

Object Oriented Programming (23CSE111)

Assignment

|  |  |
| --- | --- |
| Submitted by | |
| Name | G.J.N.S.CHAITANYA |
| Roll No | AV.SC.U4CSE24115 |
| Year/Sem/Section | 1ST Year (2 Sem) CSE-B |
| Date of Submission |  |
| Submitted to | |
| Name | Dr. B Raj Kumar |
| Department | CSE |
| Designation | Asst. Professor |

|  |  |
| --- | --- |
| Marks Evaluated |  |

INDEX

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Experiment Name** | **Page No.** |
|  | **WEEK-1** |  |
| 1 | Installation Process of JDK | 7 |
| 2 | Simple Java Program for printing basic details of student | 9 |
|  | **WEEK-2** |  |
| 1 | Write a Java Program to find the factorial of a number | 12 |
| 2 | Write a Java Program to find the Fibonacci Series of given length | 13 |
| 3 | Write a Java Program to find the temperature from Celsius to Fahrenheit | 15 |
| 4 | Write a Java Program to find the Simple Interest | 11 |
| 5 | Write a Java Program to find the area of triangle using heron’s formula | 18 |
| 6 | Write a Java Program to find the area of rectangle | 17 |
|  | **WEEK-3** |  |
| 1 | Write a java program with the following instructions:  a. Create class with name car.  b. Create 4 attributes named car color, car brand, fuel type, mileage.  c. Create 3 methods named start(), stop(), service().  d. Create 3 objects C1, C2, C3.  e. Create a constructor with parameters with car color, car brand, fuel type, mileage. | 19 |

