**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: Y Yashaswi Reddy**

**Verified By Roll No: av.sc.u4cse24351**

**WEEK 01**

**PROGRAM-1:**

**AIM:** Download and Install Java Software

**PROCEDURE:**

**Step 1: Download JDK 21**

1. Open your web browser and go to the Oracle JDK Downloads page
2. Scroll down to the Java SE Development Kit 21 section.
3. Choose the Windows x64 Installer version.
4. Click on Download, then Wait for the download to complete



**Step 2:** **Install JDK 21**

1. Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click Next on the setup wizard.
4. Choose the installation path (default is C:\Program Files\Java\jdk-21).
5. Click Next, then click Install.
6. Wait for the installation to complete.
7. Click Close once the installation is finished.



**Step 3: Setting up the path**

1) Go to “Windows C” Drive on Desktop

2) Choose Program Files, select Java, then JDK 21, then select Bin.

3) Select and copy the path at the address bar.



**Step 4: Open System Properties**

1. Press Windows + R, type sysdm.cpl , and click Ok-
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.



**Step 5: Set JAVA\_HOME**

1)Under System Variables, click New.

2)Set the Variable name as JAVA\_HOME.

3)Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).

4)Click OK.



**Step 6: Update PATH Variable**

1)In System Variables, find Path and click Edit.

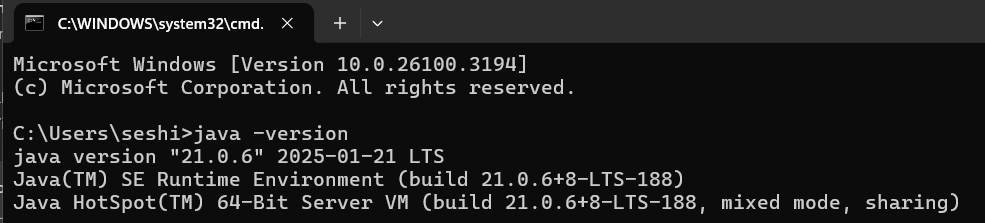
2)Click New and add: C:\Program Files\Java\jdk-21\bin

3)Click OK to save.



**Step 7:Verify Installation**

1. Open Command Prompt.
2. Type the following command: **java --version** and press Enter.



1. To check the java compiler type: **javac –version.**

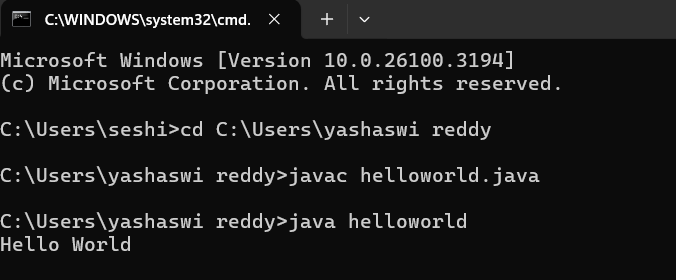
**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**



**OUTPUT:**



**ERRORS:** None found

IMPORTANT POINTS:

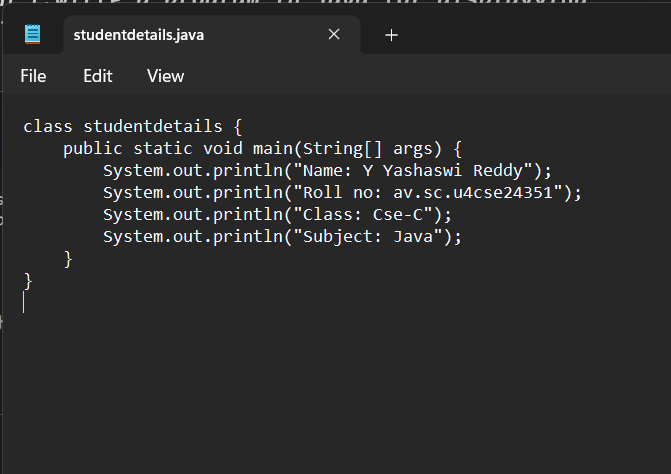
1. Make sure that the file and the class name are the same to avoid confusion.

**PROGRAM-3:**

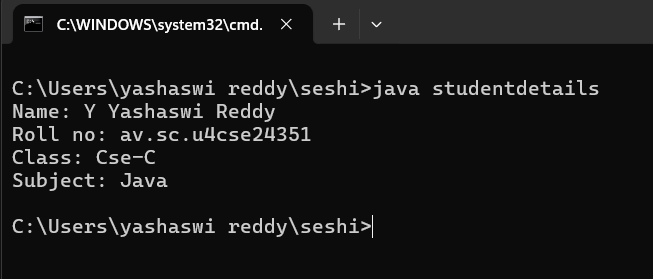
**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

**STUDENT DETAILS:**

**

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. writing small “S”in place of”S”   In system.out.println()  2)not giving strings to the name and scetion | 1. code is rectified by keeping capital “S” 2. Giving strings to name and section |

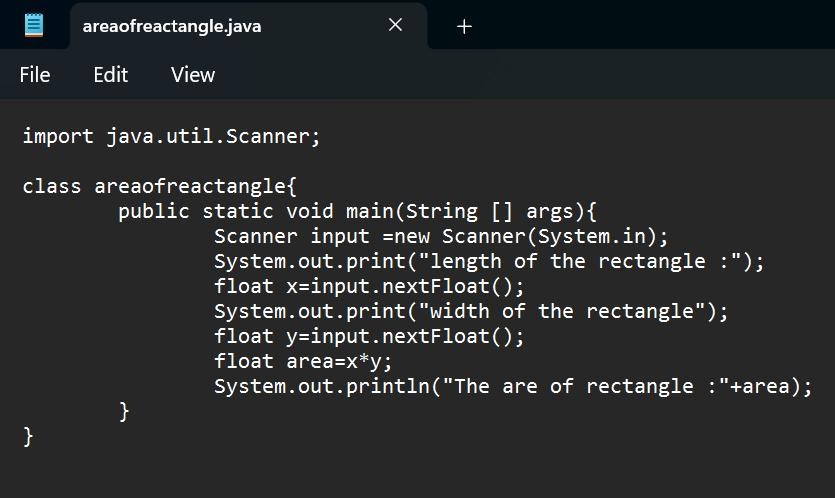
IMPORTANT POINTS:

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

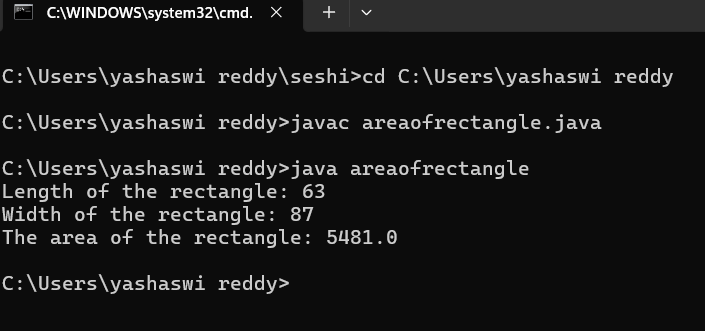
**WEEK-2:**

**1.AIM:** Write a Java Program for calculating area of rectangle

**PROGRAM:**

**

**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

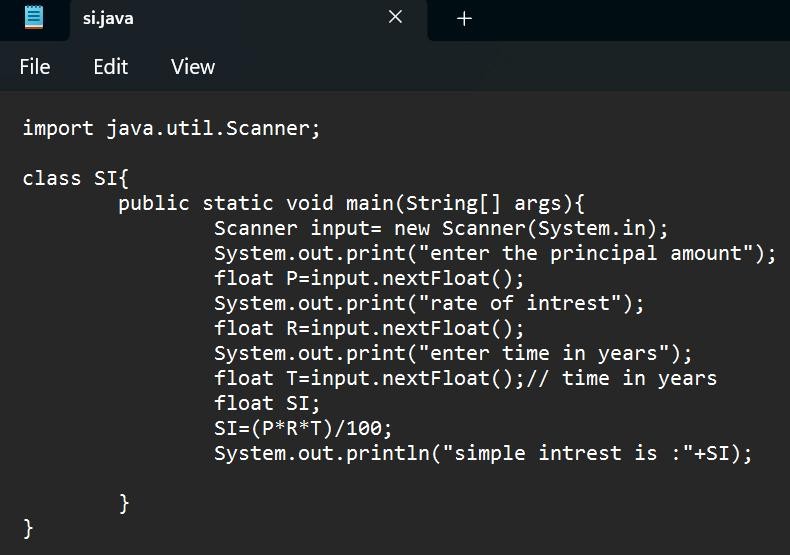
L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

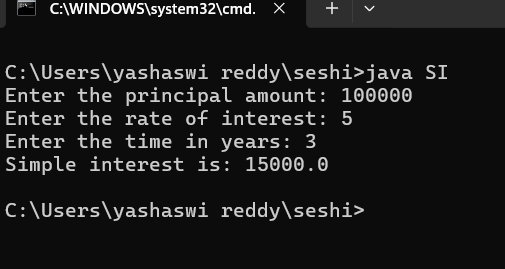
1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**2.AIM:** Write a Java Program for calculating simple intrest

**PROGRAM:**

**

**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:

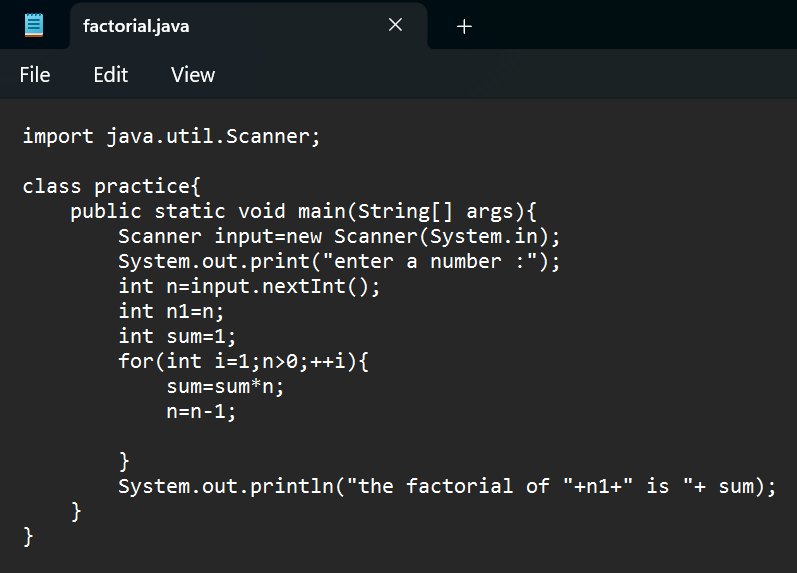
Import: tells the java compiler that we want to use a specific class or package in your code.

