23CSE111

OOPS

(Object Oriented Programming System)

LAB MANUAL

A logo with pink letters

Description automatically generated

Department of CSE

Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amaravati Campus

**NAME: S.SAI VENKAT**

**ROLLNO: AV.SC.U4CSE24322**

**SECTION: CSE-B**

|  |  |  |  |
| --- | --- | --- | --- |
| **WEEK** | **QUESTIONS** | **PAGE**  **NO** | **Date** |
| WEEK 1 | 1.To Download and install Java.  2.Execute first java program | 5 | 29/01/2025 |
| WEEK 2 | 1) Write a java program to calculate the area of the rectangle  2)Write a java program to convert the temperature from Celsius to Fahrenheit.   1. Write a java program to calculate the Simple Interest. 2. Write a java program to find the greatest of 3 numbers using terenery operators. 3. Write the java program for the factorial of a number. | 8 | 12/02/2025 |
| WEEK 3 | Q.) Create The java program.   1. Car 2. Bankaccount | 15 | 19/02/2025 |
| WEEK 4 | 1.To create a java program named book.  2.To create a java program with class named "Myclass" with a ststic variable "count" of int type initilized to zero and a constant variable "pi" of type double initilized to 3.1415 as attributes of that class now define a constructor for "Myclass" that increments the count each time the object of "Myclass" is created finally print the final values of "count" and "pi" variable. | 19 | 5/03/2025 |
| WEEK 5 | 1.•Java code for building a calculator using multilevel inheritance  •Having subclasses addition, subtraction, multiplication and division.  2.Java code for vehicle rental system having subclasses car, bike and truck with the help hierarchical inheritance. | 23 | 12/03/2025 |
| 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.  2.A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.  -UG admission require min of 60%  -PG admission require min of 70%  3.Create a calculator class with overloaded methods to perform addition.  A.Add two integers  B.Add two double  C.Add three integer.  4.Create a shape class with a method CalculateArea() that is overloaded for different shpaes (e.g square, rectangle) then, create a subclass circle that overrides the calculatearea() method for a circle. | 28 | 26/03/2025 |
| 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 calculate Volume () 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.To write a java program using an abstract class to define a method for the pattern printing. | 35 | 2/04/2025 |
| WEEK 8 | 1.Write a Java program to create an interface Shape with the getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.  2.Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.  3.Write a Java program to implement a login system using interfaces. | 41 | 9/04/2025 |
| WEEK 9 | 1.Write a java program to create a method that takes integer as parameter and throws an exception if the number is even.  2.Write a java program to create a method that reads a file and throws an exception if the file is not found.   1. Write a java program to handle arthematic exception using try, catch and finally. 2. Write a java program to simulate a University system using inner classes.  * create a outer class named university with a variable universityName. * inside this define two non-static inner classes.   1.Department - with variables like deptName and DeptCode and a method to display department details.  2.Student - with Variables like StudentNameand rollNumber and a method to display student details,  3.Create a object for each calss and call their methods to display thier methods along with their university name. | 47 | 16/04/2025 |
| WEEK 10 | 1.Write a java program to generate a password for a student using his/her initials and age. the password displayed should the string consists of first chacter of first name, middle name, last name with age.  2.design and implement a java program that will do the following operations to this string  "Welcome! you are practicing strings concept"  i)convert all alphabets to capital letters and print out the result. ii)convert all alphabets to lower-case letters and print put the result iii)print out the length of the string iv)print out the index of concept  3.Implement a java program using the below array methods  -Sorting the elements (numbers and Strings)of an array  -Convert the array elements into string  -Fill the part of an array.  -Copy the elements of one array into another.  4.Implement a java program using the below array list methods  -Insert an element at particular index in the array list  -Modify an element in the array list.  -Access an element from the array list.  -Remove an element from the array list.  -Clear the elements from the array list. | 54 | 30/04/2025 |

**WEEK-1:**

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

printing student details*.*

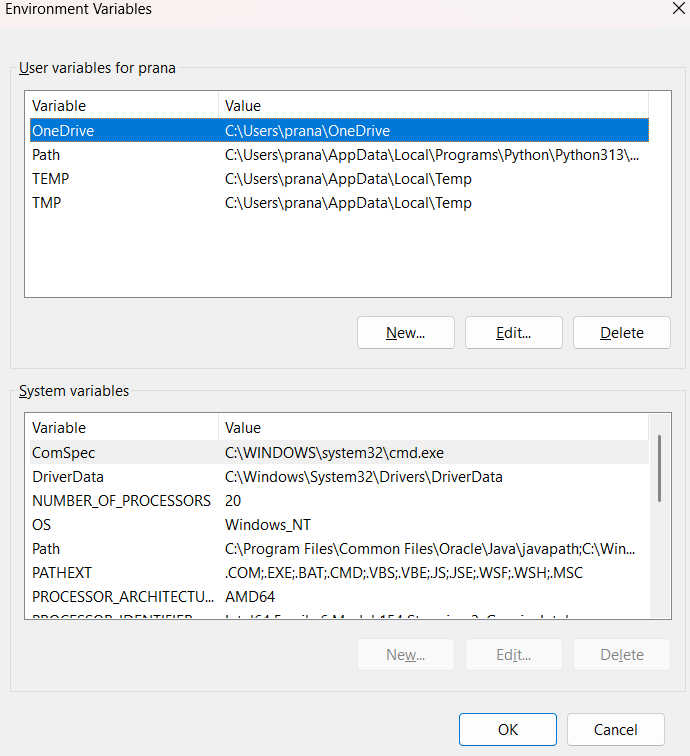
**Step-1:** Download JDK-21 from oracle website

**

**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 ->display

**PROGRAM-1:**

**Aim:** Write a program in java for displaying

student details.

**Code:**

class intro {

public static void main(String[] args) {

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

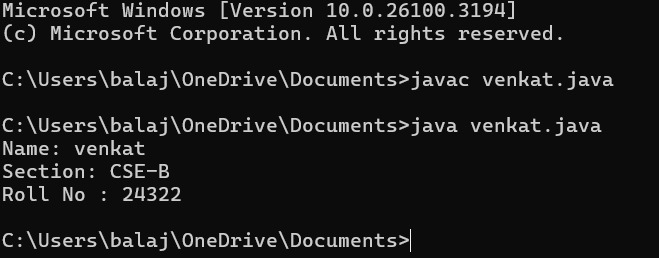
System.out.println("CSE-B ");

System.out.println("24322");

}

}

**Output:**

******

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **Code \** | **Error** | **Rectification** |
| System.out.println  ("Class: CSE-C") | Semi colon(;) is  missing at the  end. | Add a semi colon(;) at the  end.  System.out.println("Class:  CSE-C"); |

**WEEK-2:**

**PROGRAM-1:**

**Aim:** Write a java program for Simple Interest.

**Code:** import java.util.Scanner;

public class SimpleInterest {

public static void main(String[] args) {

System.out.println("S.Saivenkat, 24322, CSE-B");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the principal amount: ");

double principal = scanner.nextDouble();

System.out.print("Enter the rate of interest : ");

double rate = scanner.nextDouble();

System.out.print("Enter the time period (in years): ");

double time = scanner.nextDouble();

double simpleInterest = (principal \* rate \* time) / 100;

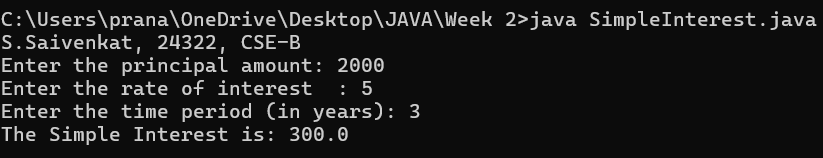
System.out.println("The Simple Interest is: " + simpleInterest);

scanner.close();

}

}

**Output:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not using the closing semi colon. | 1.Should not give space between next and Double.  2.We must put semi colon after each line when required. |

**PROGRAM-2:**

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

rectangle.