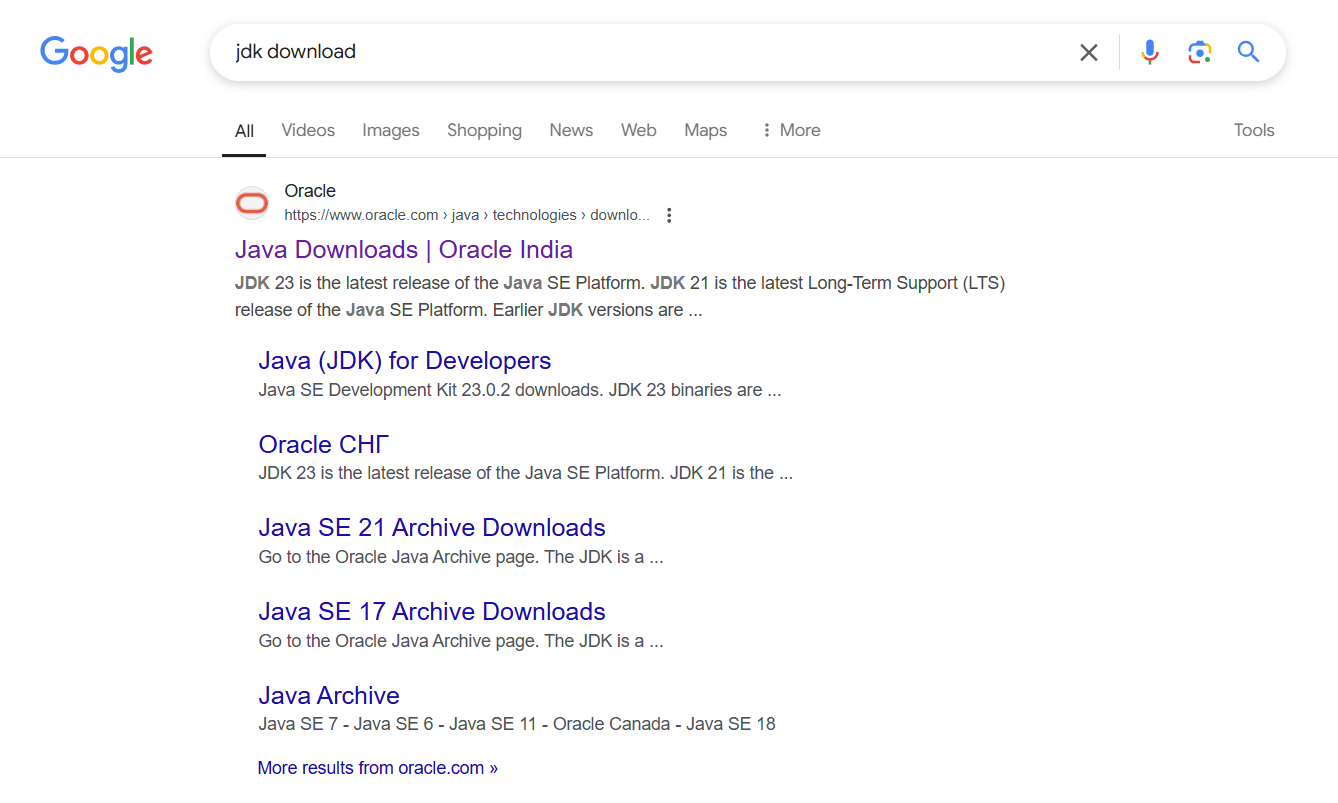
|  |  |  |
| --- | --- | --- |
| 2 | Create a class named bank account with methods deposit and withdraw. Where the deposit method should accepts a parameter and when this method is called the deposited amount should be added to current balance. In addition to that when a withdraw method is called it has to verify whether the withdraw amount is less than the current balance. If not display message saying that there are insufficient funds. Use the constructor to display the details of the customer (Name, Account number, IFSC code, Branch). Also create two customer objects C1, C2. | 23 |
|  | **WEEK-4** |  |
| 1 | Write a Java program with a class named book. The class should contain various attributes such as title, author, year of publication, and price. It should also contain a constructor with parameters that include title, author, year of publication, and price. Create a method that displays details of the book(Display the details of 3 books that is create 3 objects and display their details) | 26 |
| 2 | Create a java program with a Class named "my class" with a static variable 'count' of int type static and initialized to zero and a constant variable 'pi' of type double initialized to 3.1415 as attributes of that class now define a constructor of my class that increments the count variable each time an object of my class is created and finally prints the final values of count and pi variables. | 28 |
|  | **WEEK-5** |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output. | 30 |
| 2 | Vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each vehicle such as brand and speed.  i. Cars should have an additional property: number of doors, Seating capacity.  ii. Bikes should have a property indicating whether they have gears or not.  iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.  iv. Each class should have a constructor.  Questions:  1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.  2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?  a. Truck should include and additional property capacity (in tons).  b. Create a showTruck() method to display the truck’s capacity.  c. Write a constructor for truck that initializes all properties.  3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Finally display the details. | 33 |
|  | **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 | 38 |
| 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% | 40 |
| 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 | 42 |
| 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. | 43 |
|  | **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. | 45 |
| 2 | Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape. | 47 |
| 3 | write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method print pattern (int n) and a concrete method to display the pattern tittle. Implement two sub classes :  1.star pattern -prints a right-angled triangle of stars(\*)  2.number pattern-prints a right angled triangle of increasing numbers. In the main () method,create objects of both subclasses and print the pattern for a given number of rows. | 49 |
|  | **WEEK-8** |  |
| 1 | Write a java program to create an interface shape with getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the shape interface implement the getPerimeter() method for each of the three classes. | 52 |
| 2 | Write a java program to create an interface playable with a method play that takes no arguments and returns play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements the playable interface and override the play method to play the respective sports. | 55 |
| 3 | Write a java program to implement a login system using interfaces | 57 |
|  | **WEEK-9** |  |
| 1 | write a java program to create a method that takes a integer as a parameter and throws an exception if the public class even number | 59 |
| 2 | 2)write a java program to create a method that reads a file and throws an exception if the file is not found  imp points  buffered reader | 61 |
| 3 | write a java program to handle arthematic expression try catch and finally | 63 |
| 4 | write a java program to simulate a university system using inner classes  create an outer class named "university" with a variable universityName  inside it two non static inner classes:  1.department with variable like deptName and deptCode and a method to display department details  2.student with variables like student name and roll number and a method to display  3.create an object for each class and call their methods to display their details along with universityname  ex: public class university{  string universityname="amrita Vishwa Vidyapeetham" | 65 |
|  | **WEEK-10** |  |
| 1 | write a java program to generate a password for a student using his/her initials and age the password displayed should be string consists of first character of first name middle name last name with age | 69 |
| 2 | design and implement a java program that will do the following operations to this string "welcome! you are practicing strings concepts"  \*convert all alphabets to capital letters and print out the result  \*convert all aphabets into lower case letters and print out the resukt  \*print out the length of string  \*print out the index of course | 71 |
| 3 | Implement a java program using the below array methods  \*sorting the elements (numbers &strings) of an array  \*convert the array elements into string  \*fill the part of an array  \*copy the elements of one array into another | 72 |
| 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 | 74 |

