**LAB MANUAL**



**ROLLNO:AV.SC.U4CSE24121**

**NAME: RITHWIK G**

**SECTION: CSE-B**

**WEEK-1:**

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

printing student details*.*

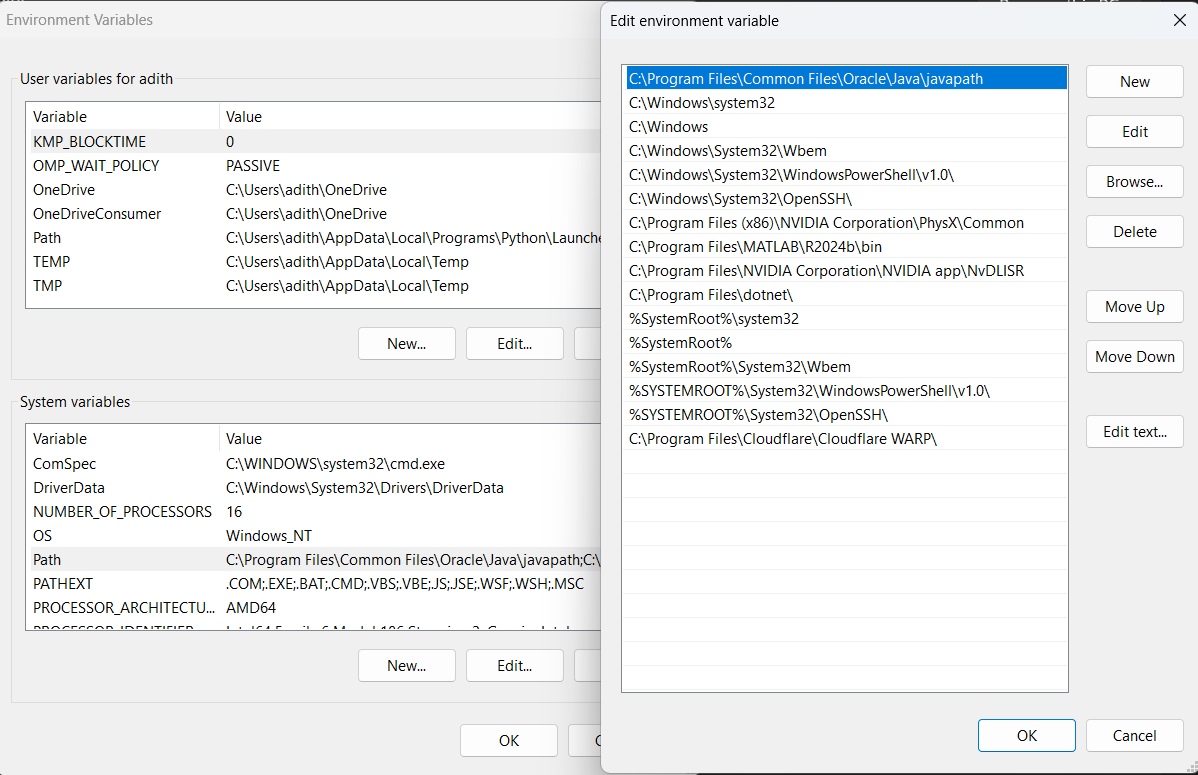
**Step-1:** Download JDK-21 from oracle website