**Code:**

import java.util.Scanner;

class areaOfRectangle{

public static void main(String[] args){

System.out.println("S.Saivenkat, 24322, CSE-B"");

Scanner scanner=new Scanner(System.in);

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

double length= scanner.nextDouble();

System.out.print("Enter the breadth: ");

double breadth= scanner.nextDouble();

double areaOfRectangle=length\*breadth;

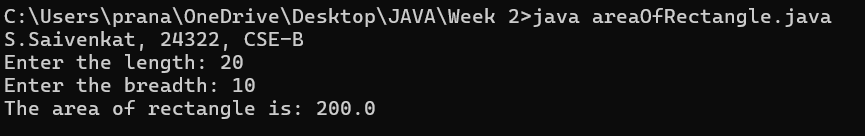
System.out.println("The area of rectangle is: " + areaOfRectangle);

scanner.close();

}

}

**Output:**

****

**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.

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

**ERROR TABLE:**

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

**PROGRAM-3:**

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

**Code:**

import java.util.Scanner;

class areaOfTriangle{

    public static void main(String[] args){

    System.out.println("S.Saivenkat, 24322, CSE-B");

    Scanner scanner=new Scanner(System.in);

    System.out.print("a: ");

    double a= scanner.nextDouble();

    System.out.print("b: ");

    double b = scanner.nextDouble();

    System.out.print("c: ");

    double c= scanner.nextDouble();

    double s=(a+b+c)/2;

    double areaOfTriangle=Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

    System.out.println("The area of triangle is: " + areaOfTriangle);

        scanner.close();

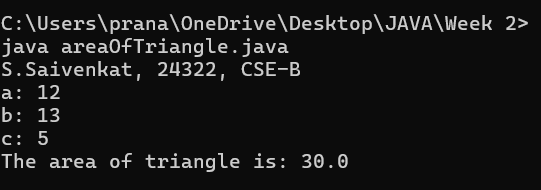
    }

}

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | Code rectification |
| 1.While printing the variable not giving + sign.  2.Declaring the data type. | 1.We should give correct indentation.  2.We should give the data type. |

**Output:**

****

**IMPORTANT POINTS:**

1. Here, we’re finding the area of a triangle using heron’s formula.
2. Heron’s formula for finding a triangle is:

S = (a +b +c)/2

Where S is the semi-perimeter of the triangle.

Now the area formula is:

Area = sqrt(s\*(s-a)\*(s-b)\*(s-c)).

**PROGRAM-4(a):**

**Aim:** Write a program in java for converting temperature from Celsius to Fahrenheit.

**Code:**

import java.util.Scanner;

public class tempConverter {

    public static void main(String[] args) {

        System.out.println(("S.Saivenkat, 24322, CSE-B");

        Scanner scanner = new Scanner(System.in);

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

        double celsius = scanner.nextDouble();

        double fahrenheit = (celsius \* 9/5) + 32;

        System.out.println(celsius + "°C is equal to " + fahrenheit + "°F");

        scanner.close();

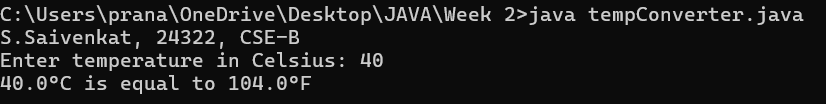
    }

}

**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. |

**OUTPUT:**



**PROGRAM-4(b):**

**Aim:** Write a program in java for converting temperature from Fahrenheit to Celsius.

**Code:**

import java.util.Scanner;

public class tempConverter2 {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B");

        Scanner scanner = new Scanner(System.in);

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

        double Fahrenheit = scanner.nextDouble();

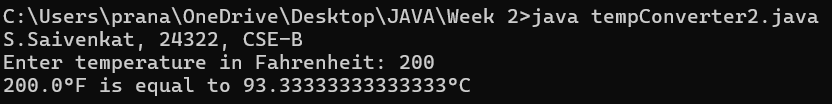
        double Celsius = (Fahrenheit -32)\*5/9;

        System.out.println(Fahrenheit + "°F is equal to " + Celsius + "°C");

        scanner.close();

    }

}

**Output:** ******

**ERROR TABLE:**

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

**PROGRAM-5:**

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

**Code:** import java.util.Scanner;

public class Factorial {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number to calculate its factorial: ");

        int number = scanner.nextInt();

        if (number < 0) {

            System.out.println("Factorial is not defined for negative numbers.");

        } else {

            long factorial = 1;

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

                factorial =factorial\*i;

            }

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

        }

        scanner.close();

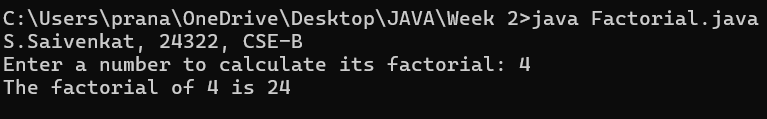
    }

}

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly. | 1.We should give iterative statements correctly. |

OUTPUT:



**PROGRAM-6:**

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

**Code:**

import java.util.Scanner;

public class Fibonacci {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of terms for the Fibonacci series: ");

        int n = scanner.nextInt();

        int first = 0, second = 1;

        System.out.print("The first " + n + " terms of Fibonacci series: ");

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

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

            int next = first + second;

            first = second;

            second = next;

        }

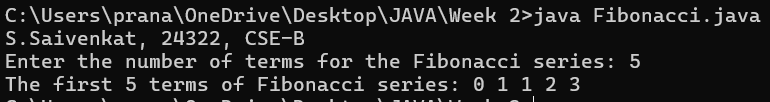
        scanner.close();

    }

}

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not putting flower brackets in the end. | 1.Should not give space between next and Double.  2.We must put giving flower brackets in the end. |

OUTPUT: 

IMPORTANT POINTS:

1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

**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

**Class Diagram:**

****

**CODE:**

public class Car {

    private String color;

    private String brand;

    private String fueltype;

    private String mileage;

    public void start() {

        System.out.println("The car has started");

    }

    public void stop() {

        System.out.println("The car has stopped");

    }

    public void service() {

        System.out.println("The car has for service");

    }

    public static void main(String args[]) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        Car car = new Car();

        car.color = "white";

        car.brand = "audi";

        car.fueltype = "petrol";

        car.mileage = "20";

        car.start();

        System.out.println("car color:" + car.color + ", car brand: " + car.brand + ", fuel type: " + car.fueltype + ", mileage: " + car.mileage);

        Car car1 = new Car();

        car1.color = "black";

        car1.brand = "BMW";

        car1.fueltype = "petrol";

        car1.mileage = "25";

        car1.stop();

        System.out.println("car color: " + car1.color + ", car brand: " + car1.brand + ", fuel type: " + car1.fueltype + ", mileage: " + car1.mileage);

        Car car2 = new Car();

        car2.color = "white";

        car2.brand = "Toyota";

        car2.fueltype = "petrol";

        car2.mileage = "30";

        car2.service();

        System.out.println("car color: " + car2.color + ", car brand: " + car2.brand + ", fuel type: " + car2.fueltype + ", mileage: " + car2.mileage);

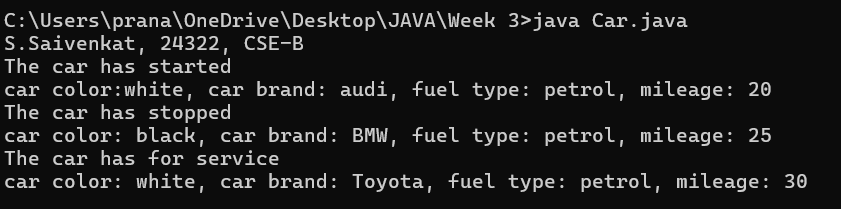
    }

}

**Error table:**

|  |  |
| --- | --- |
| Error | Rectification |
| Missing ‘;‘ | ‘;‘ added |
| Mispelled Variable call | Rectified with  Correct variable name |
| Uppercase and lowercase | rectified |

**OUTPUT:**

****

**IMPORTANT POINTS:**

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

**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

**Class diagram:**

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

**Code:**

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) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        BankAccount BA = new BankAccount("Hardik", 24244, 1000000);

        BA.withdrawal(50000);

        BA.deposit(150000);

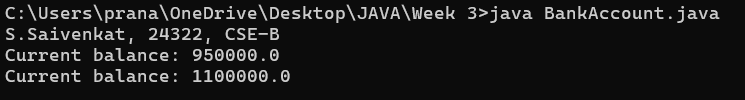
    }

}

**Error table:**

|  |  |  |
| --- | --- | --- |
| Error name | Error name | Rectification |
| Name Error | Undefined name | Correct variable  Name replaced |
| Syntax Error | Missing Parenthesis | Parenthesis Added |
| Logical Error | Incorrect Condition | Condition Rectified |

**OUTPUT:**

****

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 and Year of Publication. It should also contain a constructor with parameter which initializes Title, Author and Year of publication. Create a method which displays the details of the book. Display the details of two books.