WEEK-1

Aim: To download and install JAVA Compiler on our laptops.

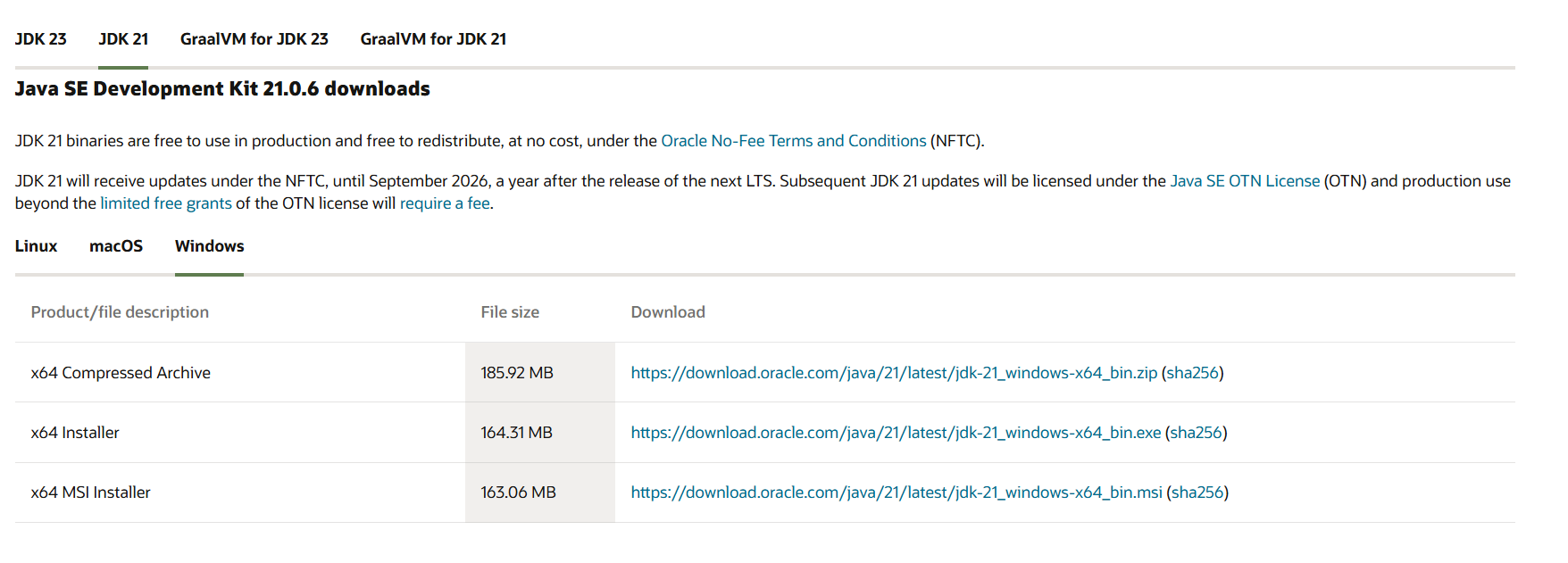
Procedure:

Step1: Search “JDK download” using a search engine



Step2: Click on the ORACEL website to download java

Step3: Select the 21.0.6 version which is suitable and stable for your system



Step4: Choose the appropriate operating system

Step5: Start downloading.