**

**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

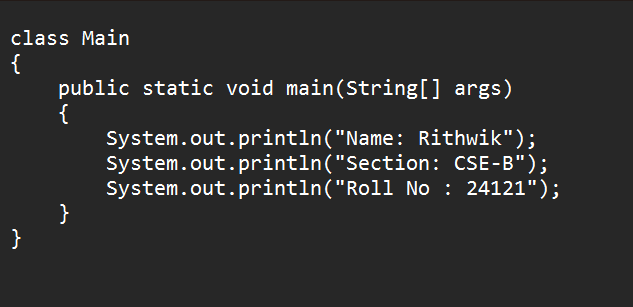
Open cmd-> type java --version

~command propt

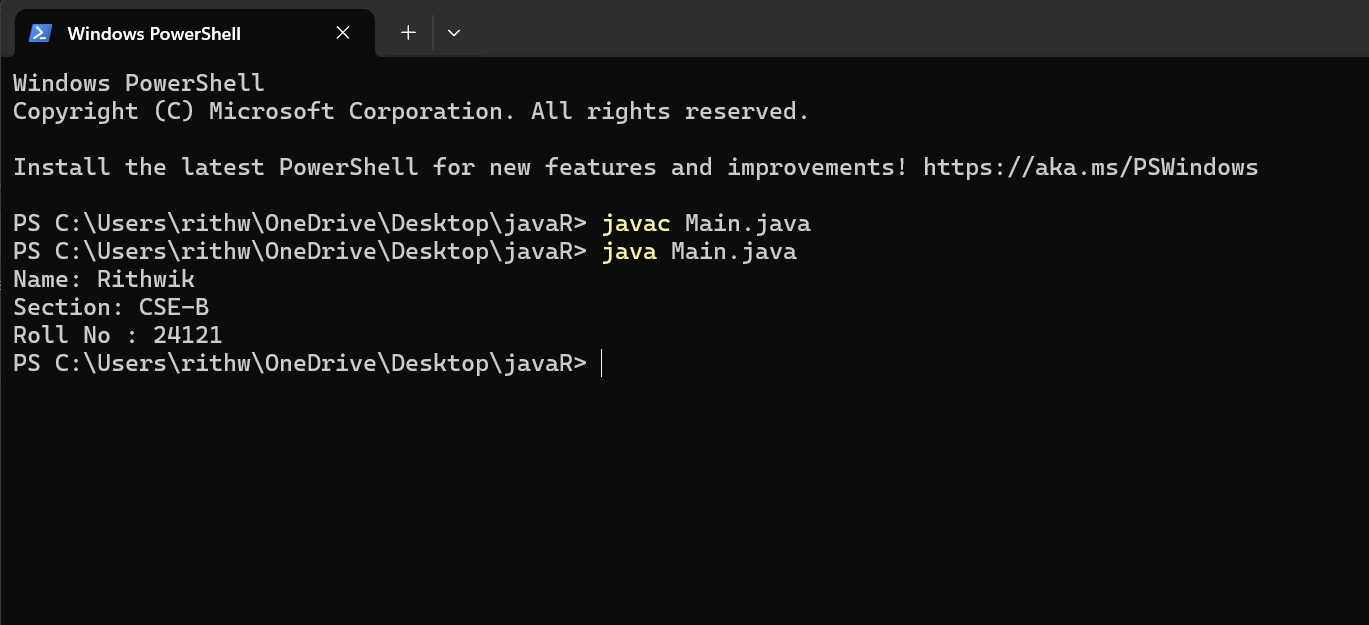
Javac filename.java ->compiling.

Java filename.java ->displaying

**PROGRAM-1(Rectified):**

******

**Output:**



***IMPORTANT POINTS:***

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

**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. |

***IMPORTANT POINTS:***

1. ***Simple interest formula is: (p\*t\*r)/100, where:***

***P: Principal amount***

***R: Rate of interest***

***T: Time period***

1. ***The data type double indicates the floating points in the integers.***
2. ***The line “import java.util.Scanner” indicates:***

***Import: tells the java compiler that we want to use a specific class or package in your code.***

***Java.util : This is the package that contains utility classes for Java programming, including the “Scanner” class.***

***Scanner: this is the class that allows you to read input from the keyboard.***

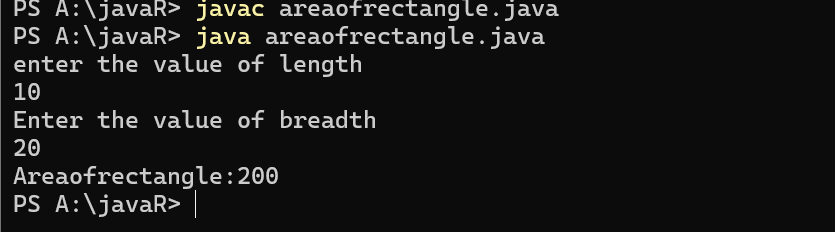
**PROGRAM-2:**

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

rectangle.

**

**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. |

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.