**Class Diagram:**

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

**CODE:**

class book {

public String title;

public String author;

public String year\_of\_publication;

public void book(){

this.title=title;

this.author=author;

this.year\_of\_publication=year\_of\_publication;

}

public static void main(String[] args){

book book1=new book();

book book2=new book();

book1.book();

book1.title="Sherlock Holmes ";

book1.author="Arthur Conan Doyle";

book1.year\_of\_publication="1887";

book2.book();

book2.title="Harry Potter";

book2 .author="J.K. Rowling";

book2.year\_of\_publication="1997";

System.out.println("Book-1");

System.out.println("Title :" +book1.title);

System.out.println("Author :" +book1.author);

System.out.println("Year of publication :" +book1.year\_of\_publication);

System.out.println("Book-2");

System.out.println("Title :" +book2.title);

System.out.println("Author :" +book2.author);

System.out.println("Year of publication :" +book2.year\_of\_publication);

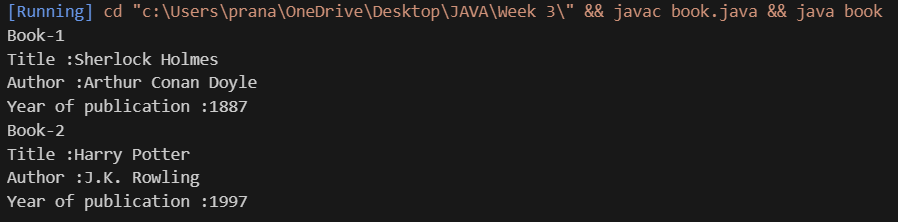
}

}

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Rectification** |
| **1.** | **Missing “;” after calling method.** | **Added “;”** |

**OUTPUT:**

****

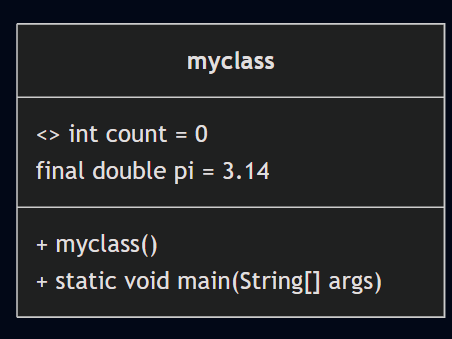
IMPORTANT POINTS:

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.

**PROGRAM-2:**

**Aim:** Create a java program with class named myclass with a static variable count of int type initialized to 0 and a constant variable “Pi” of type double initialized to 3.14 has attributes of that class. Now define a constructor for “myclass” that increments the count variable each time an object of myclass is created finally print final values count and Pi variables. Create three objects

**Class Diagram:**

****

**Code:**

class myclass {

    static int count = 0;

    final double pi = 3.14;

    myclass() {

        count++;

    }

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        myclass object1 = new myclass();

        myclass object2 = new myclass();

        myclass object3 = new myclass();

        System.out.println("The value of Count variable : " + count);

        System.out.println("The value of Pi variable :" + object1.pi);

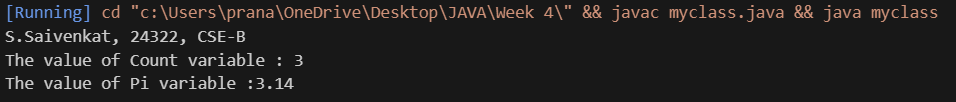
    }

}

Error Table:

|  |  |  |
| --- | --- | --- |
| S.No | Error | Rectification |
| 1. | Not typing “}”at the end of the code. | Added “}”. |

**OUTPUT:**

****

IMPORTANT POINTS:

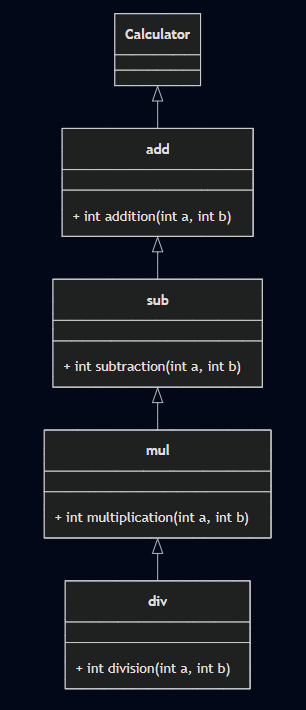
1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

**WEEK-5:**

**PROGRAM-1**

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

**Class Diagram:**

****

**Code:** class Calculator {

}

class add extends Calculator {

    public int addition(int a, int b) {

        return a + b;

    }

}

class sub extends add {

    public int subtraction(int a, int b) {

        return a - b;

    }

}

class mul extends sub {

    public int multiplication(int a, int b) {

        return a \* b;

    }

}

class div extends mul {

    public int division(int a, int b) {

        return a / b;

    }

}

public class Multilevel {

    public static void main(String[] args) {

System.out.println("S.Saivenkat, 24322, CSE-B ");

        div obj = new div();

        System.out.println("Addition: " + obj.addition(10, 2));

        System.out.println("Subtraction: " + obj.subtraction(10, 2));

        System.out.println("Multiplication: " + obj.multiplication(10, 2));

        System.out.println("Division: " + obj.division(10, 2));

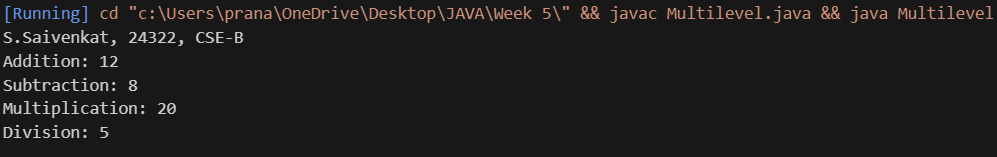
    }

}

**ERROR TABLE :**

|  |  |  |
| --- | --- | --- |
| s.no | Error | Error rectification |
| 1 | Syntax error on token ")” | ‘{’ should be mentioned |
| 2 | calculator cannot be resolved to a type | ‘c’ should be capital. |

**Output:**

****

**IMPORTANT POINTS:**

We use multilevel inheritance in this

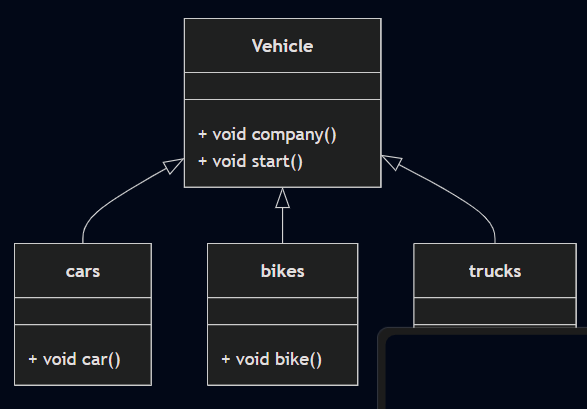
Each subclass inherits methods from its parent class, gaining access to addition, subtraction,

Multiplication and division.

**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 bikes and they need a program to store details about each vehicle such as brand & speed.

**Class Diagram**:****

**Code:**

class Vehicle{

    void company(){

        System.err.println("Vehicle Company");

    }

    void start(){

    System.out.println("Vehicle is starting");

    }

}

class cars extends Vehicle{

    void car(){

        System.out.println("It has 4 doors");

    }

}

class bikes extends Vehicle{

    void bike(){

        System.out.println("It has gears");

    }

}

class trucks extends Vehicle{

    void truck(){

        System.out.println("It has gears");

    }

}

public class Hierarchial{

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        cars c = new cars();

        c.car();

        c.start();

        c.company();

        bikes b = new bikes();

        b.bike();

        b.start();

        b.company();

        trucks t = new trucks();

        t.truck();

        t.start();

        t.company();

    }

}

ERROR TABLE :

|  |  |  |
| --- | --- | --- |
| s.no | Error | Error rectification |
| 1 | Syntax error on token ")” | ‘{’ should be mentioned |
| 2 | Heirarchial cannot be resolved to a type | ‘h’ should be capital |

**Output:** 

**IMPORTANT POINTS:**

The code demonstrates inheritance using a parent class Vehicle and three child classes: Car, Bike, and

Truck.

Each subclass has its own constructor that extends the parent class constructor

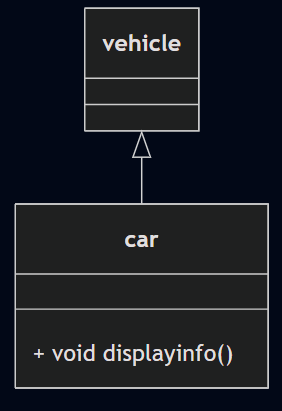
**WEEK-6**

**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.

**Class Diagram:**

****

**Code**:

class vehicle{

}

class car extends vehicle{

    void displayinfo(){

    System.out.println("This is a Ferrari 488 GTB.");

    System.out.println("Engine: 3.9-liter twin-turbocharged V8");

    System.out.println("Horsepower: 661 hp (at 8,000 rpm)");

    System.out.println("Starting price: Around $250,000 USD");

    }

}

public class ovveride{

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        car c1=new car();

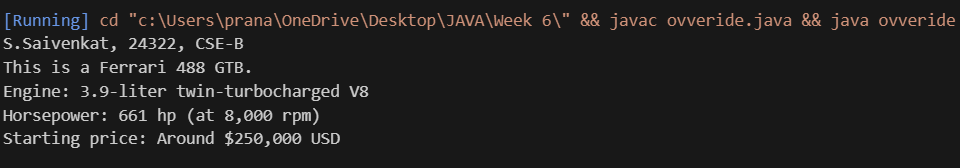
        c1.displayinfo();

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **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) |

**Output:** 

**IMPORTANT POINTS:**

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

**Program 2**

**Aim:** A college is developing an automated admission system that verifies students eligibility for undergraduate(UG) and post graduate(PG) programs Each program has different eligibility criteria based on theh student’s percentage in the previous qualification.

UG admission require a minimum of 60%.

PG admission require a minimum of 70%.

**Class Diagram:**

|  |
| --- |
| **adm** |
| elg():void |

|  |  |
| --- | --- |
| ug | pg |
| +elg():void | +elg():void |

**Code:**

class student{

    String name;

    double percentage;

public student(String name,double percentage){

    this.name=name;

    this.percentage=percentage;

    }

}

class UG extends student{

    public UG(String name,double percentage){

        super(name,percentage);

    }

    void eligibility(){

        if(percentage>=60){

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

        }

        else{

            System.err.println(name+" is not eligible for UG addmission.");

        }

    }

}

class PG extends student{

    public PG(String name,double percentage){

        super(name,percentage);

    }

    void eligibility(){

        if(percentage>=70){

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

        }

        else{

            System.err.println(name+" is not eligible for PG addmission.");

        }

    }

}

public class Admission{

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        UG s1=new UG("Dhanush",90);

        PG s2=new PG("Karthik",90);

        s1.eligibility();

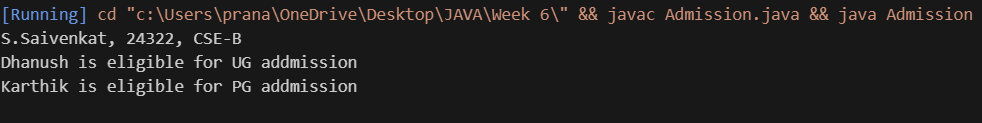
        s2.eligibility();

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **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)** |

**Output:** 

**IMPORTANT POINTS:**

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

**Program 3**

**Aim:** Create a calculator class with overloaded methods to perform addition.

1. Add 2 integers.
2. Add 2 doubles.
3. Add 3 integers.

Class Diagram:

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

**Code:**

class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

    public double add(double a, double b) {

        return a + b;

    }

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

        return a + b + c;

    }

}

public class Overload {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        Calculator a = new Calculator();

        System.out.println("Sum is " + a.add(5, 6));

        System.out.println("Sum is " + a.add(5.6, 6));

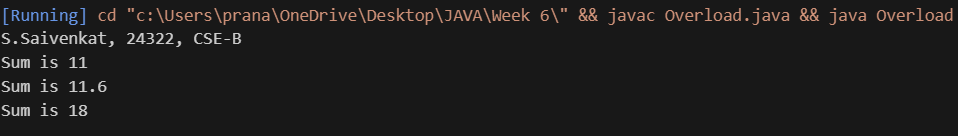
        System.out.println("Sum is " + a.add(5,6,7));

    }

}

**Error table:**

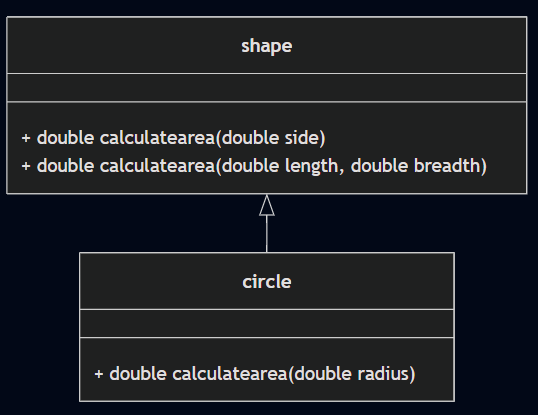
|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| 2 | } | Ending the class and main method is required |

Output: 

**Program 4**

**AIM**: Create a shape class with a method calculateArea() that is overloaded for different shapes. Then create a subclass circle that overrides the calculator method for a circle.

**CLASS DIAGRAM:**

****

Code:

class shape{

    public double calculatearea(double side){

        return side\*side;

    }

    public double calculatearea(double length,double breadth){

        return length\*breadth;

    }

}

class circle extends shape{

    public double calculatearea(double radius){

        return 3.14\*radius\*radius;

    }

}

public class overrides {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        circle c=new circle();

        shape s=new shape();

        shape r=new shape();

        System.out.println("Area of circle = "+c.calculatearea(5));

        System.out.println("Area of square = "+s.calculatearea(5));

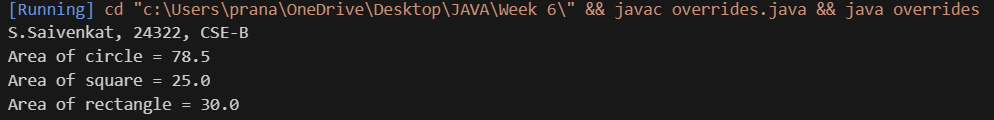
        System.out.println("Area of rectangle = "+s.calculatearea(5,6));

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| 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) |

Output: 

**IMPORTANT POINTS:**

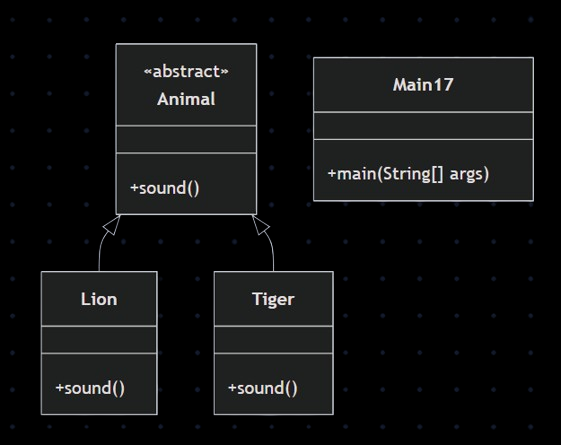
In this program we use both method overloading and overriding to calculate area of different shapes.

**WEEK-7**

**Program 1**

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

Class Diagram:



Code:

abstract class Animal {

    abstract void sound();

}

class Lion extends Animal {

    void sound() {

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

    }

}

class Tiger extends Animal {

    void sound() {

        System.err.println("Tiger Roars");

    }

}

public class Main17 {

    public static void main(String[] args) {

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        Lion l = new Lion();

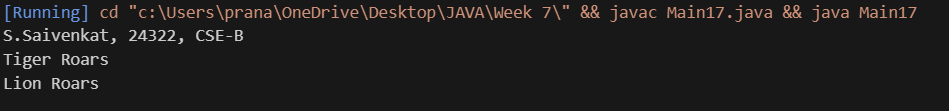
        Tiger t = new Tiger();

        t.sound();

    l.sound();

    }

}

Output:  **IMPORTANT POINTS:**

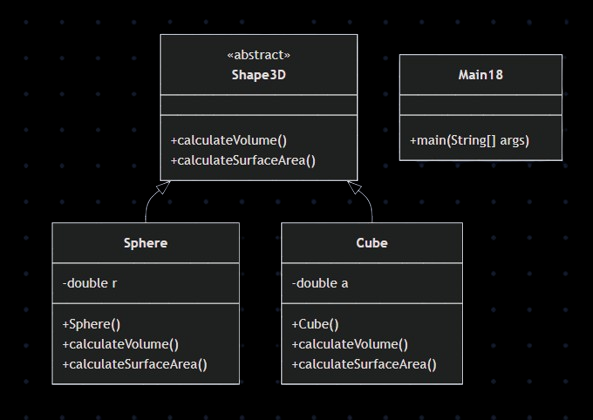
Both Lion and Tiger override the sound () method from the Animal abstract class to provide their own specific implementation.

The class Animal is declared as abstract meaning it cannot be instantiated directly. It serves as a blueprint for other classes.

**Program 2**

**AIM**: Write a java program to create an abstract class Shape3D with abstract methods calcluateVolume() 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.

Class Diagram:



Code:

abstract class Shape3D{

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D{

    private double radius;

    public Sphere(double radius){

        this.radius=radius;

    }

    public double calculateVolume(){

        return (4.0/3.0)\*Math.PI\*Math.pow(radius,3);

    }

    public double calculateSurfaceArea(){

        return 4.0\*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 6.0\*Math.pow(side,2);

    }

}

public class Main18{

    public static void main(String[] args){

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        Sphere s=new Sphere(5);

        System.err.println("Surface area of Sphere= "+s.calculateSurfaceArea());

        System.err.println("Volume of Sphere= "+s.calculateVolume());

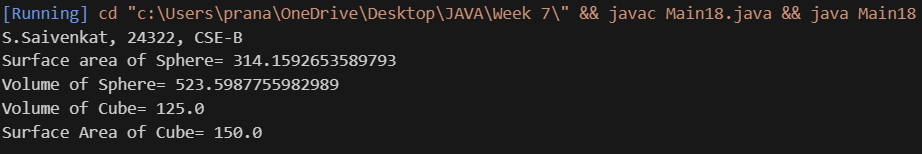
        Cube c=new Cube(5);

        System.err.println("Volume of Cube= "+c.calculateVolume());

        System.err.println("Surface Area of Cube= "+c.calculateSurfaceArea());

    }

}

Output: 

IMPORTANT POINTS:

Each shape class:

Calculates and prints volume and surface area using its respective formulas.

Demonstrates method overriding with specific implementations in each subclass.

**Program 3**

AIM: write a java program using an abstract class to define a method for pattern printing

create an abstract class named PatternPrinter with an abstract method printPattern(int n) and a concrete method to display the

pattern title.

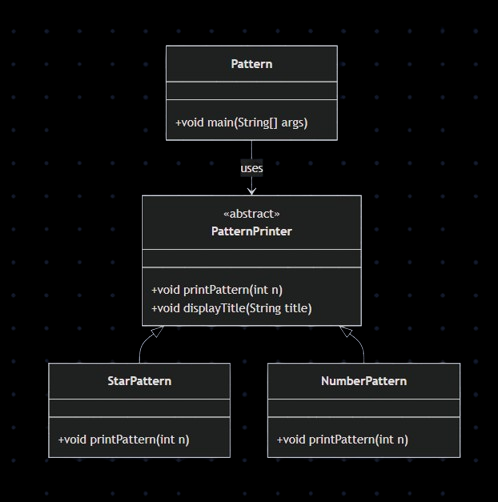
implement the two subclasses:

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

2. NumberPattern-prints a right angled triangle of increasing numbers

in the main() method create objects of both subclasses and print the patterns for a given number of rows.

**Class Diagram:**



**Code:**

import java.util.Scanner;

abstract class PatternPrinter {

abstract void printPattern(int n);

void displayTitle(String title) {

System.out.println("\n" + title);

}

}

class StarPattern extends PatternPrinter {

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 {

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.err.println("S.Saivenkat, 24322, CSE-B ");

Scanner input = new Scanner(System.in);

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

int rows = input.nextInt();

PatternPrinter star = new StarPattern();

star.displayTitle("Star Pattern:");

star.printPattern(rows)

PatternPrinter number = new NumberPattern();

number.displayTitle("Number Pattern:");

number.printPattern(rows);

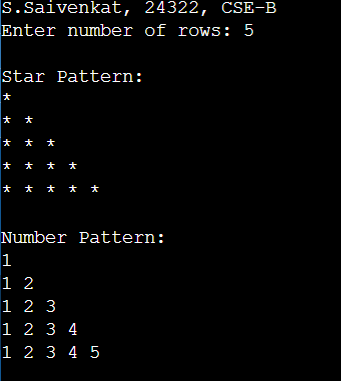
input.close();

}}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| s.no | ERRORS | Error rectification |
| 1. | Output lacks spacing between patterns for clarity. | Add a  System.out.println(); between the two patterns. |

**Output:**



**IMPORTANT POINTS:**

Method Overriding  
 print Pattern () is overridden in both Star Pattern and Number Pattern.

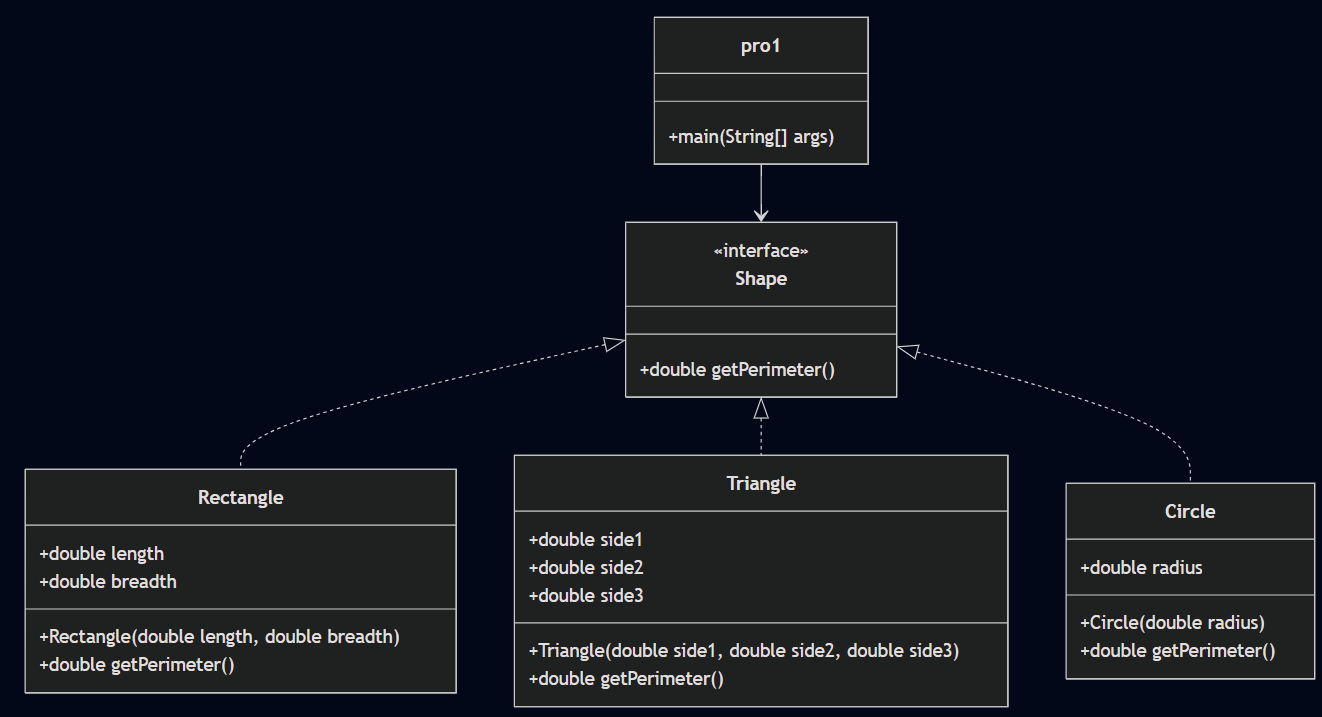
Polymorphism  
 Parent reference (Pattern Printer) is used to call subclass methods.

**WEEK 8**

**Program 1**

**AIM:** Write a java program creating an interface Shape with the get perimeter method create 3 classes rectangle, triangle and circle that implements the shape interface, implement the getperimeter method for each of the three classes.

Class Diagram:



**Code:**

interface Shape{

    double getPerimeter();

}

class Rectangle implements Shape{

    double length;double breadth;

    Rectangle(double length,double breadth){

        this.length=length;

        this.breadth=breadth;

    }

    public double getPerimeter(){

        return 2\*(length+breadth);

    }

}

class Triangle implements Shape{

double side1;double side2;double side3;;

    Triangle(double side1,double side2,double side3){

        this.side1=side1;

        this.side2=side2;

        this.side3=side3;

    }

    public double getPerimeter(){

        return side1+side2+side3;

    }

}

class Circle implements Shape{

double radius;

    Circle(double radius){

        this.radius=radius;

    }

    public double getPerimeter(){

        return 2\*3.14\*radius;

    }

}

class pro1{

    public static void main(String[] args) {

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        Shape Rectangle=new Rectangle(2,6);

        Shape Triangle=new Triangle(3,4,5);

        Shape Circle=new Circle(2);

        System.err.println("Perimeter of Rectangle= "+Rectangle.getPerimeter());

        System.err.println("Perimeter of Triangle= "+Triangle.getPerimeter());

        System.err.println("Perimeter of Circle= "+Circle.getPerimeter());

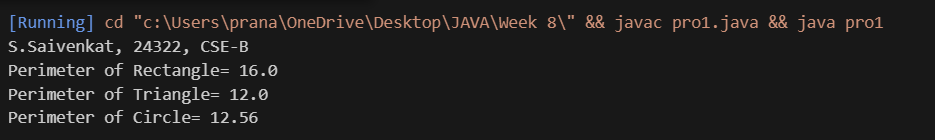
    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.No | ERRORS | Error rectification |
| 1. | Declared getPerimeter method as void. | Rectified it by changing it to double. |

**Output:**



**IMPORTANT POINTS:**

Shape is an interface with a method double getPerimeter().

Rectangle, Triangle, and Circle implement the Shape interface.

Method getPerimeter() returns a double value, not void.

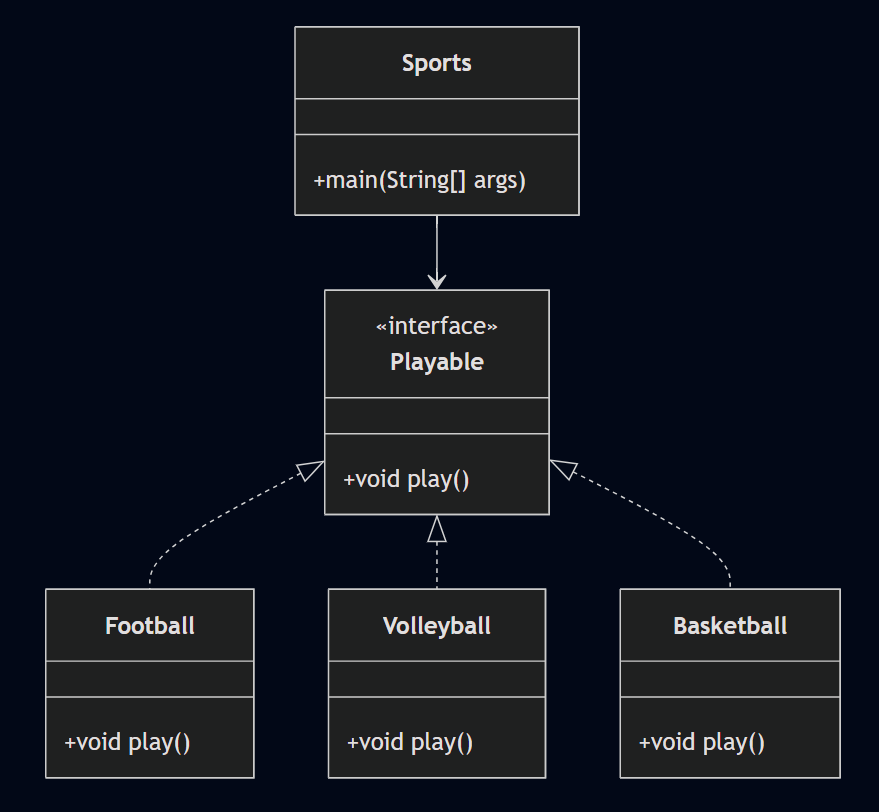
Each shape class provides its own logic to calculate perimeter.

Constructors are used to initialize dimensions of each shape.

**PROGRAM 2**

**AIM:** Write a java program to create an interface playable with a method play ()

That takes no arguments and returns void create three classes football, volleyball and basketball that implements the playable and override the play method to play the respective sports.

**Class Diagram:** 

**Code:**

interface Playable{

    void play();

}

class Football implements Playable{

    public void play(){

        System.out.println("Ronaldo is one of the greatest football players");}

}

class Volleyball implements Playable{

    public void play(){

    System.out.println("Gilberto Amauri de Godoy Filho is one of the greatest volleyball players");}

}

class Basketball implements Playable{

    public void play(){

        System.out.println("Micheal Jordan is one of the greatest Basketball players");}

}

class Sports{

    public static void main(String[] args){

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        Playable f=new Football();

        Playable v=new Volleyball();

        Playable b=new Basketball();

        f.play();

        v.play();

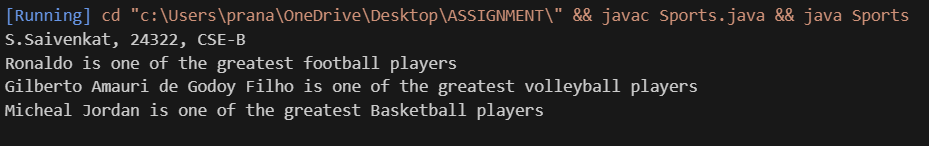
        b.play();

    }

}

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.No | Error | Rectification |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

**Output:** 

**IMPORTANT POINTS:**

Playable Interface:

The Playable interface defines a contract for any class that wants to implement it. It requires the play() method, which is intended to describe an action related to playing a sport.

Football, Volleyball, and Basketball Classes:

Each of these classes implements the Playable interface.

Polymorphism:

In the Sports class, polymorphism is demonstrated. The Playable reference type is used to refer to objects of different classes (Football, Volleyball, and Basketball).

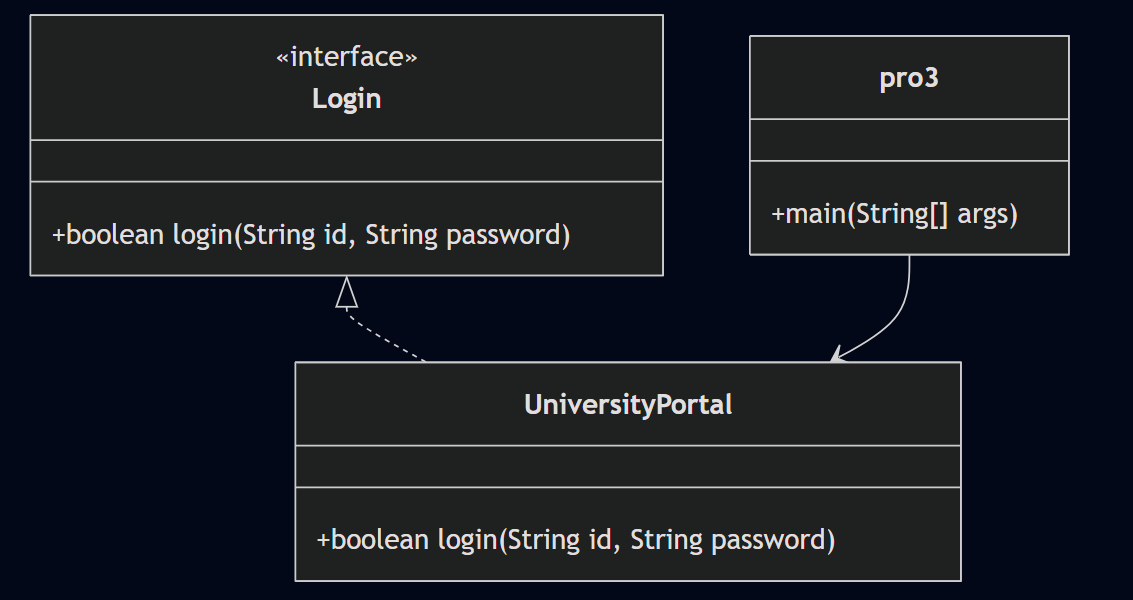
Despite being of different types, each object calls the play() method, but the output differs based on the specific class implementation.

Main Method:

In the main() method, instances of Football, Volleyball, and Basketball are created.

PROGRAM 3

AIM:Write a java program to implement a login system using interfaces.

Class Diagram: 

**Code:**

interface Login {

    boolean login(String id, String password);

}

class UniversityPortal implements Login {

    public boolean login(String id, String password) {

        if (id.equals("student123") && password.equals("pass123")) {

            return true;

        } else {

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

            return false;

        }

    }

}

public class pro3 {

    public static void main(String[] args) {

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        UniversityPortal portal = new UniversityPortal();

        boolean loginSuccess1 = portal.login("student123", "pass123");

        System.out.println("Login successful: " + loginSuccess1);

        boolean loginSuccess2 = portal.login("student123", "wrong pass");

        System.out.println("Login Successful: " + loginSuccess2);

    }

}

Error Table:

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| 1 | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| 2 | } | Ending the class and main method is required |

Output: 

IMPORTANT POINTS

Login is an interface with a method Boolean login (String id, String password).

UniversityPortal class implements the Login interface.

The login method checks if the provided credentials match hardcoded values.

If credentials are correct, it returns true.

If credentials are incorrect, it prints "Invalid Credentials" and returns false.

The main method demonstrates two login attempts: one successful and one failed.

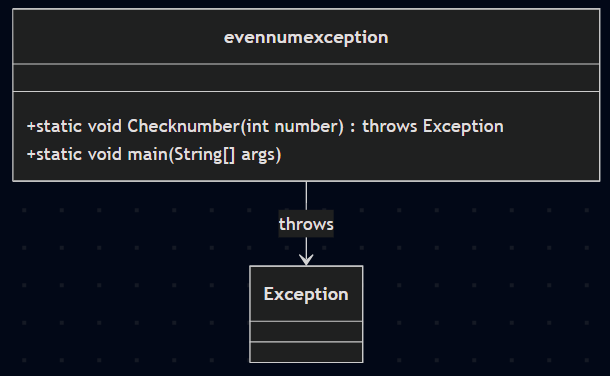
Return type Boolean is used to indicate login success or failure.

**WEEK 9**

**Program 1**

**AIM:** Write a Java program to create a method that takes integer as parameter and throws an exception if the number is even.

**Class Diagram:**



**Code:**

public class evennumexception {

    public static void Checknumber(int number)

    throws Exception {

        if (number % 2 == 0) {

            throw new Exception("It is an even number: " + number);

        } else {

            System.out.println("It is an odd number: " + number);

        }

    }

    public static void main(String[] args) {

        try {

            System.out.println("S.Saivenkat, 24322, CSE-B ");

            Checknumber(14);

        } catch (Exception e) {

            System.out.println(e.getMessage());

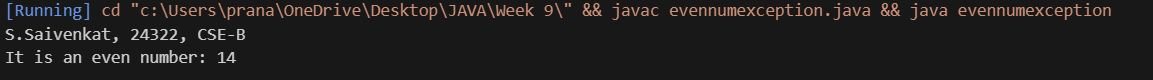
        }

    }

}

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Expected Error | Reason |
| 1 | Types “throw” instead of “throws” | Replaced “throw” with “throws” |
| 2 | Missed “}” | Ending the main class and main method is required |

**Output:** 

**IMPORTANT POINTS:**

Class names should use PascalCase (e.g., EvenNumException)

Method names should use camelCase (e.g., checkNumber) Method throws an exception if the number is even

Uses try-catch block in main to handle the exception

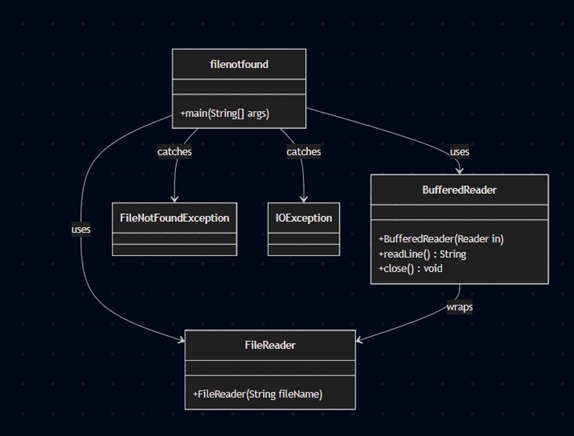
Outputs message based on whether number is even or odd

Uses getMessage() to display the exception message

**Program 2**

**AIM:** Write a Java program to create a method that reads a file and throws an exception if the file is not found exception.

**Class Diagram:**



**Code:**

import java.io.\*;

public class filenotfound {

    public static void main(String[] args) {

        try {

            BufferedReader br = new BufferedReader(new FileReader("C:/Users/prana/OneDrive/Desktop/1.txt"));

            String line;

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

                System.out.println(line);

            }

            br.close();

        } catch (FileNotFoundException e) {

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

        } catch (IOException e) {

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

        }

    }

}

**Error Table:**

|  |  |  |
| --- | --- | --- |
| S.no | Error | Reason |
| 1 | Did not change “/” to “\” | Replaced “/ with “\” |
| 2 | Missed “}” | Ending the main class and main method is required |

If file exists:

Output:



If file does not exist:

Output:



IMPORTANT POINTS:

Check if the file exists at the specified path

Verify the file path is correct and has no typos

Ensure the file has the correct extension (e.g. not 12.txt.txt)

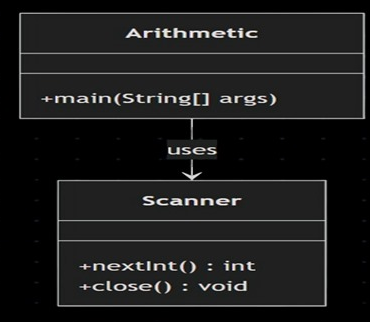
Use double backslashes if using backslash in the path

Make sure the file is not open or locked by another application

**Program 3**

**AIM:** Write a Java program to handle arithmetic exception using try, catch, and finally block.

**Class Diagram:**

****

**Code:**

import java.util.Scanner;

public class airthmetic{

    public static void main(String[] args) {

        Scanner Scanner=new Scanner(System.in);

        try {

            System.err.print("Enter first (numerator): ");

            int a=Scanner.nextInt();

            System.err.print("Enter first (denominator): ");

            int b=Scanner.nextInt();

            int result=a/b;

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

        }

        catch (Exception e) {

            System.err.println("An error occured");

        }

        finally{

            System.err.println("finally");

        }

        Scanner.close();

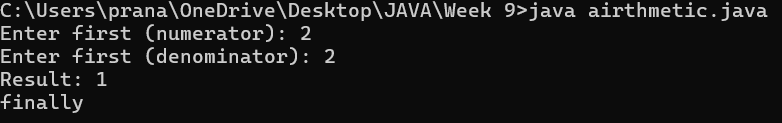
    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.no | Error | Reason |
| 1. | Missed “;” in first line | Ended the statement with “;” |

Output:



**IMPORTANT POINTS:**

Import Scanner class to read user input

Create a Scanner object to take input from the console

Use try block to wrap code that might throw an exception

Read numerator and denominator using scanner.nextInt()

Perform division inside try block (this might cause ArithmeticException)

Use catch block to handle exceptions and display an error message

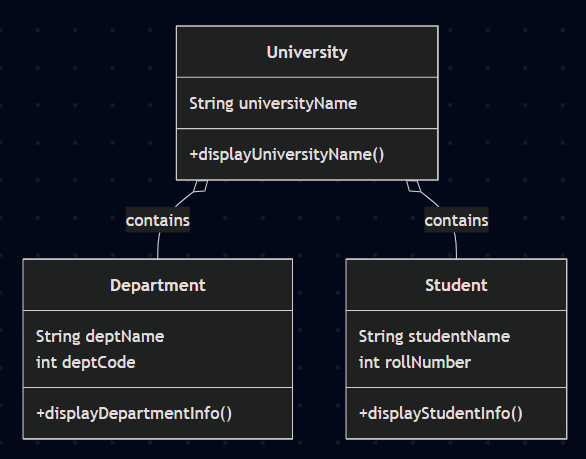
catch block uses Exception class, which is a general exception handler

Use finally block to execute code regardless of whether an exception occurred

**Program 4**

**AIM:** Write a Java program to simulate a university system using inner class

* Create an outer class named University with a variable universityName
* Inside it define two non-static inner class:
  1. Department – with variables like deptName and deptCode and a method to display department details
  2. Student – with variables like studentName and rollNumber and a method to display student details
* Create an object for each class and call their methods to display the details along with the university name

Class Diagram: 

**Code:**

public class university {

    String UniversityName = "Amrita Vishwa Vidyaapetham";

    void name() {

        System.out.println("University Name: " + UniversityName);

    }

    class Department {

        String DeptName = "Computer Science";

        int Deptcode = 101;

        void Departmentinfo() {

            System.err.println("Department Name: " + DeptName);

            System.err.println("Department Code: " + Deptcode);

        }

    }

    class Student {

        String Studentname = "Yuvraj";

        int RollNumber = 12;

        void Studentinfo() {

            System.err.println("Student Name: " + Studentname);

            System.err.println("Student Code: " + RollNumber);

        }

    }

    public static void main(String[] args) {

System.err.println("S.Saivenkat, 24322, CSE-B ");

        university u = new university();

        u.name();

        university.Department dept = u.new Department();

        dept.Departmentinfo();

        university.Student student = u.new Student();

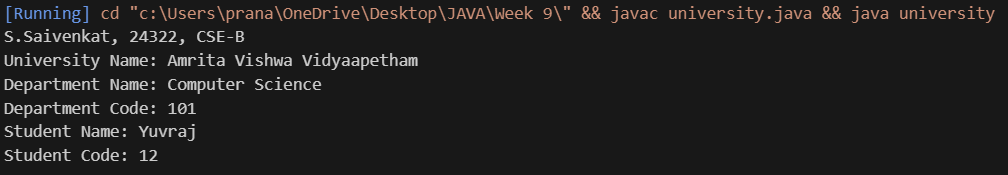
        student.Studentinfo();

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.no | Error | Reason |
| 1. | Missed “;” in 2nd line | Ended the statement with “;” |

Output: 

IMPORTANT POINTS:

The outer class is named University

It has a member variable universityName and a method to display it

Two non-static inner classes are defined inside the outer class: Department and Student

Inner class Department contains deptName, deptCode, and a method to display them

Inner class Student contains studentName, rollNumber, and a method to display them

Non-static inner classes require an instance of the outer class to be created

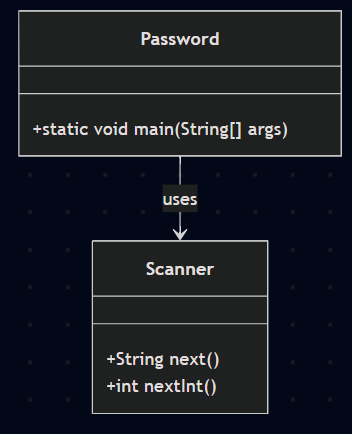
Inner class objects are created using outerClassInstance.new InnerClass() syntax

**Week 10**

**Program 1**

**AIM:** Write a java program to generate a password for a student using his/her initials and age. the password displayed should be the string consists of first character of first name, middle name, last name with age.

**Class Diagram:**



**Code:**

import java.util.Scanner;

class Password {

    public static void main(String[] args) {

        System.err.println("S.Saivenkat, 24322, CSE-B ");

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your first name: ");

        String firstName = scanner.next();

        System.out.print("Enter your middle name: ");

        String middleName = scanner.next();

        System.out.print("Enter your last name: ");

        String lastName = scanner.next();

        System.out.print("Enter your age: ");

        int age = scanner.nextInt();

        char fname=firstName.charAt(0);

        char mname=middleName.charAt(0);

        char lname=lastName.charAt(0);

        System.err.println("Password is: "+fname+mname+lname+age);

    }

}

ERROR TABLE:

|  |  |  |
| --- | --- | --- |
| S.No | ERRORS | Error rectification |
| 1. | Forget to create a scanner object. | Created a scanner object named “scanner”. |

Output:



**IMPORTANT POINTS:**

The program is encapsulated in a class called Password (following Java naming conventions with a capitalized class name).

The program uses Scanner to accept user input for the first name, middle name, last name, and age

charAt(0) is used to get the first letter (initial) of each name.

The password is generated by concatenating the initials (first character) of the first, middle, and last names along with the user's age.

**Program 2**

**AIM:** Design and implement a Java program that will do the following operations to this string "Welcome! You are practicing strings concept."

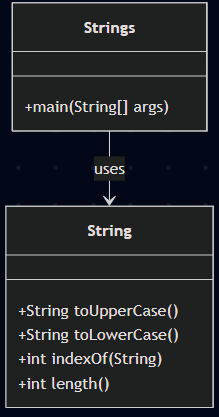
a. convert all alphabets to capital letters and print out the result.

b. convert all alphabets to lower-case letters and print out the result.

c. print out the length of the string.

d. print out the index of Course.

**Class Diagram:**



**Code:**

class Strings{

    public static void main(String[] args) {

System.out.println("S.Saivenkat, 24322, CSE-B ");

        String string=" Welcome! You are practicing strings concept.";

        String UC=string.toUpperCase();

        System.err.println("Converted all alphabets to upper-case letters."+" "+UC);

        String LC=string.toLowerCase();

        System.err.println("Converted all alphabets to lower-case letters."+" "+LC);

        int index=string.indexOf("Course");

        System.err.println("Index of Course is: "+index);

        int length=string.length();

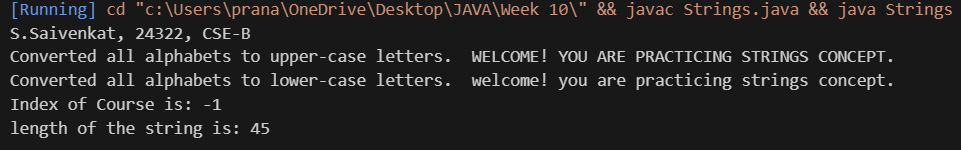
        System.err.println("length of the string is: "+length);

    }}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.No | ERRORS | Error rectification |
| 1. | Used “uppercase” instead of “UpperCase” | Replaced it. |
| 2. | Used “lowercase” instead of “LowerCase” | Replaced it. |

**Output:**



**IMPORTANT POINTS:**

Strings in Java are immutable.

Use toUpperCase() to convert all letters to uppercase.

Use toLowerCase() to convert all letters to lowercase.

Use length() to find the total number of characters in a string.

Use indexOf("substring") to find the starting index of a substring.

**Program 3**

**AIM**: Implement a java program using the below array methods.

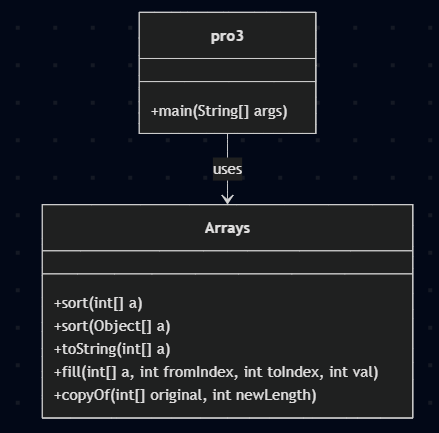
a. Sorting the elements (numbers and Strings) of an array.

b. convert the array elements into string.

c. fill the part of an array.

d. copy the elements of one array into another.

**Class Diagram:**



**Code:**

import java.util.Arrays;

public class pro3 {

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B ");

        int[] numbers = {45,33,18, 1, 93};

        String[] names = {"Rohit","Virat","Jasprit","Hardik","Rahul"};

        Arrays.sort(numbers);

        Arrays.sort(names);

        System.out.println("Sorted numbers: " + Arrays.toString(numbers));

        System.out.println("Sorted names: " + Arrays.toString(names));

        String numberString = Arrays.toString(numbers);

        System.out.println("Array elements as string: " + numberString);

        int[] filledArray = new int[5];

        Arrays.fill(filledArray, 1, 4, 8);

        System.out.println("Partially filled array: " + Arrays.toString(filledArray));

        int[] sourceArray = {10, 23, 3};

        int[] destinationArray = Arrays.copyOf(sourceArray, sourceArray.length);

        System.out.println("Copied array: " + Arrays.toString(destinationArray));

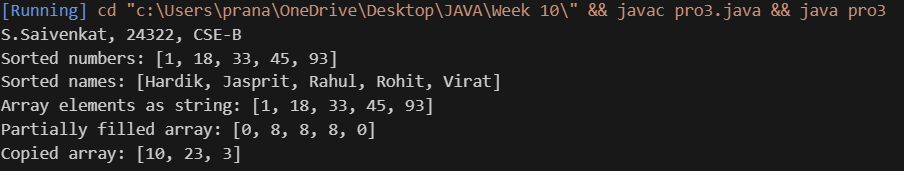
    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.No | ERRORS | Error rectification |
| 1. | Did not place “;” after importing array package. | Added “;” |

**Output:**



**IMPORTANT POINTS:**

Use Arrays.sort() to sort numeric and string arrays in ascending order.

Use Arrays.toString() to convert an array into a printable string format.

Use Arrays.fill(array, fromIndex, toIndex, value) to fill part of an array with a specific value.

Use Arrays.copyOf(originalArray, length) to copy one array into another.

The fromIndex in fill() is inclusive; toIndex is exclusive.

All arrays of primitive types are initialized with default values (0 for int).

**Program 4**

**AIM:** Implement a java program using the below Array List methods.

a. insert an element at particular index in the array list.

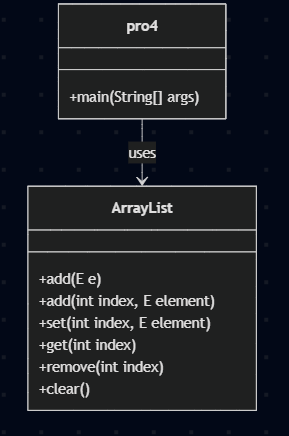
b. Modify an element in the array list.

c. Access an element from the array list.

d. Remove an element from the array list.

e. clear the elements from the array list.

**Class Diagram:**



**Code:**

import java.util.ArrayList;

public class pro4{

    public static void main(String[] args) {

        System.out.println("S.Saivenkat, 24322, CSE-B 5");

        ArrayList<String> cricketers = new ArrayList<>();

        cricketers.add("Rohit Sharma");

        cricketers.add("Virat Kohli");

        cricketers.add("Shikhar Dhawan");

        cricketers.add(1, "Jasprit Bumrah");

        System.out.println("After insertion: " + cricketers);

        cricketers.set(2, "Ravindra Jadeja");

        System.out.println("After modification: " + cricketers);

        String cricketerAtIndex1 = cricketers.get(1);

        System.out.println("Element at index 1: " + cricketerAtIndex1);

        cricketers.remove(3);

        System.out.println("After removal: " + cricketers);

        cricketers.clear();

        System.out.println("After clearing: " + cricketers);

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S. No | ERRORS | Error rectification |
| 1. | Did not place “;” after importing array package. | Added “;” |

Output: 

**IMPORTANT POINTS:**

ArrayList Creation ArrayList<String> cricketers = new ArrayList<>();

Inserting an Element cricketers.add(1, "Jasprit Bumrah");

Modifying an Element cricketers.set(2, "Ravindra Jadeja");

Accessing an Element String cricketerAtIndex1 = cricketers.get(1);

Removing an Element cricketers.remove(3);

Clearing the List cricketers.clear()

.get(1);

Removing an Element cricketers.remove(3);

Clearing the List cricketers.clear();