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

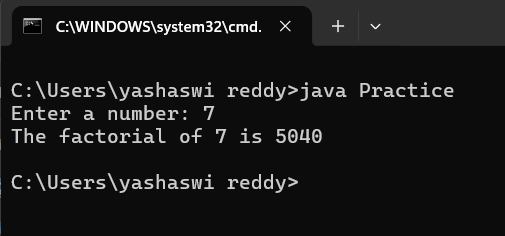
Scanner: this is the class that allows you to read input from the keyboard.

**3.AIM:** Write a Java Program for finding factorial of a number

**PROGRAM:**

**

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

Test expression and

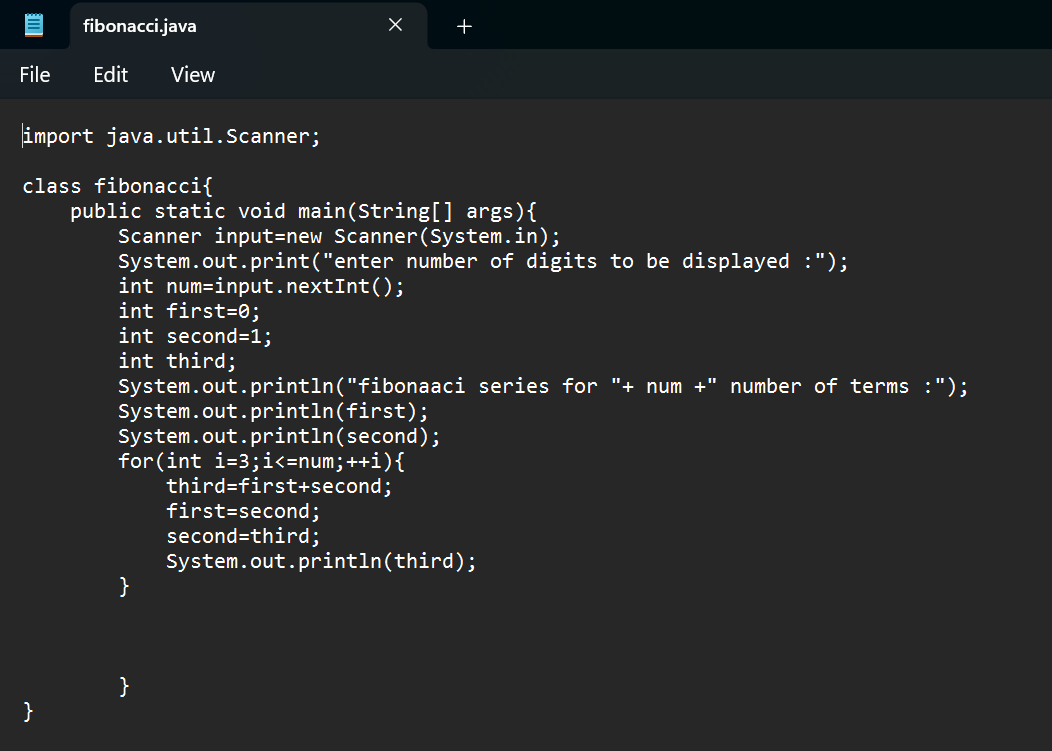
Update expression.

1. Here “factorial\*=I” means factorial = factorial\*I.

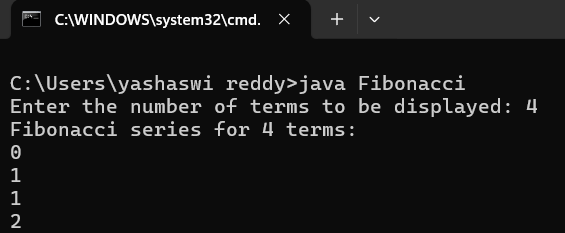
Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points

**4.AIM:** Write a Java Program for finding Fibonacci series of a number

**PROGRAM:**

**

**OUTPUT:**

**

ERROR TABLE:

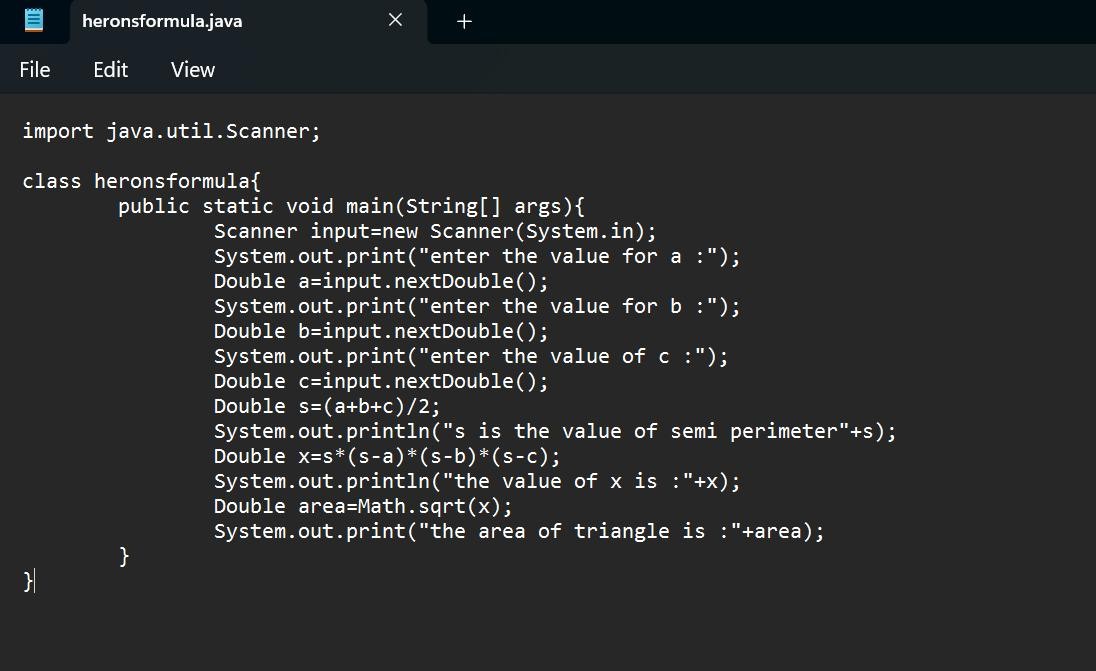
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not giving parenthesis after closing the input. | 1.Should not give space between next and Double.  2.We must put parenthesis after closing the input. |

IMPORTANT POINTS:

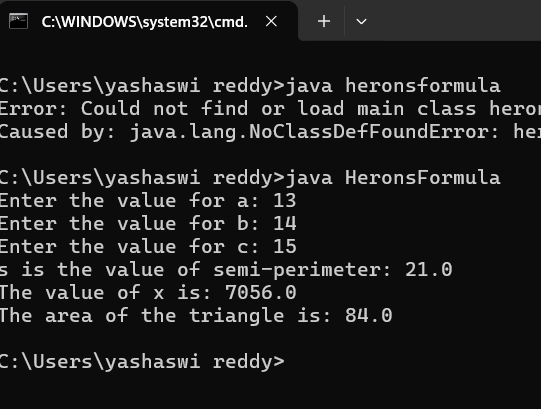
1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

**5.AIM**: Write a Java Program for calculating area of triangle using herons formula

**PROGRAM:**



**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. 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

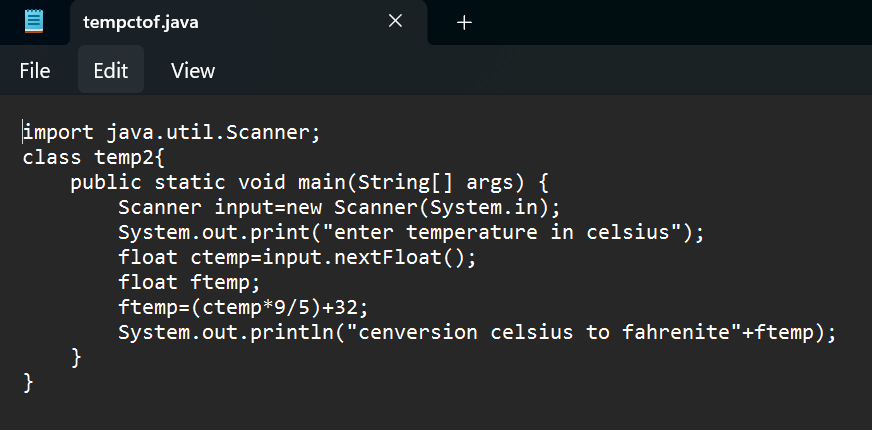
Where S is the semi-perimeter of the triangle.

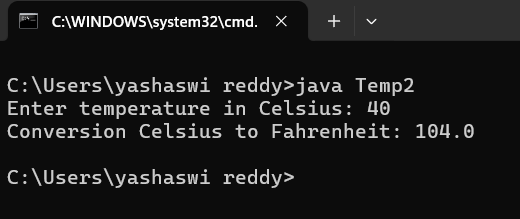
Now the area formula is:

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

**6. I.) AIM:** Write a Java Program for conversion from Celsius to farenheit

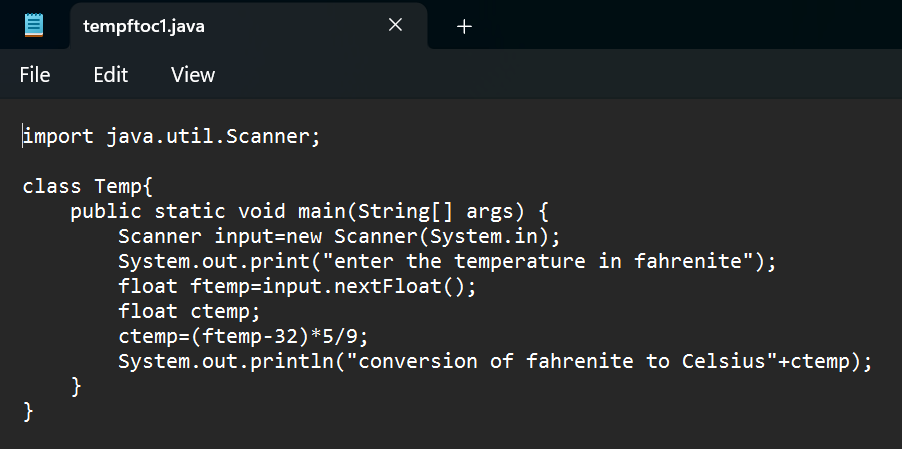
**PROGRAM:**

****OUTPUT:**

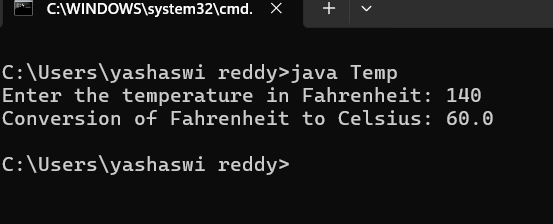
******

**II.) AIM:** Write a Java Program for conversion from farenheit to celsius

**PROGRAM:**

******

**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. The formula to convert a Fahrenheit to Celsius is

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

1. The formula to convert a Celsius to Fahrenheit is

Fahrenheit = (Celsius\*9/5)+32.

1. The line “Scanner input = new Scanner(System.in),” tends to create a new Scanner object named “input” that reads input from the standard input stream (System.in), like keyboard.

**WEEK -3:**

**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

**PROGRAM:**

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";

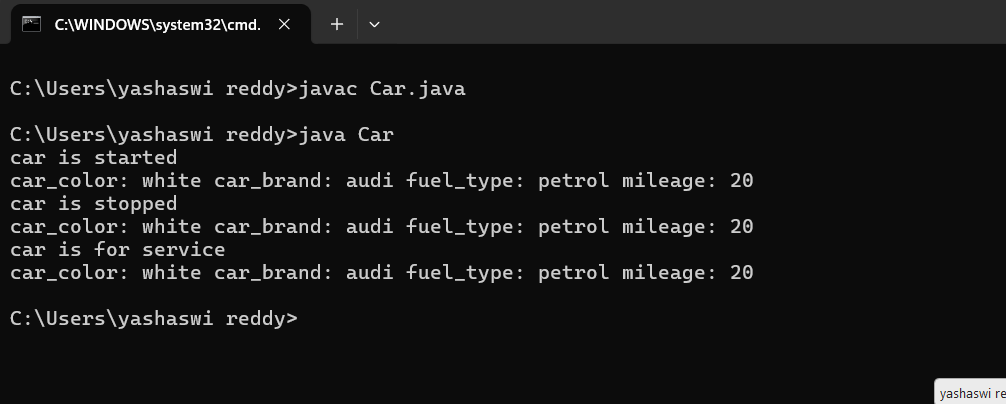
car2.service();

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

}

}

**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Start, Stop, Service not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

IMPORTANT POINTS:

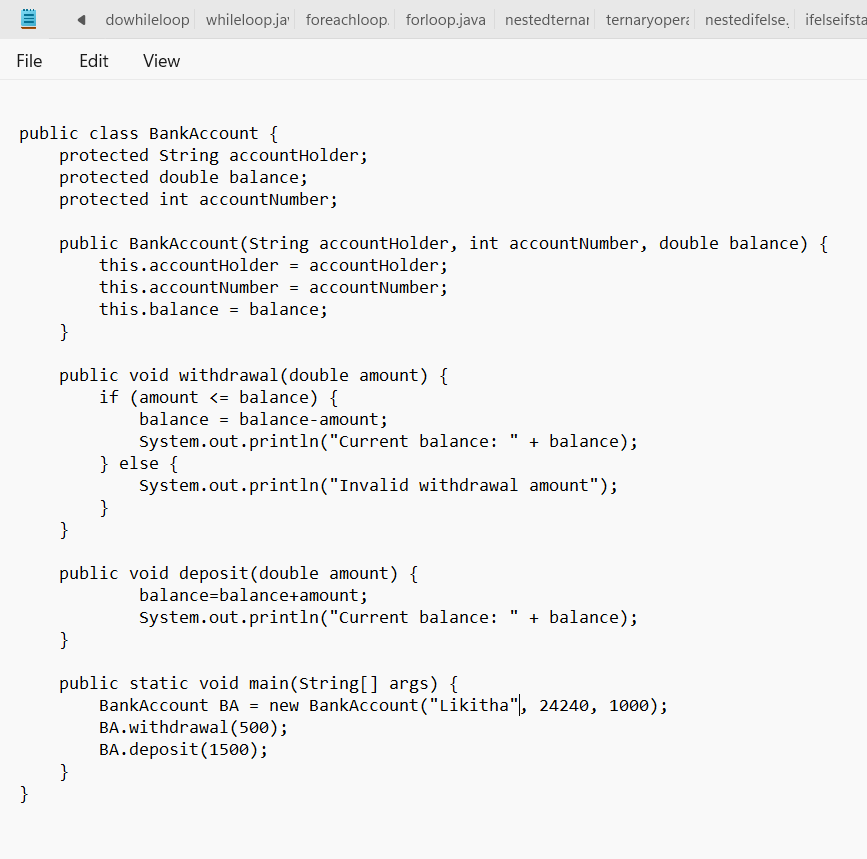
1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function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

CLASS DIAGRAM:

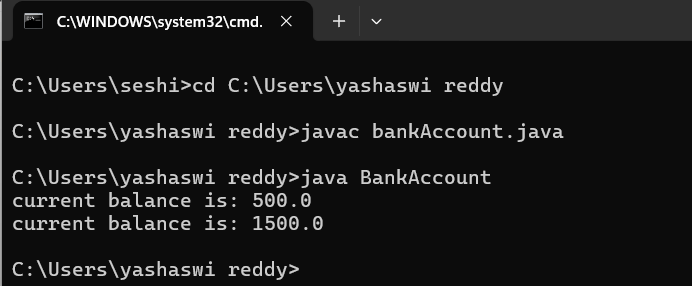
|  |
| --- |
| car  -----------------------  -car\_color:string  -car\_brand:string  -fuel\_type:string  -milage:double  ----------------------  +start():void  +stop():void  +service():void |

**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

**PROGRAM:**

****

**OUTPUT:**



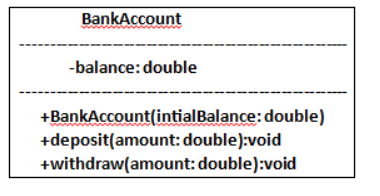
ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Withdrawal,deposit not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

IMPORTANT POINTS:

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

CLASS DIAGRAM:



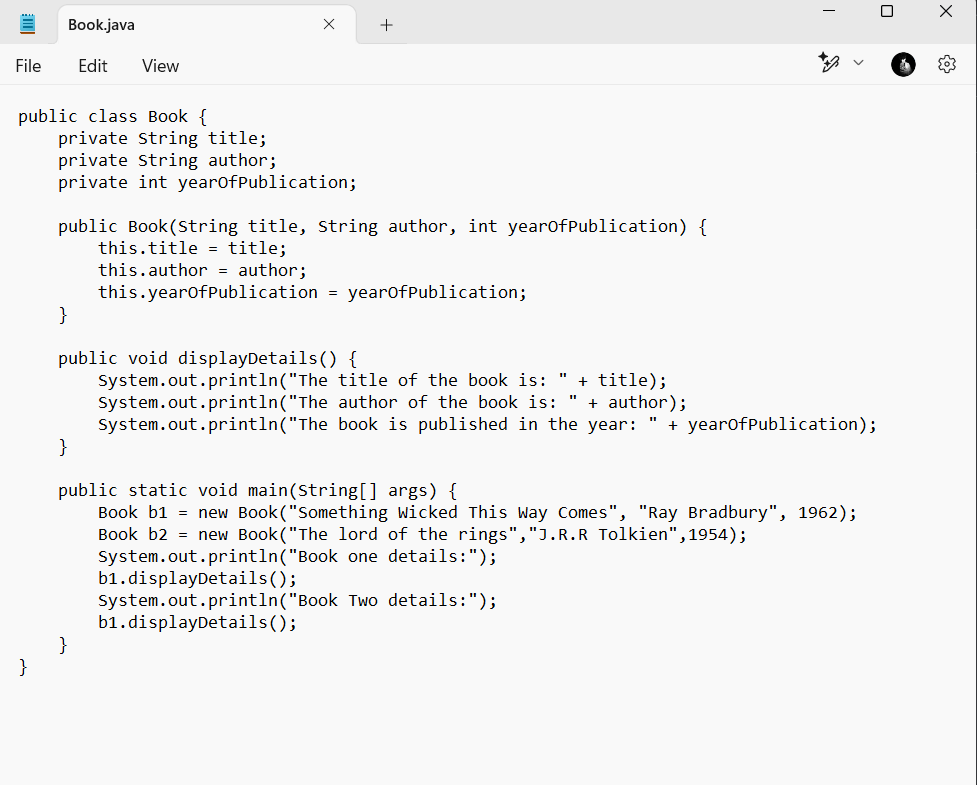
WEEK -4

PROGRAM – 1:

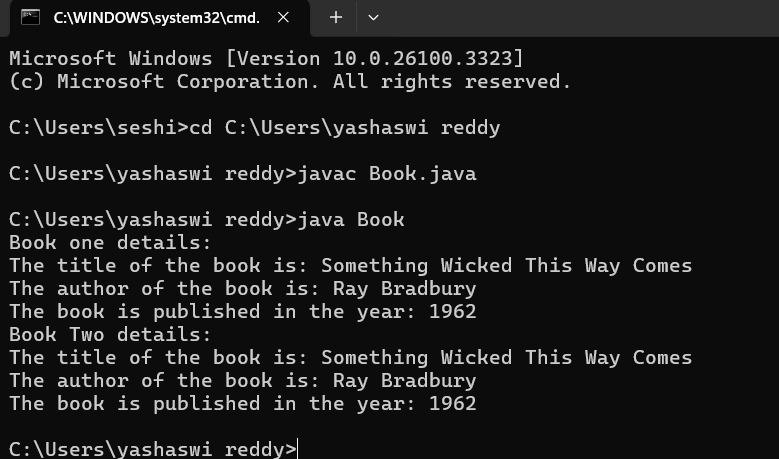
AIM: Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

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.

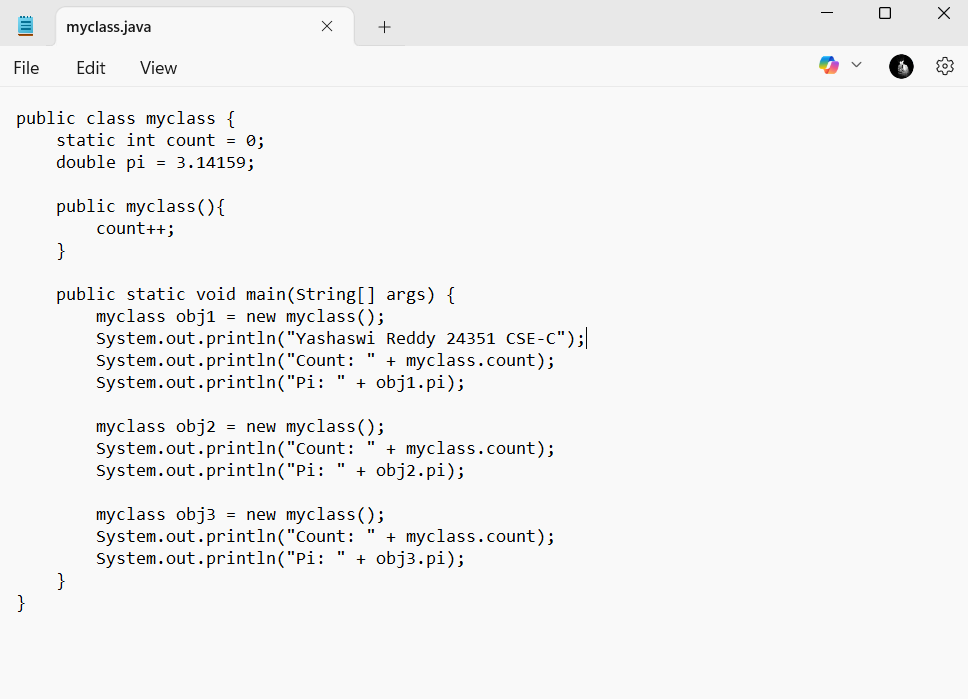
CLASS DIAGRAM:

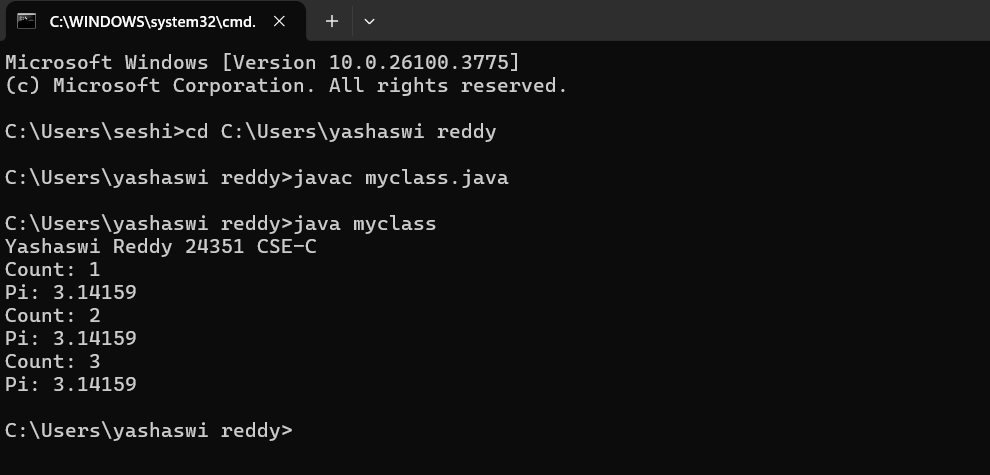
|  |
| --- |
| Book   * Title: String * Author: String * Year of publication: int   + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

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

**1.** **Create a calculator using the operations including**

**addition, subtraction, multiplication and division using**

**multilevel in heritance and display the desired output**

**Code:**

**import java.util.Scanner;**

**class Calculator {**

**public void add(double a, double b) {**

**System.out.println("Sum of two numbers is: " + (a + b));**

**}**

**public void subs(double a, double b) {**

**System.out.println("Difference of two numbers is: " + (a - b));**

**}**

**}**

**class Calculator1 extends Calculator {**

**public void mul(double a, double b) {**

**System.out.println("Multiplication of two numbers is: " + (a \* b));**

**}**

**}**

**class Calculator2 extends Calculator1 {**

**public void div(double a, double b) {**

**if (b == 0) {**

**System.out.println("Division is not allowed when b = 0");**

**} else {**

**System.out.println("Division of two numbers is: " + (a / b));**

**}**

**}**

**}**

**public class Cals {**

**public static void main(String[] args) {**

**Calculator2 C = new Calculator2();**

**Scanner input = new Scanner(System.in);**

**System.out.println("Yashaswi reddy 24351 CSE-C”)**

**System.out.println("Enter two numbers:");**

**double a = input.nextDouble();**

**double b = input.nextDouble();**

**C.add(a, b);**

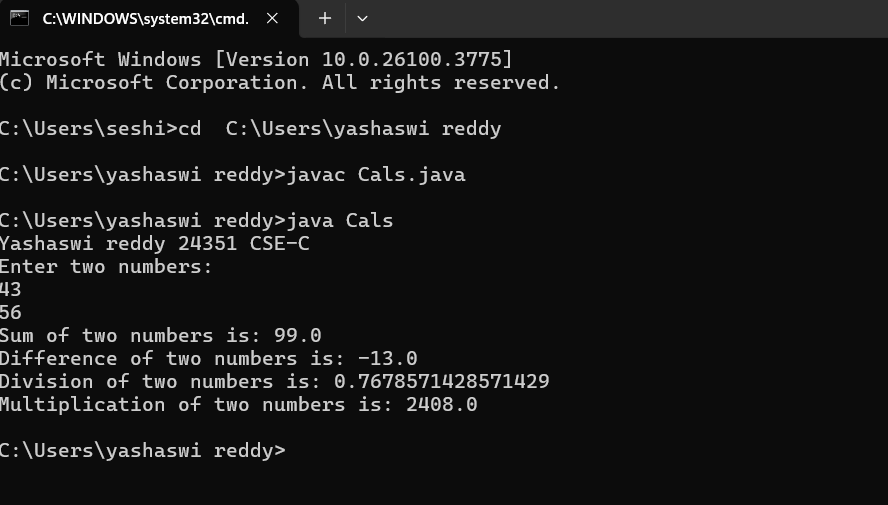
**C.subs(a, b);**

**C.div(a, b);**

**C.mul(a, b);**

**}**

**}**

****

**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

**Class Diagram:**

**+-----------------------------+**

**| Calculator |**

**+-----------------------------+**

**| + add(double, double): void |**

**| + subs(double, double): void|**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator1 |**

**+-----------------------------+**

**| + mul(double, double): void |**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator2 |**

**+-----------------------------+**

**| + div(double, double): void |**

**+-----------------------------+**

**+-----------------------------+**

**| Cals |**

**+-----------------------------+**

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

**+-----------------------------+**

**2.** **A vehicle rental company wants to develop a system**

**that maintains information about different types of vehicles**

**available for rent. The company rents out cars and bikes and**

**they need a program to store details about each vehicle**

**such as brand and speed.**

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

**Seating capacity.**

**ii. Bikes should have a property indicating whether they have**

**gears or not.**

**iii. The system should also include a function to display**

**details about each vehicle and indicate when a vehicle is**

**starting.**

**iv. Each class should have a constructor.**

**Questions:**

**1. Which OOP concept is used in the above program?**

**Explain why it is useful in this scenario.**

**2. If the company decides to add a new type of vehicle**

**‘Truck’, how would you modify the program?**

**a. Truck should include and additional property capacity**

**(in tons).**

**36**

**b. Create a showTruck() method to display the truck’s**

**capacity.**

**c. Write a constructor for truck that initializes all properties.**

**3. Implement the truck class and update the main method to**

**create a Truck object and also create an object for car and**

**bike subclasses. Finally display the details**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**Vehicle(String brand, int speed) {**

**this.brand = brand;**

**this.speed = speed;**

**}**

**void Details() {**

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

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

**}**

**}**

**class Cars extends Vehicle {**

**int doors;**

**int capacity;**

**public Cars(String brand, int speed, int doors, int capacity) {**

**super(brand, speed);**

**this.doors = doors;**

**this.capacity = capacity;**

**}**

**void cardetails() {**

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

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

**}**

**}**

**class Bikes extends Vehicle {**

**boolean gears;**

**Bikes(String brand, int speed, boolean gears) {**

**super(brand, speed);**

**this.gears = gears;**

**}**

**void bikedetails() {**

**if (gears)**

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

**else**

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

**}**

**}**

**class Trucks extends Vehicle {**

**int tons;**

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

**super(brand, speed);**

**this.tons = tons;**

**}**

**void truckdetails() {**

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

**}**

**}**

**public class Rent {**

**public static void main(String[] args) {**

**Cars c = new Cars("Toyota", 120, 5, 5);**

**c.cardetails();**

**c.Details();**

**Bikes b = new Bikes("KTM", 80, true);**

**b.bikedetails();**

**b.Details();**

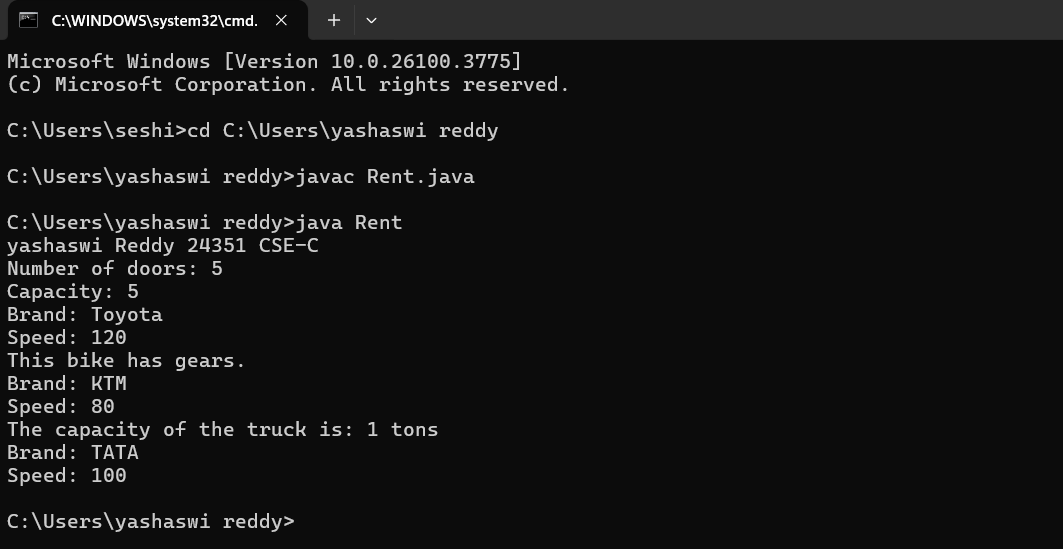
**Trucks t = new Trucks("TATA", 100, 1);**

**t.truckdetails();**

**t.Details();**

**}**

**}**

****

**Class Diagram:**

**+-----------------------------+**

**| Vehicle |**

**+-----------------------------+**

**| - brand: String |**

**| - speed: int |**

**+-----------------------------+**

**| + Details(): void |**

**+-----------------------------+**

**▲ ▲ ▲**

**| | |**

**+------------------+ +------------------+ +------------------+**

**| Cars | | Bikes | | Trucks |**

**+------------------+ +------------------+ +------------------+**

**| - doors: int | | - gears: boolean | | - tons: int |**

**| - capacity: int | +------------------+ +------------------+**

**| + cardetails() | | + bikedetails() | | + truckdetails() |**

**+------------------+ +------------------+ +------------------+**

**+-----------------------------+**

**| Rent |**

**+-----------------------------+**

**| + main(String[]): void |**

**+-----------------------------+**

**Important points:**

**Multi-inheritance: It is one of the types of the inheritance**

**where subclass 2 inherits subclass1 and superclass.**

**Here Vehicle is the super class or parent class and**

**remaining cars, bikes, trucks are the subclasses or child**

**classes.**

**Week 6**

**1.** **Write a Java program to create a vehicle class with a**

**method displayInfo(). Override this method in the car**

**subclass to provide specific information about a car, model,**

**fuel type, and colour using the constructor**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**public Vehicle() {**

**this.brand = "Unknown";**

**this.speed = 0;**

**}**

**void displayInfo() {**

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

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

**}**

**}**

**class Car extends Vehicle {**

**String model;**

**String company;**

**int mileage;**

**String fuelType;**

**public Car(String brand, int speed, String model, String company, int mileage, String fuelType) {**

**this.brand = brand;**

**this.speed = speed;**

**this.model = model;**

**this.company = company;**

**this.mileage = mileage;**

**this.fuelType = fuelType;**

**}**

**@Override**

**void displayInfo() {**

**super.displayInfo();**

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

**System.out.println("Company: " + company);**

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

**System.out.println("Fuel Type: " + fuelType);**

**}**

**}**

**public class Help {**

**public static void main(String[] args) {**

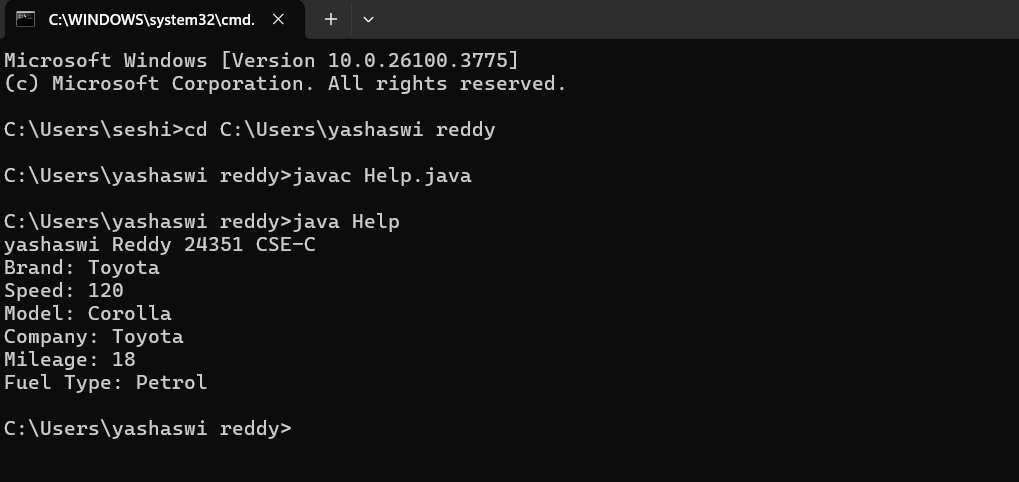
**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**Car car = new Car("Toyota", 120, "Corolla", "Toyota", 18, "Petrol");**

**car.displayInfo();**

**}**

**}**

****

**Class Diagram:**

**+------------------+**

**| Vehicle |**

**+------------------+**

**| - brand: String |**

**| - speed: int |**

**+------------------+**

**| + Vehicle() |**

**| + displayInfo() |**

**+------------------+**

**▲**

**|**

**|**

**+------------------------------+**

**| Car |**

**+------------------------------+**

**| - model: String |**

**| - company: String |**

**| - mileage: int |**

**| - fuelType: String |**

**+------------------------------+**

**| + Car(brand, speed, ...) |**

**| + displayInfo() |**

**+------------------------------+**

**Main Method:**

**+----------------------------+**

**| Help |**

**+----------------------------+**

**| + main(String[] args) |**

**+----------------------------+**

**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

**2.** **Create a Java program for the scenario.**

**A college is developing an automated admission system that**

**verifies student eligibility for undergraduate (UG) and**

**postgraduate(PG) programs. Each program has different**

**eligibility criteria based on the student's percentage in their**

**previous qualification.**

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

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

**Code:**

**class AdmissionSystem {**

**String studentName;**

**double percentage;**

**public AdmissionSystem(String studentName, double percentage) {**

**this.studentName = studentName;**

**this.percentage = percentage;**

**}**

**void checkEligibility() {**

**System.out.println("Checking eligibility for: " + studentName);**

**}**

**}**

**class UGAdmission extends AdmissionSystem {**

**public UGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 60) {**

**System.out.println(studentName + " is eligible for UG admission.");**

**} else {**

**System.out.println(studentName + " is not eligible for UG admission.");**

**}**

**}**

**}**

**class PGAdmission extends AdmissionSystem {**

**public PGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 70) {**

**System.out.println(studentName + " is eligible for PG admission.");**

**} else {**

**System.out.println(studentName + " is not eligible for PG admission.");**

**}**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**UGAdmission ugStudent = new UGAdmission("John", 65);**

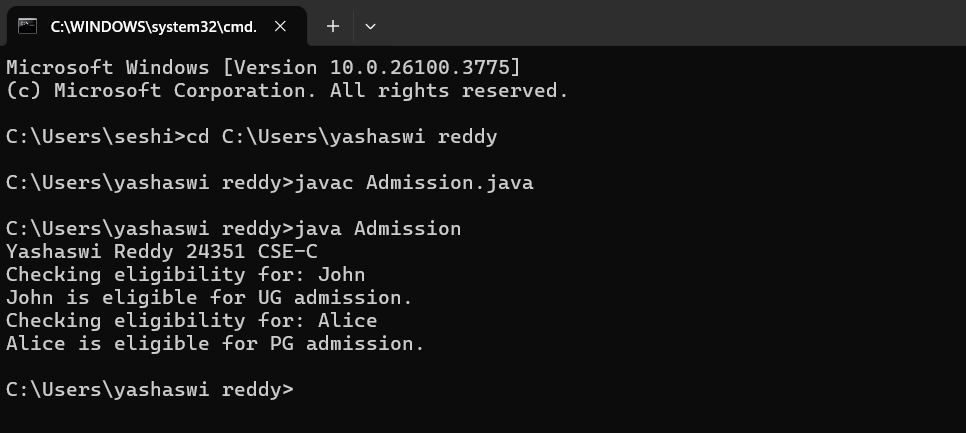
**ugStudent.checkEligibility();**

**PGAdmission pgStudent = new PGAdmission("Alice", 72);**

**pgStudent.checkEligibility();**

**}**

**}**

****

**+-------------------------------+**

**| AdmissionSystem |**

**+-------------------------------+**

**| - studentName: String |**

**| - percentage: double |**

**+-------------------------------+**

**| + AdmissionSystem(name, %) |**

**| + checkEligibility() |**

**+-------------------------------+**

**▲ ▲**

**| |**

**+------------------+ +------------------+**

**| UGAdmission | | PGAdmission |**

**+------------------+ +------------------+**

**| + UGAdmission(...)| | + PGAdmission(...)|**

**| + checkEligibility()| | + checkEligibility()|**

**+------------------+ +------------------+**

**Main Method:**

**+------------------------------+**

**| Main |**

**+------------------------------+**

**| + main(String[] args) |**

**+------------------------------+**

**Important points:**

**Super keyword is used take the method,variable,constructor**

**from the super class.**

**3.** **Write a Java Program to create a Calculator class with**

**overloaded methods to perform addition: Take the integer**

**values a and b from the user.**

**i) Add two integers**

**ii) Add two doubles**

**iii) Add three integer**

**Code:** **class Calculatoroverloading {**

**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 loading {**

**public static void main(String[] args) {**

**Sysstem.out.println(“Yashaswi Reddy 24351 CSE-C”);**

**Calculatoroverloading c = new Calculatoroverloading();**

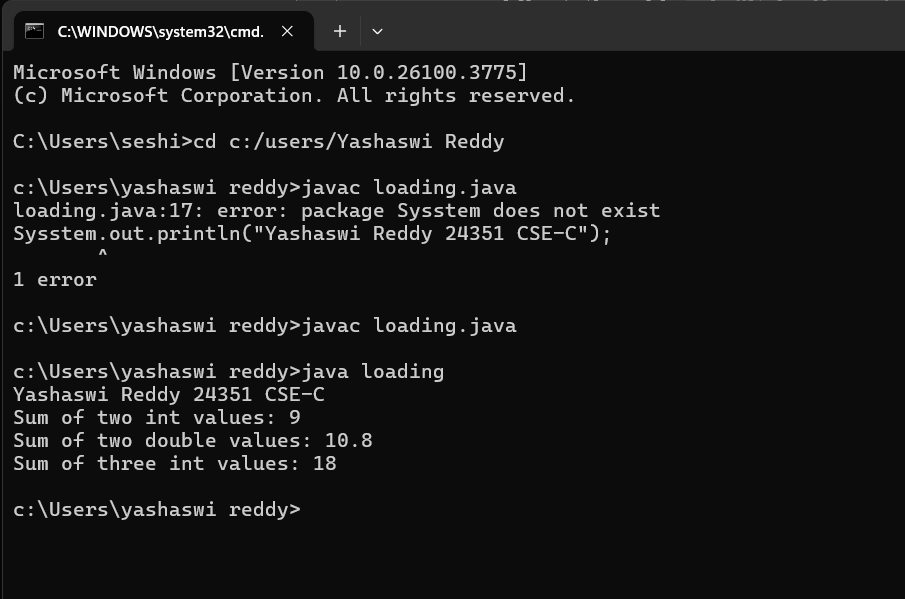
**System.out.println("Sum of two int values: " + c.add(5, 4));**

**System.out.println("Sum of two double values: " + c.add(5.9, 4.9));**

**System.out.println("Sum of three int values: " + c.add(5, 4, 9));**

**}**

**}**

****

**Class Diagram:**

**+------------------------------------+**

**| Calculatoroverloading |**

**+------------------------------------+**

**| + add(int a, int b): int |**

**| + add(double a, double b): double |**

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

**+------------------------------------+**

**▲**

**|**

**+------------------------------+**

**| loading |**

**+------------------------------+**

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

**+------------------------------+**

**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.**

**4.** **Write a Java Program to create a shape class with a**

**method calculateArea() that is overloaded for different**

**shapes(e.g., Square, Rectangle ). Then create a subclass**

**Circle that overrides the**

**calculateArea() method for a circle.**

**Code:**

**class Shape {**

**void calculatearea(int l, int b) {**

**int area = l \* b;**

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

**}**

**void calculatearea(int x) {**

**int area = x \* x;**

**System.out.println("Area of Square is: " + area);**

**}**

**}**

**class Circle {**

**public void calculatearea(double pi, double r) {**

**double area = pi \* (r \* r);**

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

**}**

**}**

**public class Areas2 {**

**public static void main(String[] args) {**

**Shape s = new Shape();**

**Circle c = new Circle();**

**System.out.println(“Yashaswi Reddy 24351 CSE-C”);**

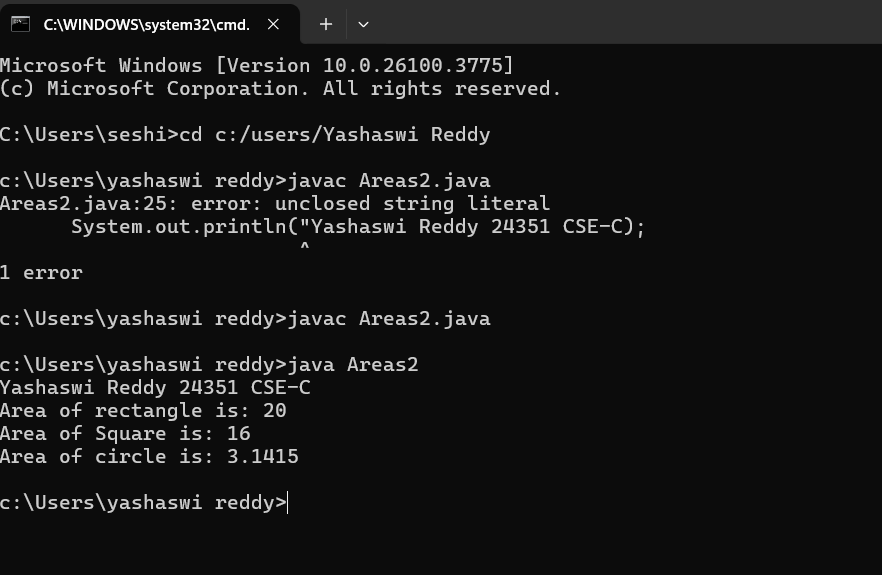
**s.calculatearea(4, 5);**

**s.calculatearea(4);**

**c.calculatearea(3.1415, 1);**

**}**

**}**

****

**Class Diagram:**

**+---------------------------+**

**| Shape |**

**+---------------------------+**

**| + calculatearea(int, int): void | ← Rectangle**

**| + calculatearea(int): void | ← Square**

**+---------------------------+**

**+---------------------------+**

**| Circle |**

**+---------------------------+**

**| + calculatearea(double, double): void | ← Circle**

**+---------------------------+**

**+---------------------------+**

**| Areas2 |**

**+---------------------------+**

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

**+---------------------------+**

**Important points:**

**In this program we use both method overloading and**

**overriding to calculate area of diterent 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:**

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

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

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

**Code:**

**abstract class Animal {**

**abstract void sound();**

**static class Tiger extends Animal {**

**@Override**

**void sound() {**

**System.out.println("Tiger growls");**

**}**

**}**

**static class Lion extends Animal {**

**@Override**

**void sound() {**

**System.out.println("Lion growls");**

**}**

**}**

**public static void main(String[] args) {**

**Tiger tiger = new Tiger();**

**tiger.sound();**

**Lion lion = new Lion();**

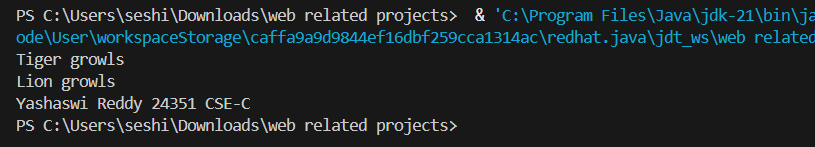
**lion.sound();**

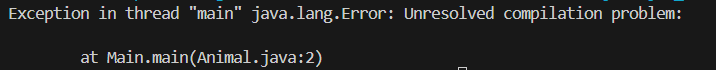
**System.out.println("Yashaswi Reddy 24351 CSE-C");**

**}**

**}**

**Output:**

****

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Error while printing the variables.** 2. **Incorrect declaration of integer.** | 1. **Give the plus sign while printing.** 2. **Give input.nextInt(), where I should be capital.** |

**IMPORTANT POINTS:**

1. **We override the methods in the superclass.**
2. **Here we are using the heirarchial inheritance.**

**PROGRAM-2**

**AIM:**

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

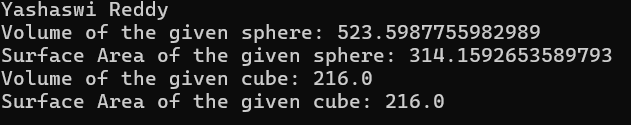
**CLASS DIAGRAM:**

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

|  |
| --- |
| **Sphere**  **-radius: double**  **+calculateVolume()**  **+calculateSurfaceArea()** |

|  |
| --- |
| **Cube**  **-side: double**  **+calculateVolume()**  **+calculateSurfaceArea()** |

**Output:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Wrong datatype entered.** 2. **Object not defined.** | 1. **Enter the correct datatype i.e double instead of int.** 2. **Enter the correct object and if not create new one.** |

**IMPORTANT POINTS:**

1. **Here we used the abstract to declare an abstract class.**
2. **Abstract classes and methods help us to declare the methods without declaring the return type in them.**
3. **To get the values, we declared a constructor for each subclass and initialized values for them.**

**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 concrete method to display the pattern title.**

**. implement two subclasses:**

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

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

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

**Example Output for n = 5: Number Pattern**

**Star Pattern 1**

**\* 1 2**

**\* \* 1 2 3**

**\* \* \* 1 2 3 4**

**\* \* \* \* 1 2 3 4 5**

**\* \* \* \* \***

**CLASS DIAGRAM:**

|  |
| --- |
| **Pattern Printer**  **+printPattern(n): void**  **+displayTitle(title): void** |

|  |  |  |
| --- | --- | --- |
| **Star Pattern**  **+printPattern(n): void** |  | **NumberPattern**  **+printPattern(n):void** |

**CODE:**

**abstract class patternprinter {**

**abstract void printpattern(int n);**

**public void displayTitle(String title) {**

**System.out.println(title);**

**}**

**}**

**class starpattern extends patternprinter {**

**public void printpattern(int n) {**

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

**for (int j = 1; j <= i; j++) {**

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

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

**}}}**

**class NumberPattern extends patternprinter {**

**public void printpattern(int n) {**

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

**for (int j = 1; j <= i; j++) {**

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

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

**} }}**

**public class ptest {**

**public static void main(String[] args) {**

**System.out.println("YashaswiReddy 24351 CSE-C");**

**int n = 5;**

**patternprinter star = new starpattern();**

**star.displayTitle("Star Pattern");**

**star.printpattern(n);**

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

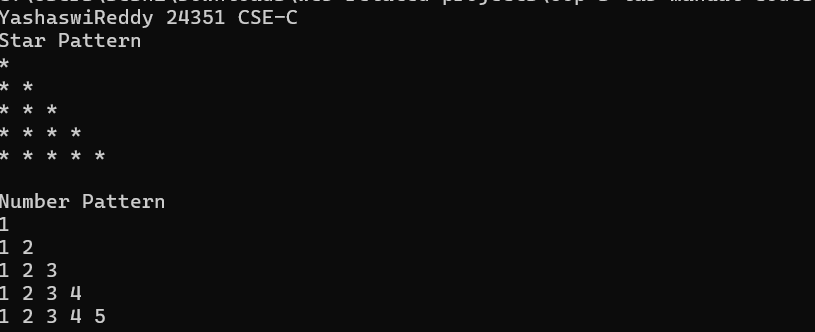
**patternprinter number = new NumberPattern();**

**number.displayTitle("Number Pattern");**

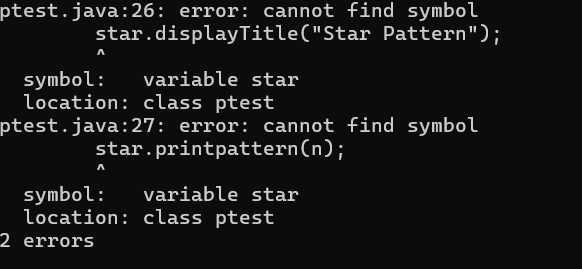
**number.printpattern(n);**

**}}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Not mentioning public to override the abstract methods.** | 1. **Mention the public to override the method.** |

**IMPORTANT POINTS:**

1. **Use abstract classes to enforce a common structure for pattern printing.**
2. **PatternPrinter is the abstract class defining the common template.**
3. **Subclasses (StarPattern, NumberPattern) provide specific implementations.**
4. **displayTitle() is a concrete method shared by all subclasses.**

**WEEK-8**

**PROGRAM – 1**

**AIM**

**Write a java program to create an interface Shape with the getperimeter() method. Create three classes Rectangle, circle, and Triangle that implement the Shape interface. Implement the getperimeter() method for each of the three classes.**

**CLASS DIAGRAM:**

|  |
| --- |
| **Shape**  **+getperimeter(): double** |

|  |
| --- |
| **Rectangle**  **+length**  **+width**  **+getperimeter()** |

|  |
| --- |
| **Triangle**  **+side1**  **+side2**  **+side3** |

|  |
| --- |
| **Circle**  **+radius** |

**CODE:**

**interface Shape {**

**double getperimeter();}**

**class rectangle implements Shape {**

**public int l;**

**public int b;**

**public rectangle(int l, int b) {**

**this.l = l;**

**this.b = b;**

**}**

**public double getperimeter() {**

**return 2\*(l+b);**

**}**

**}**

**class circle implements Shape {**

**public int r;**

**public circle(int r) {**

**this.r = r;**

**}**

**public double getperimeter() {**

**return 2\*3.14\*r;**

**}**

**}**

**class triangle implements Shape {**

**public int a;**

**public int b;**

**public int c;**

**public triangle (int a, int b, int c) {**

**this.a = a;**

**this.b = b;**

**this.c = c;**

**}**

**public double getperimeter() {**

**return a+b+c;**

**}**

**}**

**public class Shapetestt {**

**public static void main(String[]args) {**

**System.out.println("YashaswiReddy 24351 CSE-C");**

**Shape r = new rectangle(2,3);**

**Shape c = new circle(4);**

**Shape t = new triangle(3,4,5);**

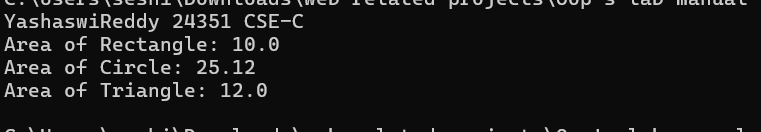
**System.out.println("perimeter of rectangle " + r.getperimeter());**

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

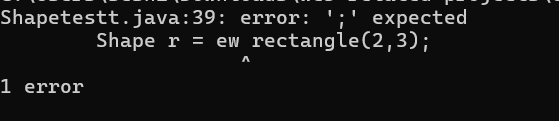
**System.out.println("perimeter of Triangle: " + t.getperimeter());**

**}}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **the constructor rectangle(int, double) is undefined.** | 1. **Give int, int values or declare a constructor with double and int values.** |

**IMPORTANT POINTS:**

1. **For interface classes, first declare the main class and then give the subclass name.**
2. **For methods in interface classes and subclasses, we need to declare public.**

**PROGRAM-2**

**AIM:**

**Write a java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.**

**CLASS DIAGRAM:**

|  |
| --- |
| **Playable**  **+ play(): void** |

|  |
| --- |
| **Football**  **+play()** |

|  |
| --- |
| **Volleyball**  **+play()** |

|  |
| --- |
| **BasketBall**  **+play()** |

**CODE:**

**interface Playable {**

**abstract void play();**

**}**

**class Football implements Playable {**

**public void play() {**

**System.out.println("some people play football in the Ground.");**

**}**

**}**

**class Volleyball implements Playable {**

**public void play() {**

**System.out.println("some people play volleyball in the Stadium.");**

**}**

**}**

**class Basketball implements Playable {**

**public void play() {**

**System.out.println("some people play basket ball in the Rain.");**

**}**

**}**

**public class playtest {**

**public static void main(String[]args) {**

**System.out.println("YashaswiReddy 24351 CSE-C");**

**Playable v = new Volleyball();**

**Playable b = new Basketball();**

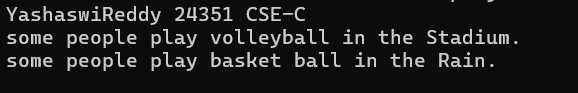
**v.play();**

**b.play();**

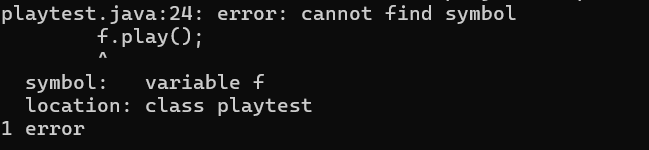
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Declaring an abstract class instead of interface class.** 2. **Not declaring public in each class.** | 1. **Declare an interface class instead of abstract class.** 2. **Declare public infront of each class.** |

**IMPORTANT POINTS:**

1. **The playable interface abstracts the play() method, ensuring different classes implement it differently**
2. **The play() method behaves differently based on the object type football, volleyball, basketball.**
3. **Each class encapsulates its own implementation of how the sport is played, hiding the details from the user.**

**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:**

|  |
| --- |
| **Evenno**  **+message: String**  **+EvenNumberex(String)** |

|  |
| --- |
| **Numberchecker**  **+checkNumber(int): void throws**  **EvenNumberException**  **+main(String[]): void** |

**CODE:**

**class EvenNumberException extends Exception {**

**public EvenNumberException(String message) {**

**super(message);**

**}**

**}**

**public class Numberchecker {**

**public void checkNumber(int number) throws EvenNumberException {**

**if (number % 2 == 0) {**

**throw new EvenNumberException("Even number is: " + number);**

**} else {**

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

**}**

**}**

**public static void main(String[] args) {**

**Numberchecker checker = new Numberchecker();**

**int testNumber = 4;**

**try {**

**checker.checkNumber(testNumber);**

**} catch (EvenNumberException e) {**

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

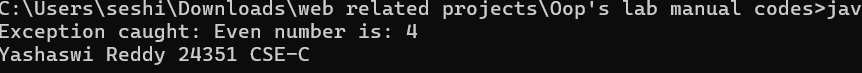
**}**

**System.out.println("Yashaswi Reddy 24351 CSE-C");**

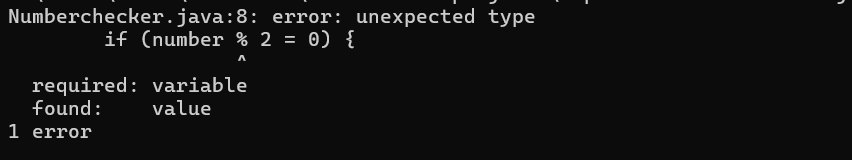
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Unhandled exception type Even number exception.** 2. **Compilation error.** | 1. **If you call checkOddNumber() without using try-catch or without declaring throws.** 2. **If constructor of EvennumberException is missing or incorrectly defined.** |

**IMPORTANT POINTS:**

1. **Here, we 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() mehod.**
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 reads a file and throws an exception if the file is not found.**

**CLASS DIAGRAM:**

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

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

**CODE:**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**import java.util.Scanner;**

**public class filereader {**

**public static void readfile(String filename) throws FileNotFoundException {**

**File file = new File(filename);**

**Scanner input = new Scanner(file);**

**while(input.hasNextLine()) {**

**String line = input.nextLine();**

**System.out.println(line);**

**}**

**input.close();**

**}**

**public static void main(String[] args) {**

**String filename = "ex.txt";**

**try {**

**readfile(filename);**

**} catch (FileNotFoundException e) {**

**System.out.println("File not found: " + filename);**

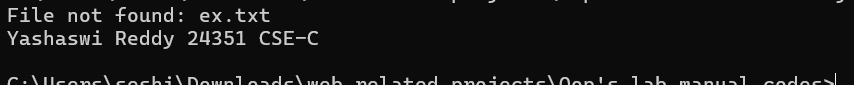
**System.out.println("Yashaswi Reddy 24351 CSE-C");**

**}**

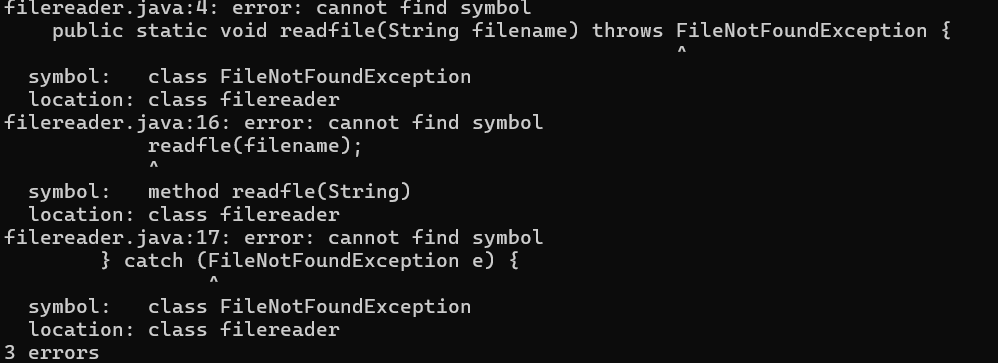
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **File not found exception.** 2. **IOException** | 1. **Occurs if th specified file path is wrong or file does not exist.** 2. **Occurs while reading file if an input/output error happens.** |

**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.**

**CLASS DIAGRAM:**

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

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

**CODE:**

**public class ArithmeticExceptionExample {**

**public static void main(String[] args) {**

**int num = 10;**

**int den = 0;**

**try {**

**int result = num / den;**

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

**} catch (ArithmeticException e) {**

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

**} finally {**

**System.out.println("Finally block executed.");**

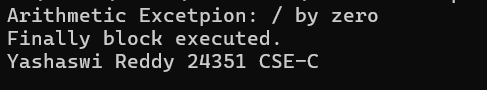
**}**

**System.out.println("Yashaswi Reddy 24351 CSE-C");**

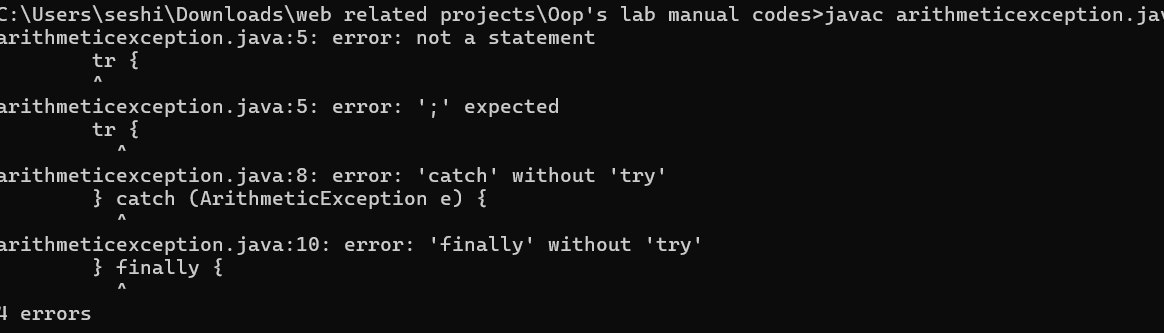
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Method println(String) is undefined for the type printstream.** | 1. **Correct it to pritnln instead of prinlltn.** |

**IMPORTANT POINTS:**

1. **Division by zero(10/0) causes an Arithmetic Exception at runtime.**
2. **The try-catch block handles the excpetion and prevents the program from crashing.**
3. **The finally block always runs, whether an exception occurs or not.**
4. **After handling the error, the program continues running normally and prints additional lines.**

**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.**
2. **Inside it, define two non-static inner classes:**
3. **Department- with variables like deptname and deptcode, and a method to display department details.**
4. **Student: with variables like studentname and rollnumber, and a method to display student details.**
5. **Create on object for each class and call their methods to display their details along with the universityName;**

**CLASS DIAGRAM:**

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

**Inner Classes:**

**- Department**

**- deptName: String**

**- deptCode: String**

**+ displayDepartment(): void**

**- Student**

**- studentName: String**

**- rollNumber: int**

**+ displayStudent(): void**

**CODE:**

**public class UniversityTest {**

**public static void main(String[] args) {**

**University uni = new University("Amrita Vishwa Vidyapeetham");**

**University.Department dept = uni.new Department("CSE", "CSE101");**

**University.Student student = uni.new Student("Y Yashaswi Reddy", "AV.SC.U4CSE24351");**

**dept.displayDepartmentDetails();**

**student.displayStudentDetails();**

**System.out.println("Yashaswi Reddy 24351 CSE-C");**

**}**

**}**

**class University {**

**String universityname;**

**public University(String universityname) {**

**this.universityname = universityname;**

**}**

**class Department {**

**String deptname;**

**String deptcode;**

**public Department(String deptname, String deptcode) {**

**this.deptname = deptname;**

**this.deptcode = deptcode;**

**}**

**public void displayDepartmentDetails() {**

**System.out.println("University: " + universityname);**

**System.out.println("Department Name: " + deptname);**

**System.out.println("Department Code: " + deptcode);**

**}**

**}**

**class Student {**

**String studentname;**

**String rollNo;**

**public Student(String name, String roll) {**

**this.studentname = name;**

**this.rollNo = roll;**

**}**

**public void displayStudentDetails() {**

**System.out.println("University: " + universityname);**

**System.out.println("Student Name: " + studentname);**

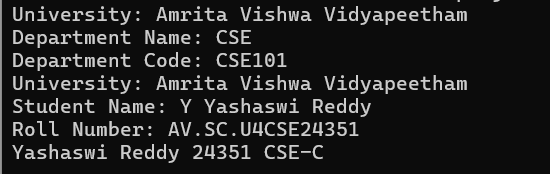
**System.out.println("Roll Number: " + rollNo);**

**}**

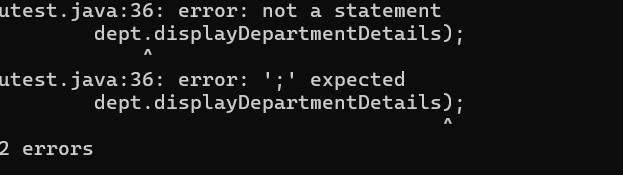
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Syntax error** 2. **Compilation error** | 1. **Wrong object creation for inner class.** 2. **Accessing outer class members wrongly** |

**IMPORTANT POINTS:**

1. **Demonstrates inner class usage.**
2. **Inner classes access outer class members easily.**
3. **Separate objects for Department and Student.**
4. **It is also a good example for encapsulation.**

**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:**

|  |
| --- |
| **Password**  **+first name: String**  **+middlename: String**  **+lastname: String**  **+age: int**  **+password(): String**  **+ main(String[]): void** |

**CODE:**

**import java.util.Scanner;**

**public class password {**

**public static void main(String[] args) {**

**Scanner input = new Scanner(System.in);**

**System.out.print("First Name: ");**

**String fname = input.nextLine();**

**System.out.print("Middle Name: ");**

**String mname = input.nextLine();**

**System.out.print("Last Name: ");**

**String lname = input.nextLine();**

**System.out.print("Age: ");**

**int age = input.nextInt();**

**String password = "" + fname.charAt(0) + mname.charAt(0) + lname.charAt(0) + age;**

**System.out.println("Password: " + password);**

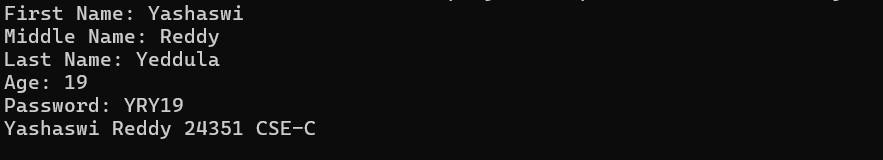
**System.out.println("Yashaswi Reddy 24351 CSE-C");**

**input.close();**

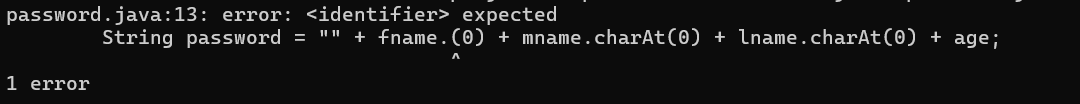
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Identifier expected after the token.** | 1. **Give a suitable identifier that will give the output.** |

**IMPORTANT POINTS:**

1. **The program takes the first, middle, lastnames, and age as input.**
2. **It creates a password using the first letter of each name plus the age.**
3. **The logic is organized using a class and method.**
4. **It uses the Scanner class to read input from the user.**
5. **The program follows object-oriented principles with clean structure and reusability.**

**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:**

|  |
| --- |
| **Strings**  **+string: public**  **+string.uppercase()**  **+string.lowercase()**  **+string.length()**  **+string.index()**  **+void main(String[])args:** |

**CODE:**

**public class strings {**

**public static void main(String[]args) {**

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

**System.out.println(string);**

**System.out.println("Captial letters: " + string.toUpperCase());**

**System.out.println("Lower case: " + string.toLowerCase());**

**System.out.println("Length: " + string.length());**

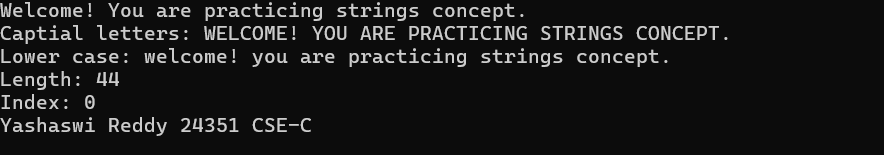
**System.out.println("Index: " + string.indexOf(string));**

**System.out.println("Yashaswi Reddy 24351 CSE-C");**

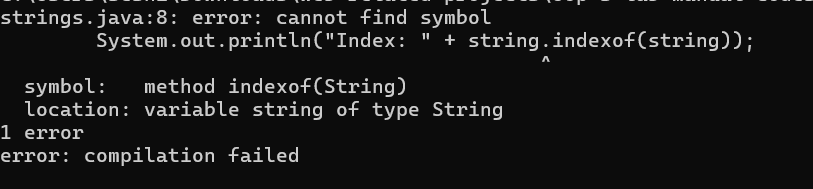
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Toupper is undefined for the type string.** | 1. **Type toUpperCase() instead of to uppercase().** |

**IMPORTANT POINTS:**

1. **The program defines a string message and displays it.**
2. **It uses touppercsae() and toLowerCase() to change the case of the string.**
3. **Length() is used to find the length of the string**
4. **IndexOf() returns 0, because the stirng is searching for itself at index0.**

**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:**

|  |
| --- |
| **Arrays**  **+ sort(arr:T[]): void**  **+tostring(arr: T[]): string**  **+fill(arr: T[], value: T): void**  **+fill(arr: T[], fromindex: int, toindex: int, value: T): void**  **+ copyof(arr: T[], new Length: int): T[]** |

|  |
| --- |
| **Arrays**  **+ sort(arr:T[]): void**  **+tostring(arr: T[]): string**  **+fill(arr: T[], value: T): void**  **+fill(arr: T[], fromindex: int, toindex: int, value: T): void**  **+ copyof(arr: T[], new Length: int): T[]** |

**CODE:**

**import java.util.Arrays;**

**public class arrays {**

**public static void main(String[] args) {**

**Integer[] intArray = {5, 3, 8, 1, 2};**

**System.out.println("Original Integer array: " + Arrays.toString(intArray));**

**Arrays.sort(intArray);**

**System.out.println("Sorted Integer array: " + Arrays.toString(intArray));**

**String[] strArray = {"Apple", "Orange", "Banana", "Grape"};**

**System.out.println("Original String array: " + Arrays.toString(strArray));**

**Arrays.sort(strArray);**

**System.out.println("Sorted String array: " + Arrays.toString(strArray));**

**String intArrayString = Arrays.toString(intArray);**

**System.out.println("Integer array as a string: " + intArrayString);**

**String strArrayString = Arrays.toString(strArray);**

**System.out.println("String array as a string: " + strArrayString);**

**Integer[] fillArray = new Integer[10];**

**Arrays.fill(fillArray, 5);**

**System.out.println("Array filled with 5: " + Arrays.toString(fillArray));**

**Arrays.fill(fillArray, 2, 6, 10);**

**System.out.println("Array filled: " + Arrays.toString(fillArray));**

**Integer[] copiedArray = Arrays.copyOf(intArray, intArray.length);**

**System.out.println("Copied Integer array: " + Arrays.toString(copiedArray));**

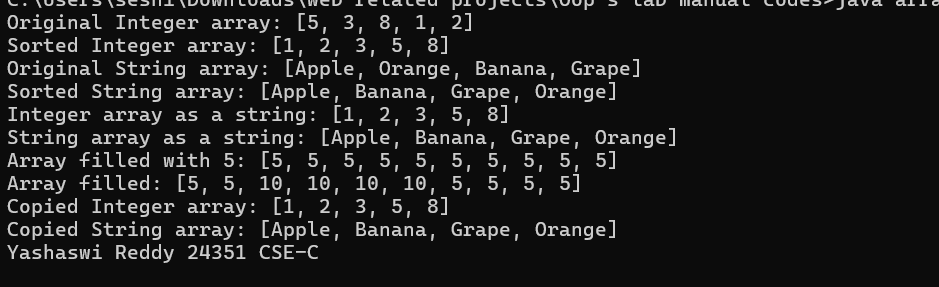
**String[] copiedStrArray = Arrays.copyOf(strArray, strArray.length);**

**System.out.println("Copied String array: " + Arrays.toString(copiedStrArray));**

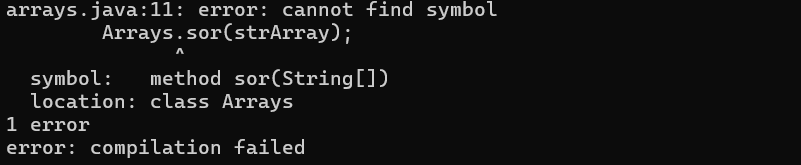
**System.out.println("Yashaswi Reddy 24351 CSE-C");**

**}}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **The method toString() in the object is not applicable for the arguments(integer[]).** | 1. **Declare Array instead of array.** |

**IMPORTANT POINTS:**

1. **The code demonstrates sorting arrays to both integers and strings using Arrays.sort().**
2. **It converts arrays to string representations with Arrays.toString()**
3. **It uses Arrays.fill() to fill entire arrays or parts of arrays with specifiec values.**
4. **The code copies arrays using Arrays.copyOf() to create new arrays with the same elements.**

**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:**

|  |
| --- |
| **ArrayList Example**  **+fruits: ArrayList<string>**  **+main(args: String[]): void** |

|  |
| --- |
| **ArrayList<T>**  **+ add(E e): Boolean**  **+add(int index, E element): void**  **+set(int index, E element): E**  **+ get(int index): E**  **+ remove(int index): E**  **+ clear(): void** |

**CODE:**

**import java.util.ArrayList;**

**public class arraylists {**

**public static void main(String[] args) {**

**ArrayList<String> fruits = new ArrayList<String>();**

**fruits.add("Apple");**

**fruits.add("Banana");**

**fruits.add("Papaya");**

**fruits.add(1, "Orange");**

**System.out.println("After adding: " + fruits);**

**fruits.set(2, "Mango");**

**System.out.println("After changing: " + fruits);**

**String element = fruits.get(1); // Get element at index 1**

**System.out.println("Element 1: " + element);**

**fruits.remove(3);**

**System.out.println("After removing: " + fruits);**

**fruits.clear();**

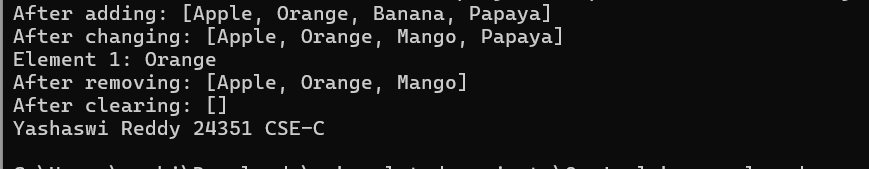
**System.out.println("After clearing: " + fruits);**

**System.out.println("Yashaswi Reddy 24351 CSE-C");**

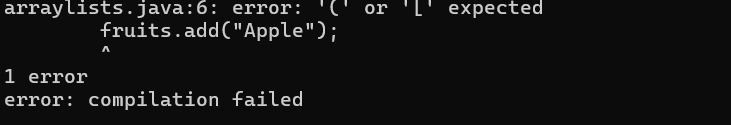
**}**

**}**

**Output:**

****

**Negative case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. **Array list not defined.** 2. **Exception in main.java.lang.error:** | 1. **Define the array list class.** 2. **Insert () to completer classinstancecreationexpression.** |

**IMPORTANT POINTS:**

1. **The program demonstrates how to create and manipulate an arraylist in java**
2. **Elements are inserted at specific positions using the add(index, element) method.**
3. **The set(index, element) method is used to modify existing elements.**
4. **Elements are accessed and removed using the get(index) and remove(index) methods respectively.**
5. **The entire list can be cleared using the clear() method to remove all elements.**