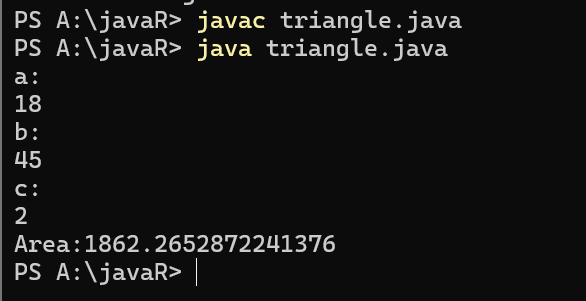
1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**PROGRAM-3:**

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



**Output:**

******

ERROR TABLE:

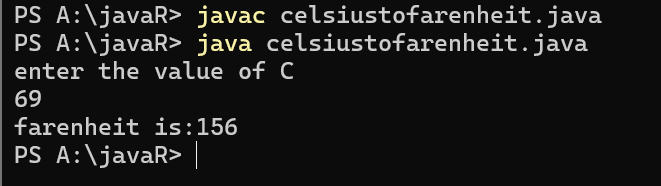
|  |  |
| --- | --- |
| **Code Error** | Code rectification |
| 1.While printing the variable not giving + sign.  2.Not closing the scanner. | 1.We should give correct indentation.  2.Closing the scanner is must. |

**PROGRAM-4(a):**

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



OUTPUT:

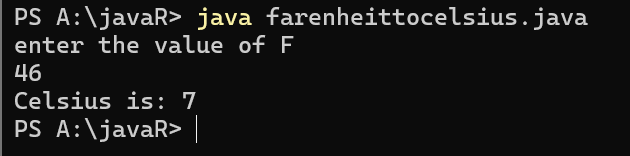


**PROGRAM-4(b):**

**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. |

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.

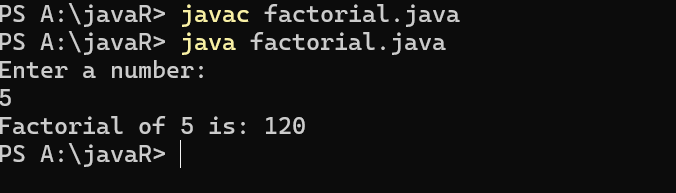
1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**PROGRAM-5:**

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

******

OUTPUT:



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.

ERROR TABLE:

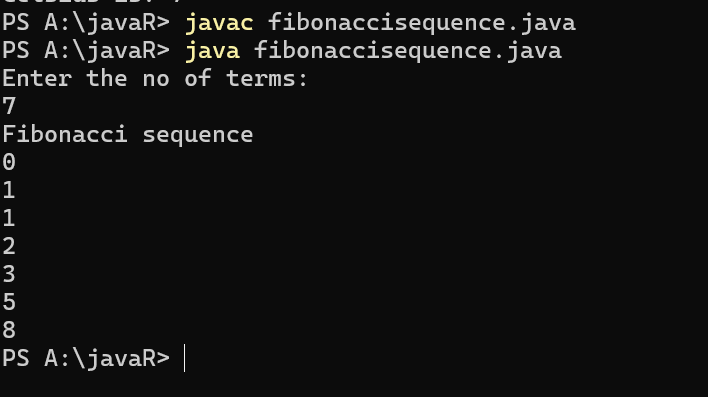
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type as double instead of int. | 1.We should give iterative statements correctly.  2.We should give the data type as int for integers. |

**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:** To create java program with following instructions :

1.Create a class with name Car

2.Create four attributes named car\_color,car\_brand, fuel\_type, mileage

3.Create these methods named start(),stop(),service()

4.Create the objects named car, car1,car2

**CODE:**

public class Car {

private String car\_color;

private String car\_brand;

private String fuel\_type;

private String mileage;

public void start() {

System.out.println("car is started");

}

public void stop() {

System.out.println("car is stopped");

}

public void service() {

System.out.println("car is for service");

}

public static void main(String args[]) {

Car car = new Car();

car.car\_color = "white";

car.car\_brand = "audi";

car.fuel\_type = "petrol";

car.mileage = "20";

car.start();

System.out.println("car\_color: " + car.car\_color + " car\_brand: " + car.car\_brand + " fuel\_type: " + car.fuel\_type + " mileage: " + car.mileage);

Car car1 = new Car();

car1.car\_color = "white";

car1.car\_brand = "audi";

car1.fuel\_type = "petrol";

car1.mileage = "20";

car1.stop();

System.out.println("car\_color: " + car1.car\_color + " car\_brand: " + car1.car\_brand + " fuel\_type: " + car1.fuel\_type + " mileage: " + car1.mileage);

Car car2 = new Car();

car2.car\_color = "white";

car2.car\_brand = "audi";

car2.fuel\_type = "petrol";

car2.mileage = "20";