Step6: Once downloaded, navigate to the downloads folder and open the file

Step7: Click Next, accept the terms and conditions, and proceed with the installation

Step8: Java compiler will now be installed successfully.

Step9: After installation, open environment variables by searching for it on your laptop

A screenshot of a computer program

Description automatically generated

Step10: In environment variables, navigate to the system variables and select path option.

Step11: Click Edit, create a new path, and enter the required details.

A screenshot of a computer program

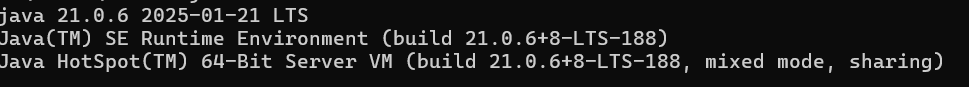
Description automatically generated

Step12: Copy the path, Click OK, return to System variables, and add a new variable named JAVA\_HOME.Paste the copied path and click OK to save.

A screenshot of a computer program

Description automatically generated

Step13: To verify the installation, open the command prompt and check the java version using: java –version.

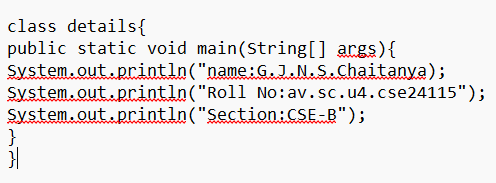


Step14: if the version is displayed, java has been successfully installed

AIM: To print the details of student using Java.

Procedure:

Step1: Open notepad and write the java code.



Step2: Save the file in a designated folder.

Step3: Open the command prompt.

Step4: Navigate to the file location and compile the code using: javac FILEname.java

A screenshot of a computer screen

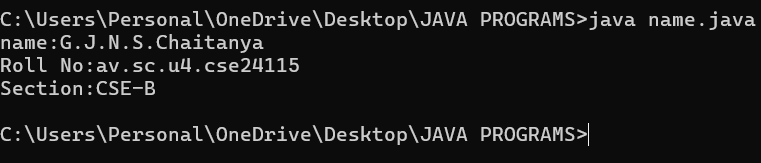
Description automatically generated

Step5: After successful compilation, a.class file will be generated

A screenshot of a computer

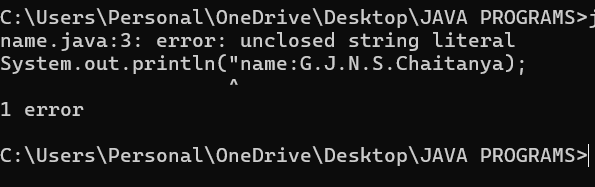
Description automatically generated

Step 6: To see the output type java and file name in command prompt by java FILE name



Step 7: The output will be displayed in the Command Prompt.

MY ERROR:



Ensure that after completion the statement close it with single or double quotes.

WEEK-2

1. AIM:

Simple Java Program for finding simple interest by taking input from

User

Code:



ERROR TABLE:

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

OUTPUT:



2)AIM:

Write a simple program to calculate factorial of a number and read the

input from user

code:



Output:



ERROR TABLE:

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

3) AIM:

Write a program to to calculate the fibonacii sequence and take the input from user

Code:

import java.util.\*;

class fibo

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int num;

int f3;

int f1 = 0;

int f2 = 1;

int i = 2;

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

num = sc.nextInt();

System.out.println(f1);

System.out.println(f2);

while(i<num)

{

f3 = f1+f2;

f1 = f2;

f2 = f3;

System.out.println(f3);

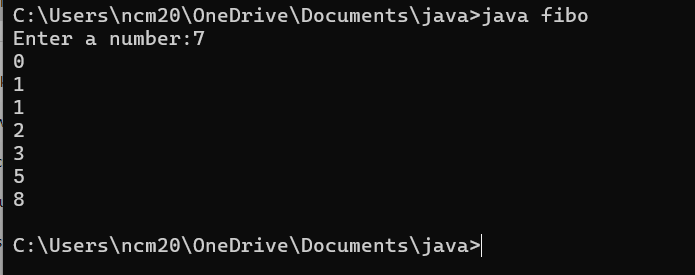
i = i+1;

}

}

}

Output:



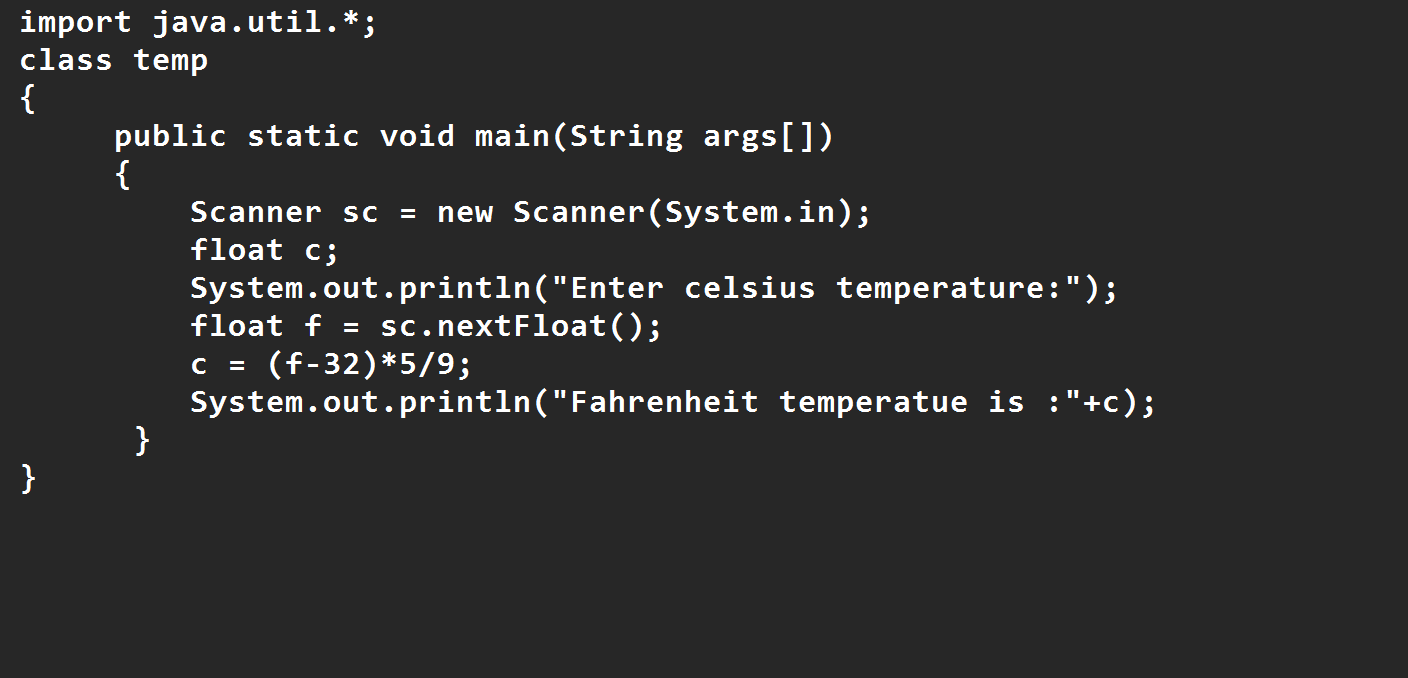
ERROR TABLE:

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

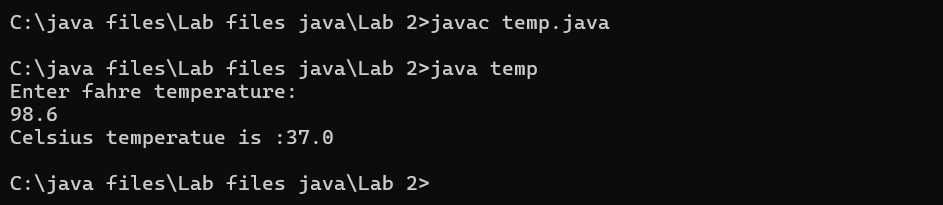
5)AIM

Write a java program to convert temperature from Fahrenheit to celsius

Code:



Output:



Error Table:

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

AIM:

Write a java program to convert temperature from Celsius to Fahrenheit

Code



Output:



Error Table:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Runtime error | Incorrect path selection | Correct path added |
| 2 | Logical error | Incorrect logic | Correct logic |
| 3 |  |  |  |

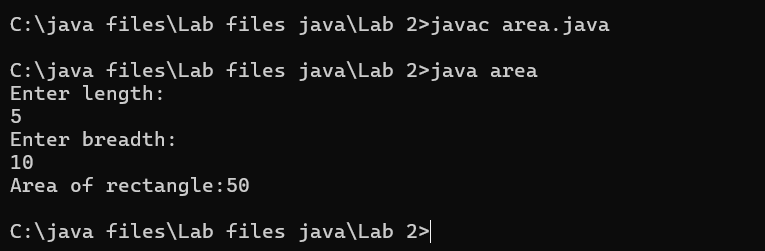
6) AIM:

Write a simple program to find the area of rectangle:

Code:



Output:



Error Table:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Syntax error | Semi colon missing | Semi colon added |
| 2 | Missing import error | Import package missing | Import package added |
| 3 |  |  |  |

7)AIM:

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

Code:



OUTPUT:



Error Table:

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

WEEK 3

Aim:

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 three methods named start(), stop(). Service()

4. Create three objects named car1,car2 and car3

Code:

import java.util.\*;

class car

{

public String Car\_color;

public String Car\_brand;

public String fuel\_type;

public int mileage;

public void start()

{

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

System.out.println("Car color is :"+Car\_color);

System.out.println("Car Brand is:"+Car\_brand);

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

System.out.println("Car mileage is:"+mileage);

}

public void service()

{

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

System.out.println("Car color is :"+Car\_color);

System.out.println("Car Brand is:"+Car\_brand);

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

System.out.println("Car mileage is:"+mileage);

}

public void stop()

{

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

System.out.println("Car color is :"+Car\_color);

System.out.println("Car Brand is:"+Car\_brand);

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

System.out.println("Car mileage is:"+mileage);

}

public static void main(String args[])

{ System.out.println("\n ncm\n\n");

car car1 = new car();

car1.Car\_color = "Blue";

car1.Car\_brand = "BMW";

car1.fuel\_type = "Deisel";

car1.mileage = 10;

car1.start();

car car2 = new car();

car2.Car\_color = "Red";

car2.Car\_brand = "Tesla";

car2.fuel\_type = "EV";

car2.mileage = 300;

car2.stop();

car car3 = new car();

car3.Car\_color = "Yellow";

car3.Car\_brand = "MAHINDRA";

car3.fuel\_type = "Petrol";

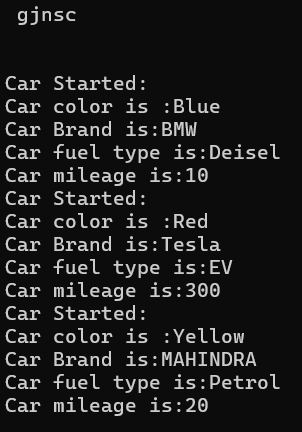
car3.mileage = 20;

car3.service();

}

}

Output:



**Important points:**

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

Class Diagram

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

2.AIM:

To create a class bankAccount with methods deposit() and withdrawl

Code:

class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

if (initialBalance > 0) {

this.balance = initialBalance;

} else {

this.balance = 0;

}

}

public void deposit(double amount) {

if (amount > 0) {

balance = balance + amount;

System.out.println("Deposited $ " + amount);

} else {

System.out.println("Deposited amount must be positive");

}

}

public double getBalance() {

return balance;

}

}

public class Main {

public static void main(String args[]) {

BankAccount account = new BankAccount(1000);

account.deposit(500);

System.out.println("Current Balance is: $" + account.getBalance());

}

}

**Important points:**

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

**CLASS DIAGRAM-**

A close-up of a bank account

AI-generated content may be incorrect.

Output:



Error Table:

|  |  |  |
| --- | --- | --- |
| Sno. | Error message | Error rectification |
| 1. | error: ';' expected  cust1.withdraw(3050) | Add a “;”    cust1.withdraw(3050); |

WEEK-4

1.AIM:

WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOUKD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.

PROGRAM:

public class Book {

public String title;

public String author;

public int year;

Book(String title, String author, int year) {

this.title = title;

this.author = author;

this.year = year;

}

public void displayDetails() {

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

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

System.out.println("Year of Publication" +year);

}

public static void main(String[] args) {

Book b1 = new Book("Math", "Ramanujan", 1950);

Book b2 = new Book("Physics", "CV Raman", 1960);

b1.displayDetails();

b2.displayDetails();

}

}

Output:



NEGATIVE CASE:

A black screen with white text

AI-generated content may be incorrect.

ERROR:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | ERROR TYPE | Reason for error | Rectification |
| 1. | Syntax error | No semicolon | Semicolon added |
| 2. | Runtime error | Incorrect path | Copied correct path |

CLASS DIAGRAM:

|  |
| --- |
| Book |
| -title: String  -author: String  -year: int |
| + Book(title: String, author:String, year: int)  + displayDetails(): void |

**Important points:**

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

2.AIM:

WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH STATIC VARIABLE COUNT OF INT TYPE INTIALIZE IT TO ZERO AND CONSTANT “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS 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 THEFINAL VALUES OF ‘COUNT’ AND ‘PI’ VARIABLES AND CREATE 3 OBJECTS.

PROGRAM:

public class MyClass {

static int count = 0;

static final double pi = 3.14;

MyClass() {

count++;

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

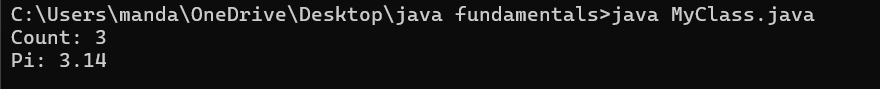
System.out.println("Count: " +count);

System.out.println("Pi: " +pi);

}

}

OUTPUT:



Error Table:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error Type | Reason for error | Rectification |
| 1. | No class | No class name declared | Created class named ‘MyClass’ |
| 2. | Syntax error | Not added keyword | Added keyword named ‘new’ |

CLASS DIAGRAM:

|  |
| --- |
| MyClass |
| -count: int (static)  -pi: double (static, final) |
| +MyClass()  +main(args: String[]):void |

**Important points:**

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

WEEK-5

1) Aim: Create a calculator using the operations including addition, subtraction, multiplication and division using

multilevel in heritance and display the desired output.

- Important Points:

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

Program:

class easy{

void add(int a,int b){

System.out.println("Sum of Numbers is: "+(a+b));

}

void subtract(int a,int b){

System.out.println("Difference of 2 Numbers: "+(a-b));

}

}

class hard extends easy{

void product(int a,int b){

System.out.println("Product of 2 numbrs is: "+(a\*b));

}

}

class ultra extends hard{

void divide(int a,int b){

if (b!=0){

System.out.println("Dividing of 2 numbers is: "+(a/b));

}

else{

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

}

}

}

class Calc{

public static void main(String[] args){

ultra d=new ultra();

d.add(6,9);

d.subtract(9,6);

d.product(23,3);

d.divide(4,2);

}

}

OUTPUT:



Error Table:

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

**Class Diagram :**

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

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

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

2) