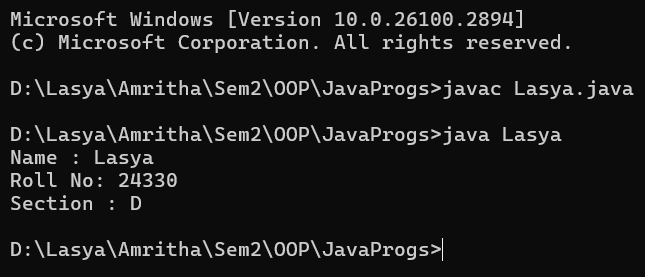
System.out.println("Roll No: 24330");

System.out.println("Section : D");

}

}

**OUTPUT:**



* **Aim:** To understand various Data types

**CODE:**

import java.io.\*;

public class DataTypes{

public static void main(String args[]){

boolean result=true;

System.out.println(result);

double num = 6.5;

System.out.println(num);

long range = 94938516;

System.out.println(range);

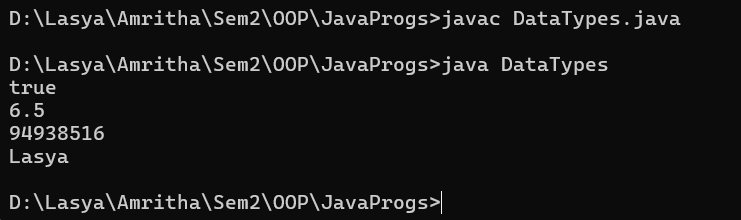
String word = "Lasya";

System.out.println(word);

}

}

**OUTPUT:**



**Week – 2**

* **Aim:** Write a Java program to calculate area of rectangle.

**CODE:**

import java.util.Scanner;

class Area {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter length: ");

float l = input.nextFloat();

System.out.print("Enter width: ");

float b = input.nextFloat();

input.close();

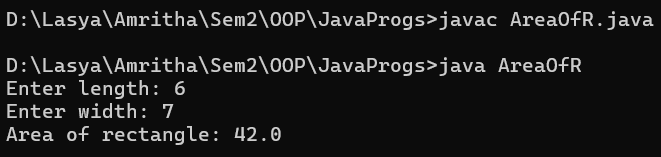
float area = l \* b;

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

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl.No:** | **Errors:** | **Error rectification** |
| 1. | error: incompatible types: Scanner cannot be converted to System  System input= new Scanner(System.in); | Change scanner into Scanner |

**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

2. Scanner input=new Scanner(System.in); - Used to create a Scanner object

3. int ln=input.nextInt(); - Used to read the integer data type stored under the object created

4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

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

**CODE:**

import java.util.Scanner;

class Temperature {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Fahrenheit: ");

float F = input.nextFloat();

input.close();

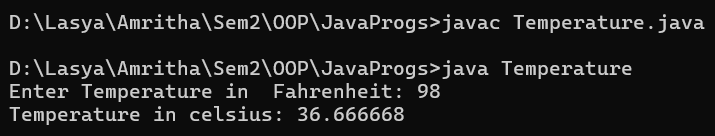
float C = (F - 32)\*5/9;

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

}

}

**OUTPUT:**

****

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

**CODE:**

import java.util.Scanner;

class temperature2 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Celsius: ");

float C = input.nextFloat();

input.close();

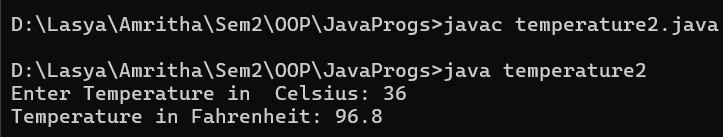
float F = (C \* 9/5) + 32;

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

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl.No:** | **Errors:** | **Error rectification** |
| 1. | error: incompatible types:  F = (C . 9/5) + 32;  not declarable variable. | Change  F = (C . 9/5) + 32; into  F = (C \* 9/5) + 32; |
| 2. | error: ';' expected  System.out.print("Enter the temp in Farenheit:") | Add a semicolon at the end of the statement  System.out.print("Enter the temp in Farenheit:"); |

**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

2. Scanner input=new Scanner(System.in); - Used to create a Scanner object

3. double fh=input.nextDouble(); - Used to read double data type stored under the object created

4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

* **Aim:** Write a Java program to calculate simple interest.

**CODE:**

import java.util.Scanner;

class Si {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter principle: ");

float P = input.nextFloat();

System.out.print("Enter time: ");

float T = input.nextFloat();

System.out.print("Enter rate: ");

float R = input.nextFloat();

input.close();

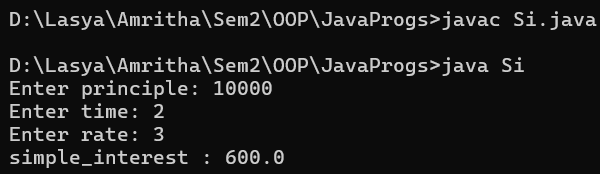
float SI = (P\*T\*R)/100;

System.out.println("simple\_interest : " + SI );

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl.No:** | **Errors:** | **Error rectification** |
| 1. | error: cannot find symbol double intr=(p\*r\*t)/100;  symbol: variable p  location: class interest | Create a reader object  double p=input.nextDouble(); |
| 2. | error: ';' expected  double intr=(p\*r\*t)/100 | Add a semicolon at the end of the statement  double intr=(p\*r\*t)/100; |

**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

2. Scanner input=new Scanner(System.in); - Used to create a Scanner object

3. double p=input.nextDouble(); - Used to read double data type stored under the object created

4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

* **Aim:** Write a Java program to calculate largest of 3 numbers using ternary operators.

**CODE:**

import java.util.Scanner;

class LargestOfThree {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number1: ");

float n1 = input.nextFloat();

System.out.print("Enter number2: ");

float n2 = input.nextFloat();

System.out.print("Enter number3: ");

float n3 = input.nextFloat();

input.close();

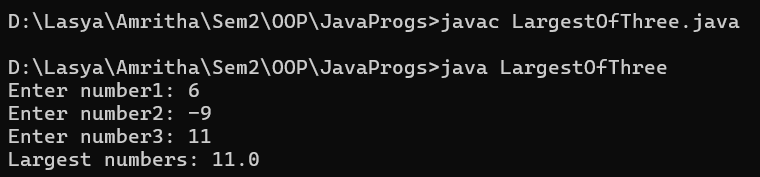
float largest = (n1 >= n2) ? ((n1 >= n3) ? n1 : n3) : ((n2 >= n3) ? n2 : n3);

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

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl.No:** | **Errors:** | **Error rectification** |
| 1. | error: ';' expected  int result=(a>b) ((a>c)? a:c) : ((b>c)? b:c);  error: not a statement  int result=(a>b) ((a>c)? a:c) : ((b>c)? b:c); | Add a ‘?’  int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); |
| 2. | error: ';' expected  int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c) | Add a ‘;’ int result=(a>b)? ((a>c)?  a:c) : ((b>c)? b:c); |

**Concepts to be known:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

2. Scanner input=new Scanner(System.in); - Used to create a Scanner object

3. int a=input.nextInt (); - Used to read integer data type stored under the object created

4. int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); - Nested Ternary operator is used here.

Syntax for ternary operator is- condition? expression 1: expression 2; , whose answer is stored in a variable and then used.

* **Aim:** Write a Java program to calculate factorial of a number.

**CODE:**

import java.util.Scanner;

class Factorial {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

input.close();

long factorial = 1;

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

factorial \*= i;

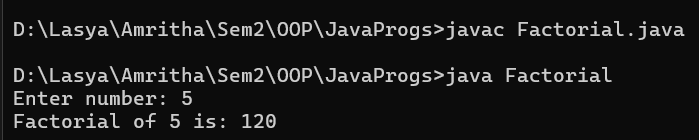
}

System.out.println("Factorial of " + n + " is: " + factorial);

}

}

**OUTPUT:**

****

**ERRORS:**

No errors found

**Concepts to be known:**

1. for (int i=1; n>=i;--n){ } - For loop syntax: for(initial expression; test expression; update expression){} The loop is executed, until the test expression evaluates to be false.

**Week – 3**

* **Aim:** To create a java program with the following instructions:

a) Create a class with name “Car”

b) Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage

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

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

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

**CODE:**

class Car {

public String car\_color;

public String car\_brand;

public String fuel\_type;

public int mileage;

// Constructor

Car(String car\_color, String car\_brand, String fuel\_type, int mileage) {

this.car\_color = car\_color;

this.car\_brand= car\_brand;

this.fuel\_type = fuel\_type;

this.mileage = mileage;

System.out.println("Welcome to car garage");

System.out.println("");

}

// Constructor ends

// Method to start the car

public void start() {

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

}

// Start method ends

// Method to service the car

public void service() {

System.out.println("Your car is at our service.");

System.out.println("car colour : "+ car\_color);

System.out.println("fuel type: "+ fuel\_type);

System.out.println("Mileage: "+ mileage);

}

// Service method ends

// Method to stop the car

public void stop() {

System.out.println("Stop");

System.out.println("");

}

// Stop method ends

public static void main(String[] args) {

// Create car objects

Car car1 = new Car ("Red", "Maruti", "Diesel", 20);

car1.start();

car1.service();

car1.stop();

Car car2 = new Car ("Black", "Mercedes", "Diesel", 40);

car2.start();

car2.service();

car2.stop();

Car car3 = new Car("White", "Mahindra", "Diesel", 30);

car3.start();

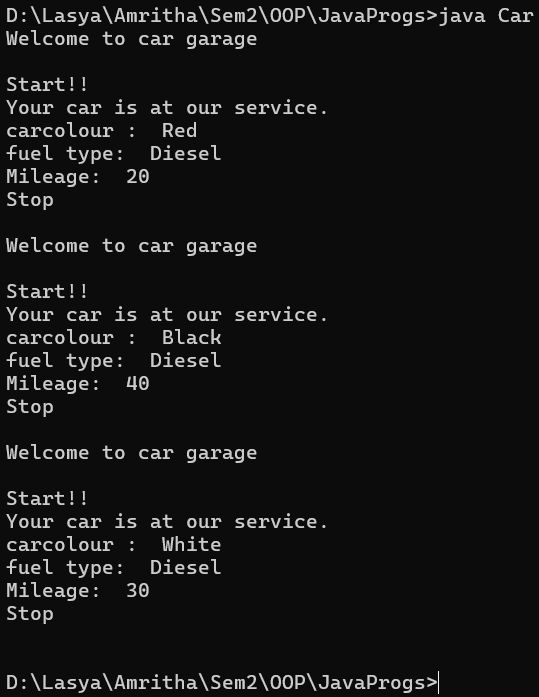
car3.service();

car3.stop();

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: ';' expected  car2.start() | Add a “;”  Car2.start(); |
| 2. | error: illegal start of type  public void stop({ | Add a “)”  public void stop(){ |

**Class Diagram:**

|  |
| --- |
| **Car** |
| + car\_color: String  + car\_brand: String  + fuel\_type: String  + mileage: int |
| + Car(): void  + start(): void  + service(): void  + stop(): void |

**Concepts to be known:**

1. public String car\_color; - Used to declare a variable named car\_color, with data type as String with public accessibility.

2. Car(String car\_color,String car\_brand,String fuel\_type,int mileage){ } – It is a constructor (method with name same as class), which requires

parameters such as car\_color (String data-type) and so on.

3. this.car\_color=car\_color; - “this” is a default method, which is used to point to the instance variables.

* **Aim:** To create a Java Program with class named "my class" with a Static Variable Count int type and initialize to o and A Constant Variable "pi" of type double initialize 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.

**CODE:**

class BankAccount {

private String name;

private int accountNo;

private double balance;

BankAccount(String name, int accountNo, double balance) {

this.name = name;

this.accountNo = accountNo;

this.balance = balance;

}

public void withdrawal(double amount) {

if (amount <= balance) {

balance -= amount;

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

}

else {

System.out.println("Insufficient funds for withdrawal.");

}

}

public void deposit(double amount) {

balance += amount;

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

}

public static void main(String[] args) {

BankAccount A = new BankAccount("Lasya", 170806, 1000);

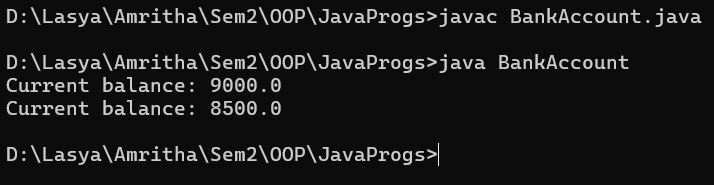
A.deposit(8000);

A.withdrawal(500);

}

}

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: ';' expected  balance -= amount | Add a “;” to  balance -= amount |
| 2. | error: illegal start of type  public void stop({ | Add a “)”  public void stop(){ |

**Class Diagram:**

|  |
| --- |
| **Car** |
| -Balance: double |
| + BankAccount(initialBalance): double  + deposit(amount): void  + withdraw(amount): void |

**Concepts to be known:**

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.

**Week – 4**

* **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 display the detail of the book (Display the details of 2 books i.e create 2 objects and display their details)

**CODE:**

class Book {

//attributes, objects

private static String title;

private static String author;

private static int year;

//initialization

public static void Details(String title, String author, int year){

Book.title = title;

Book.author = author;

Book.year = year;

}

public static void display() {

System.out.println("Title: " + title);

System.out.println("Author: " + author);

System.out.println("Year: "+ year);

System.out.println();

}

public static void main(String[] args) {

Book.Details("Death Note", "Tsugumi Ohba", 2006);

System.out.println("Details of Book 1:");

Book.display();

Book.Details("Jujutsu Kaisen", "Gege Akutami", 2018);

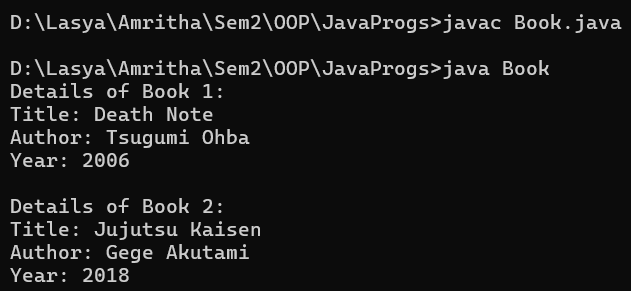
System.out.println("Details of Book 2:");

Book.display();

}

}

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: ';' expected  public static void Details(String title, String author, int year)(. | Add a ‘{‘ in the place of ‘(‘. |
| 2. | error: implicitly declared classes are a preview feature and are disabled by default.  public static void display() { | Remove the space between “Book.” And “Details”. |

**Class Diagram:**

|  |
| --- |
| **Book** |
| - title: String  - author: String  - year: int |
| + Book.display() : void |

**Concepts to be known:**

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.

* **Aim:** 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.

**CODE:**

class MyClass {

static int count = 0;

 static double pi = 3.1415;

MyClass() {

  count++;

}

static void displayCount() {

System.out.println("Total objects created: " + count);

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

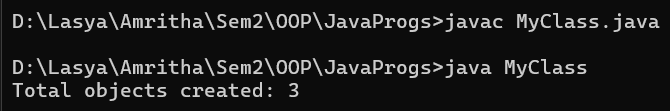
MyClass obj3 = new MyClass();

MyClass.displayCount();

}

}

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: class name not defined. '{' expected. | Add a '{' in the last and remove it after declaring variables. |
| 2. | error: ';' expected  return 3.1415 | Add a ';' after declaring the variable. |

**Class Diagram:**

|  |
| --- |
| **Book** |
| - count: int  - pi: double |
| + displayCount() : void |

**Concepts to be known:**

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.

**Week – 5**

* **Aim:** To create a Java Program of a calculator using the operations including addition Subtraction, multiplication and division using multilevel inheritance & dis Play the desired out Put

**CODE:**

class SimpleCalculator{

//attributes, objects

int a;

int b;

//initialization

public static void add(int a, int b) {

System.out.println(a+ " + " + b + " = " + (a+b));

}

public static void diff(int a, int b) {

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

}

}

class AdvCalculator extends SimpleCalculator {

// Initialization

public static void mul(int a, int b) {

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

}

}

class UltiCalculator extends AdvCalculator{

//initialization

public static void div(int a, int b) {

if(b == 0) {

System.out.println("Denominator should not be a zero ");

}

else{

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

}

}

}

class Calc{

public static void main(String[] args) {

UltiCalculator u = new UltiCalculator();

System.out.println("The calculated values are: ");

u.add(5,5);

u.diff(6,5);

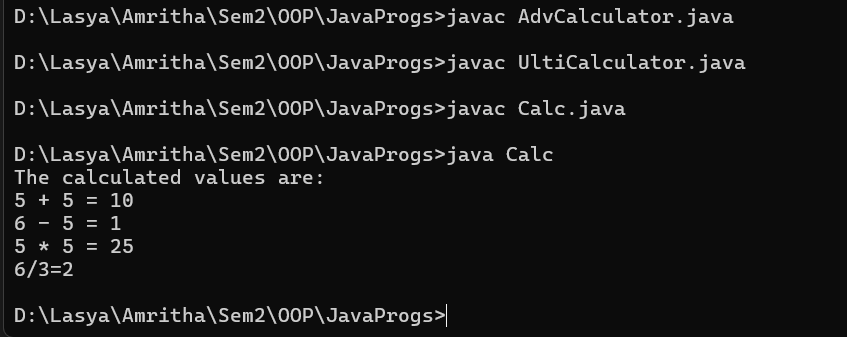
u.mul(5,5);

u.div(6,3);

} // end of the main function

} // end of the class

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: unclosed string literal System.out.println(a+ " - " + b +" = + (a-b)); | Add a '”' after the ‘=’. |
| 2. | error: '(' or '[' expected UltiCalculator u = new UltiCalculator; | Add a '()' after class name. |

**Class Diagram:**

|  |
| --- |
| **SimpleCalculator** |
| a : double  b : double |
| + add (a,b) : void  + diff (a,b) : void |

|  |
| --- |
| **AdvCalculator** |
| + mul (a,b) : void |

|  |
| --- |
| **UltiCalculator** |
| + div (a,b) : void |

|  |
| --- |
| **Calc** |
| + main(String[]) |

**Concepts to be known:**

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.

* **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?
3. Truck should include an additional property capacity (in tons)
4. Create a showTruckdetails() method to display the truck’s capacity.
5. Write a constructor for Truck that initializes all properties
6. 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:**

class Vehicle {

    String brand;

    double speed;

    public Vehicle(String brand, double speed) {

        this.brand = brand;

        this.speed = speed;

    }

    public void displayDetails() {

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

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

    }

    public void start() {

        System.out.println(brand + " vehicle is starting...");

}

}

class Car extends Vehicle {

    int numberOfDoors;

    int seatingCapacity;

    public Car(String brand, double speed, int numberOfDoors, int seatingCapacity) {

        super(brand, speed);

        this.numberOfDoors = numberOfDoors;

        this.seatingCapacity = seatingCapacity;

    }

    public void displayCarDetails() {

        super.displayDetails();

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

        System.out.println("Seating capacity: " + seatingCapacity);

    }

    public void startCar() {

        super.start();

        System.out.println("Car is ready to go!");

    }

}

class Bike extends Vehicle {

    boolean hasGears;

    public Bike(String brand, double speed, boolean hasGears) {

        super(brand, speed);

        this.hasGears = hasGears;

    }

    public void displayBikeDetails() {

        super.displayDetails();

        System.out.println("Has gears: " + (hasGears ? "Yes" : "No"));

    }

    public void startBike() {

        super.start();

        System.out.println("Bike is ready to go!");

    }

}

class Truck extends Vehicle {

    double cargoCapacity;

    public Truck(String brand, double speed, double cargoCapacity) {

        super(brand, speed);

        this.cargoCapacity = cargoCapacity;

    }

    public void displayTruckDetails() {

        super.displayDetails();

        System.out.println("Cargo capacity: " + cargoCapacity + " tons");

    }

    public void startTruck() {

        super.start();

        System.out.println("Truck is ready to go!");

    }

}

public class VehicleRentalSystem {

    public static void main(String[] args) {

        Car car = new Car("Toyota", 150, 4, 5);

        Bike bike = new Bike("Yamaha", 120, true);

        Truck truck = new Truck("Volvo", 90, 10);

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

        car.displayCarDetails();

        car.startCar();

        System.out.println("\nBike Details:");

        bike.displayBikeDetails();

        bike.startBike();

        System.out.println("\nTruck Details:");

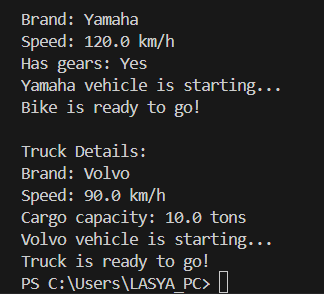
        truck.displayTruckDetails();

        truck.startTruck();

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: Declaring two super classes inside the same file. | Make two separate files to save the two super classes |
| 2. | error: Not declaring the variable using ‘this’ keyword inside the constructor. | Declare the variable using this keyword to run the program. |

**Concepts to be known:**

1. a constructor helps in initializing an object that doesn't exist.

2. a method performs functions on pre-constructed or already developed objects.

3. a double method can represent more decimal point numbers than float method.

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| - brand: string  - speed: double |
| +Vehicle()  + displayDetails() : void  + start() : void |

|  |
| --- |
| **Car** |
| - noOfDoors: int  - seatingCapacity: int |
| +Car()  + displayCarDetails() : void  + startCar() : void |

|  |
| --- |
| Bike |
| - hasGears: Boolean |
| +Bike()  + displayBikeDetails() : void  + startBike() : void |

|  |
| --- |
| **Truck** |
| - cargoCapacity: Boolean |
| + Truck()  +displayTruckDetails() : void  + startTruck() : void |

**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.
* Implement a constructor for the truck class to initialize all its properties.

**Week – 6**

* **Aim:** Write a Java Program to create a Vehicle class with a method display(). Override this method in the Car subclass. Print car model, brand, petrol type, car color and provide the information about the car.

**Code :**

class Vehicle {

String brand;

String petrolType;

String color;

public void display() {System.out.println("This is a vehicle.");

}

}

class Car extends Vehicle {

Car(String brand, String petrolType, String color) {

this.brand = brand;

this.petrolType = petrolType;

this.color = color;

}

public void display() {

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

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

System.out.println("Petrol Type: " + petrolType);

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

}

}

public class Info {

public static void main(String[] args) {

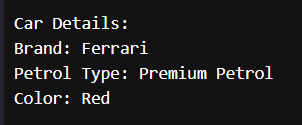
Vehicle myCar = new Car("Ferrari", "Premium Petrol", "Red");

myCar.display();

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: class Main is public, should be declared in a file named Main.java  public class Main { | Save the file name as the name of the main class |

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| + brand : String  + petrolType : String  + color : String |
| + display() : void |

|  |
| --- |
| **Car** |
| Car(String brand, String petrolType, String color) |

**Concepts to be known:**

1. We use the concept of method overriding where the names of the methods in the different classes. The method of the parent class is overridden by the method of the child class

* **Aim:** A college is developing an automated admission system that verifies students’ eligibility for under-graduation and post-graduation. Each program has different eligibility criteria base on the percentage of students in their provided qualifications
* Ug requires 60%
* PG requires 70%

**Code :**

class Student{

String name;

double percentage;

Student(String name, double percentage){

this.name = name;

this.percentage = percentage;

}

public void Eligibility(){

System.out.println(name + " must meet the general admission criteria");

}

}

class UG extends Student{

UG(String name, double percentage){

super(name, percentage);

}

public void Eligibility(){

if (percentage>59){

System.out.println(name + " is elligible for UG admission");

}

else {

System.out.println(name + " is Not elligible for UG admission");

}

}

}

class PG extends Student{

PG(String name, double percentage){

super(name, percentage);

}

public void Eligibility(){

if (percentage>69){

System.out.println(name + " is elligible for PG admission");

}

else {

System.out.println(name + " is Not elligible for PG admission");

}

}

}

public class Admission {

public static void main (String[] args){

UG ug = new UG("Laila", 69);

PG pg = new PG("Majnu", 59);

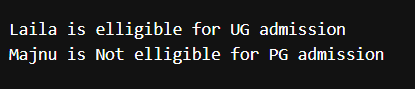
ug.Eligibility();

pg.Eligibility();

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: ';' expected System.out.println(name + " is elligible for UG admission") | Add a ‘;’ after the print statement. |

**Class Diagram:**

|  |
| --- |
| **UG** |
| UG(String name, double percentage) |

|  |
| --- |
| **Student** |
| + name : String  + percentage : double |
| + Student(String name, double percentage): void  + Eligibility(): void |

|  |
| --- |
| **PG** |
| PG(String name, double percentage) |

**Concepts to be known:**

1. The variables once declared in the super class need not be declared twice in any of the sub classes.

2. super keyword is used in sub classes to access the methods of super classes, they are basically the reverse of overriding.

* **Aim:** 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.

**CODE:**

class AddCalculator{

    AddCalculator(){

        System.out.println("This is a 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 class AddCalc{

    public static void main (String [] args){

    AddCalculator calc = new AddCalculator();

    System.out.println("The sum of the 2 numbers: 7 and 18 is "+ calc.add(7, 18));

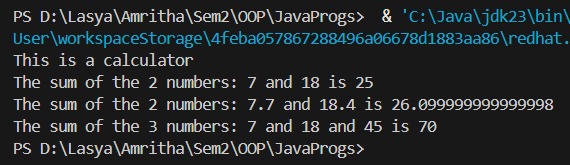
    System.out.println("The sum of the 2 numbers: 7.7 and 18.4 is "+ calc.add(7.7, 18.4));

    System.out.println("The sum of the 3 numbers: 7 and 18 and 45 is "+ calc.add(7, 18, 45));

    }

}

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: Main method not defined | Add public static void main (String [] args){ |
| 2. | error: ';' return type is not expected | Change return type from int to double in its case. |

**Class Diagram:**

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

**Concepts to be known:**

1. We use the concept of method overloading where the names of the methods in the same class are same but the parameters are given different.

* **Aim:** 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.

**Code :**

class Shape{

int calcArea(int a){

return a\*a;

}

int calcArea(int b, int h){

return b\*h;

}

}

class Circle extends Shape{

double r;

double pi = 3.141592653589793;

Circle(double r){

this.r = r;

}

double calcArea(double r){

return pi\*r\*r;

}

}

public class AreaCalc {

public static void main(String[] args) {

Circle c = new Circle(7);

System.out.println("The area of circle is " + c.calcArea(7.7));

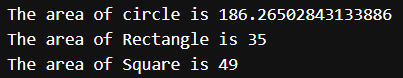
System.out.println("The area of Rectangle is " + c.calcArea(5, 7));

System.out.println("The area of Square is " + c.calcArea(7));

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: invalid method declaration; return type required  calcArea(int a){ | Enter the return type as per required. Here it is int |
| 2. | error: incompatible types: possible lossy conversion from double to int  return pi\*r\*r; | For calculating area of circle, we need to give return type double. |

**Class Diagram:**

|  |
| --- |
| **Circle** |
| + r : double  + pi : double |
| + calcArea(int r) : double |

|  |
| --- |
| **Shape** |
| + calcArea(int a) : int  + calcArea(int b, int h) : int |

**Concepts to be known:**

1. We use the concept of method overloading to calculate the area of square and rectangle in the parent class Shape.

2 we use method overriding in the child class Circle to calculate it’s area.