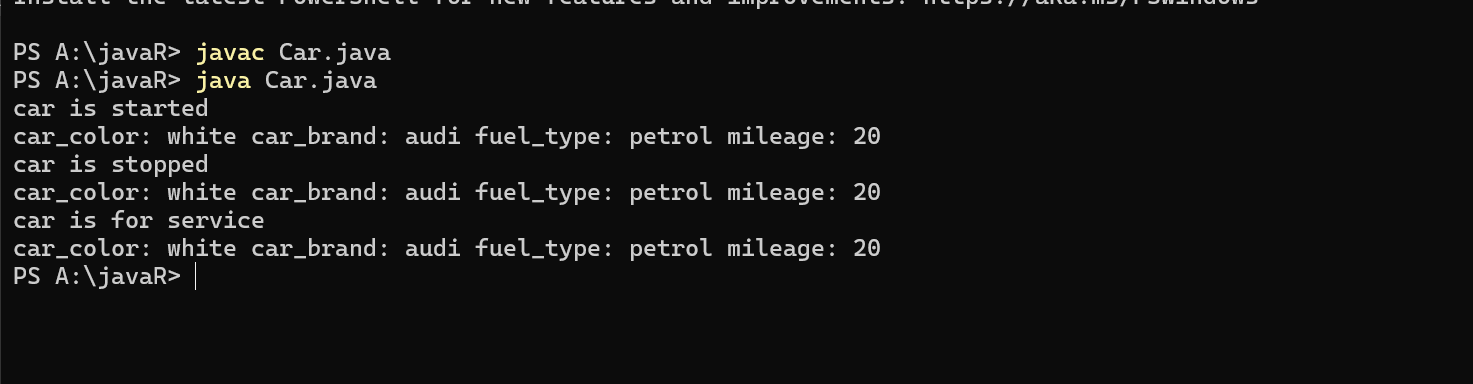
car2.service();

System.out.println("car\_color: " + car2.car\_color + " car\_brand: " + car2.car\_brand + " fuel\_type: " + car2.fuel\_type + " mileage: " + car2.mileage);

}

}

**OUTPUT:**

****

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Cause of error | Rectification |
| 1 | Syntax Error | Missing ‘{‘ | ‘{‘ added |
| 2 | Compile time Error | Mispelled Variable call | Rectified with  Correct variable name |
| 3 | Case sensitive error | Uppercase and lowercase | rectified |

**Class diagram:**

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

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

**PROGRAM-2:**

**Aim:** To create a class BankAccount with methods deposit() and withdraw() . create two subclasses savingsaccount and checkingaccount override the withdraw () method in each subclass to impose different withdrawal limits and fees

public class BankAccount {

protected String accountHolder;

protected double balance;

protected int accountNumber;

public BankAccount(String accountHolder, int accountNumber, double balance) {

this.accountHolder = accountHolder;

this.accountNumber = accountNumber;

this.balance = balance;

}

public void withdrawal(double amount) {

if (amount <= balance) {

balance = balance - amount;

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

} else {

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

}

}

public void deposit(double amount) {

balance = balance + amount;

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

}

public static void main(String[] args) {

BankAccount BA = new BankAccount("Abdul", 24248, 1000);

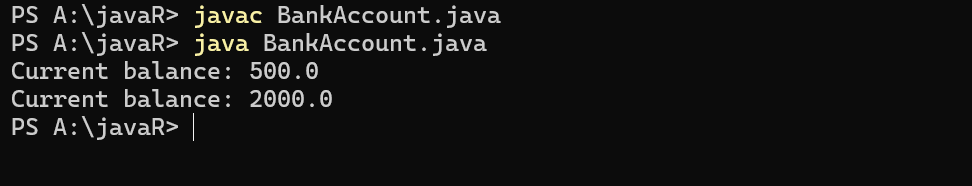
BA.withdrawal(500);

BA.deposit(1500);

}

}

**OUTPUT:**

****

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Error name | Rectification |
| 1 | Name Error | Undefined name | Correct variable  Name replaced |
| 2 | Syntax Error | Missing Parenthesis | Parenthesis Added |
| 3 | Logical Error | Incorrect Condition | Condition Rectified |

**Class diagram:**

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

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.

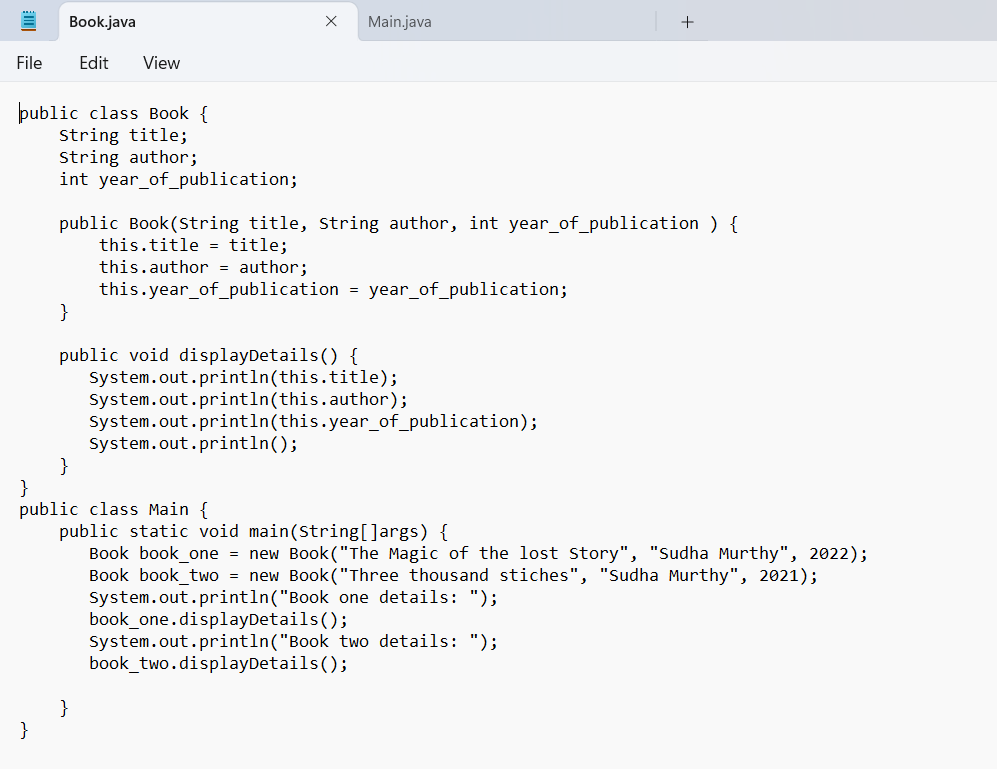
WEEK -4

PROGRAM – 1:

AIM: Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

Create a method which displays the details of the book and display the details of two books.

CODE:



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.

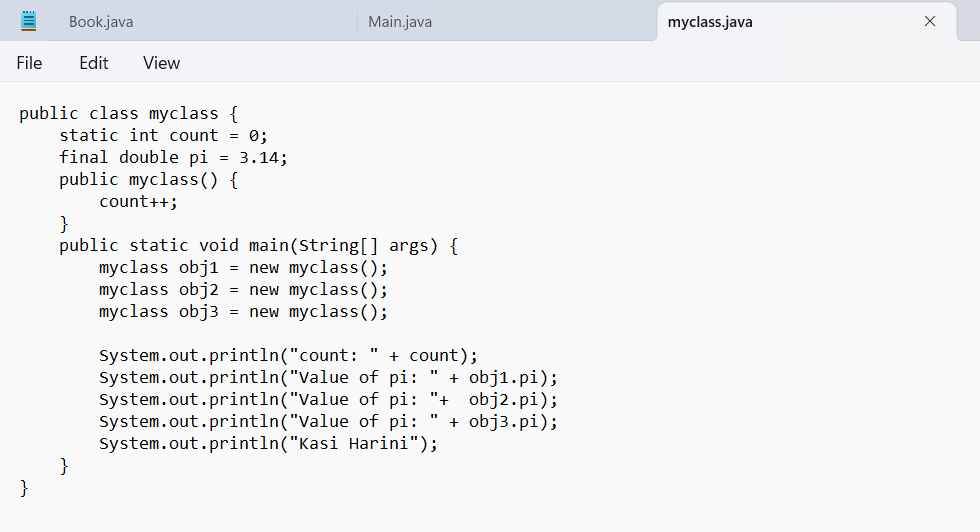
CLASS DIAGRAM:

|  |
| --- |
| 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 static variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of the class, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

CODE:



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:

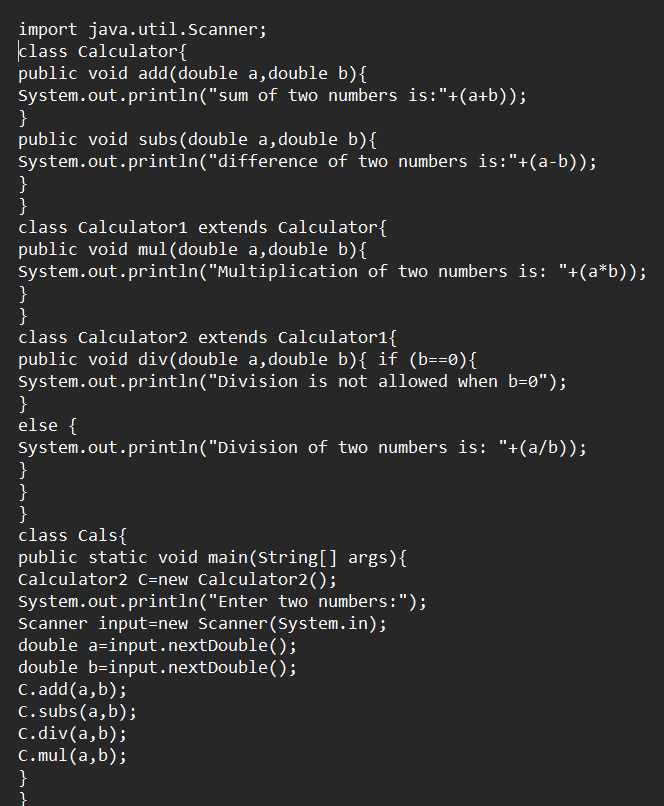
|  |
| --- |
| Myclass   * Count: int * Pi: double   + myclass( )  + main(args: String[]): void |

**Week-5**

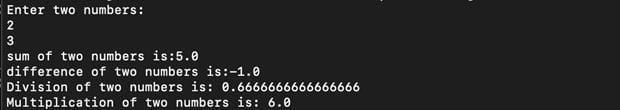
**Program 1:**

**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**.**

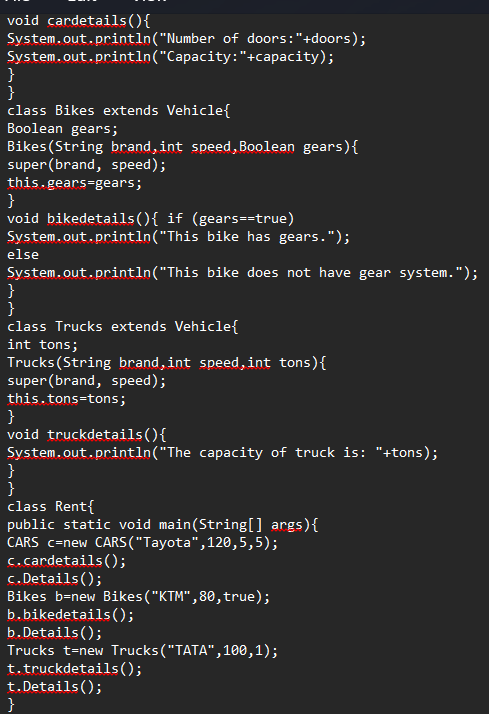
**Questions:**

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?

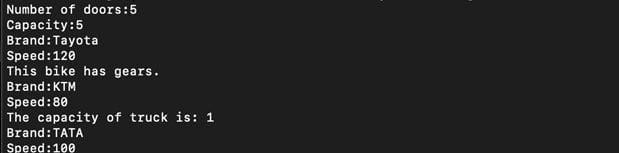
Truck should include and additionalpropertycapacity (in tons).

* + - 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:**

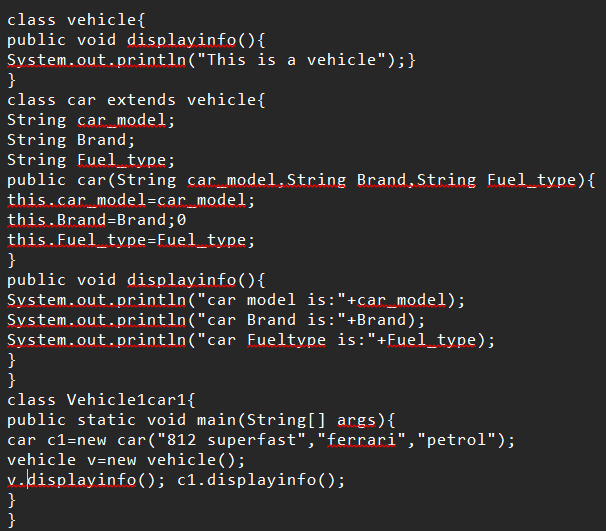
**Program 1:**

**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**.**

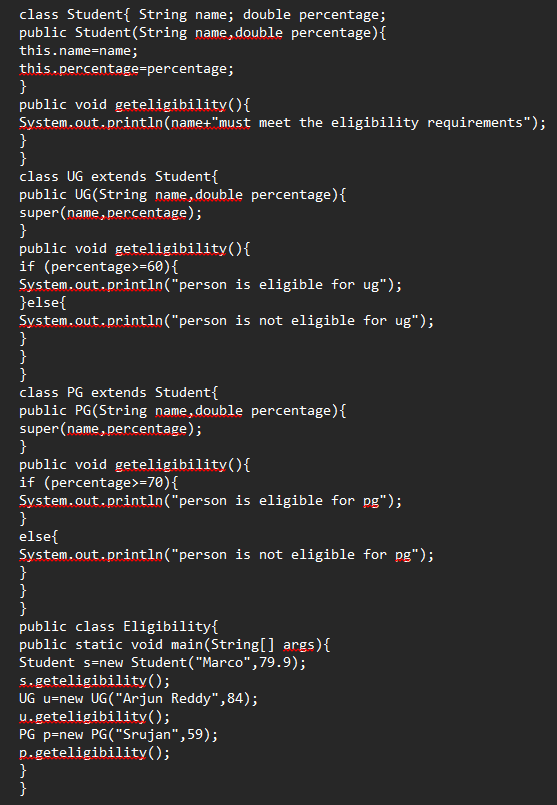
**Program 2:**

**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:**

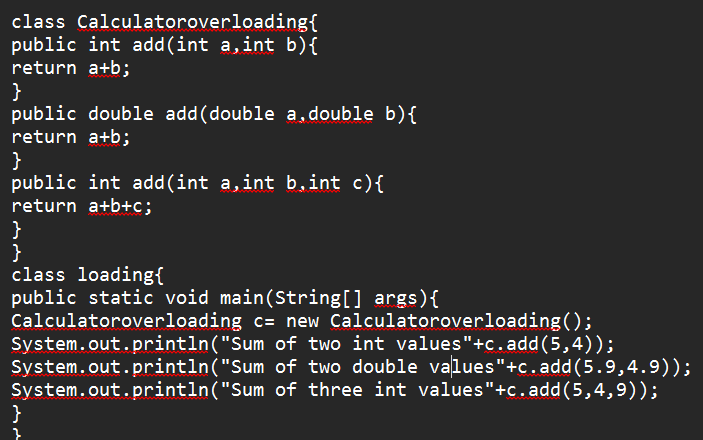
**Aim:** Write a Java Program to create a Calculator class with overloaded methods to perform addition: Take the integer values a and b from the user.

* 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.

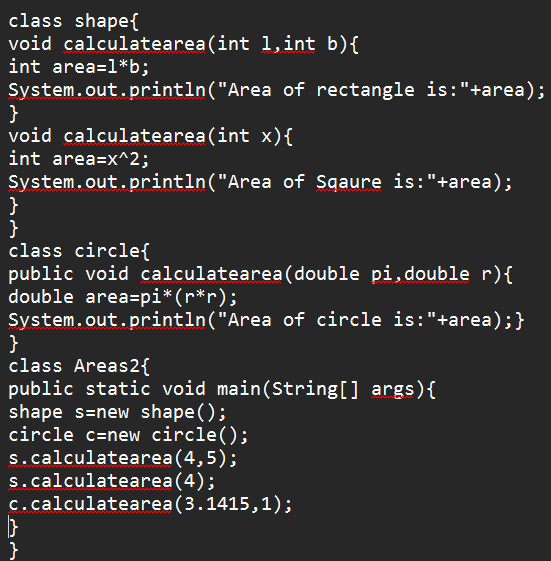
**Important points:**

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-9

PROGRAM-1:

AIM: write a java program to create a method that takes an integer as parameter and throws an exception of the number is even

CODE:

public class week9q1 {

static class EvenNumberException extends Exception {

public EvenNumberException(String message) {

super(message);

}

}

public static void checkOddNumber(int number) throws EvenNumberException {

if (number % 2 == 0) {

throw new EvenNumberException("The number " + number + " is even.");

} else {

System.out.println("The number " + number + " is odd.");

}

}

public static void main(String[] args) {

int testNumber = 20;

try {

checkOddNumber(testNumber);

} catch (EvenNumberException e) {

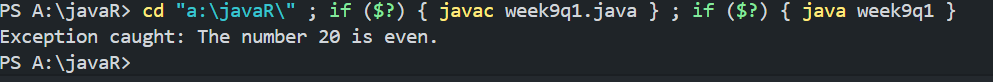
System.out.println("Exception caught: " + e.getMessage());

}

}

}

Output:



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.

PROGRAM-2:

AIM: write a java program to create a method that takes an integer as parameter and throws an exception of the number is even

CODE:

import java.io.\*;

public class week9q2 {

public static void main(String[] args) {

try {

BufferedReader reader = new BufferedReader(

new FileReader("invaild")

);

String line;

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

System.out.println(line);

}

reader.close();

} catch (FileNotFoundException e) {

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

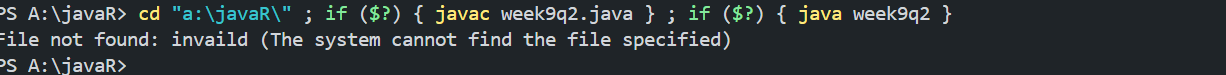
} catch (IOException e) {

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

}

}

}Output:



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()).

PROGRAM-3:

AIM: write a java program to handle an arthematic exception using try catch finally

CODE:

public class week9q3 {

public static void main(String[] args) {

int numerator = 10;

int denominator = 0;

try {

int result = numerator / denominator;

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

} catch (ArithmeticException e) {

System.out.println("ArithmeticException caught: " + e.getMessage());

} finally {

System.out.println("Finally block executed. Cleaning up resources if any.");

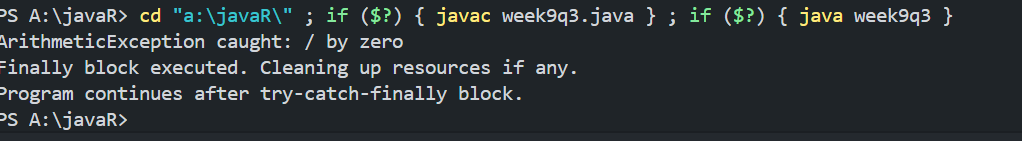
}

System.out.println("Program continues after try-catch-finally block.");

}

}

Output:



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()).

PROGRAM-4:

AIM: Write a Java program to simulate a University system using inner classes.

1) Create an outer class named University with a variable universityName. Inside it, define two non-static inner classes:

2) Department with variables like deptName and deptCode, and a method to display department details.

3) Student with variables like studentName and rollNumber, and a method to display student details.

4) Create an object for each class, their methods to display their details along with the university name.

CODE:

class University{

String University\_Name;

class Department{

String DeptName;

String Deptcode;

public Department(String DeptName, String Deptcode) {

this.DeptName = DeptName;

this.Deptcode = Deptcode;

}

void display(){

System.out.println("Department Name: " + DeptName + ", Department Code: " + Deptcode);

}

}

class Student{

String name;

String studentId;

public Student(String name, String studentId) {

this.name = name;

this.studentId = studentId;

}

void display() {

System.out.println("Student Name: " + name + ", Student ID: " + studentId);

}

}

}

class week9q4{

public static void main(String[] args) {

University university = new University();

University.Department csDept = university.new Department("Computer Science", "CSE24121");

University.Student student1 = university.new Student("RITHWIK G", "24121");

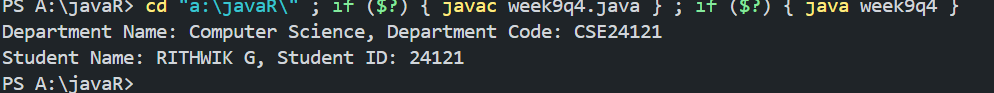
csDept.display();

student1.display();

}

}

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.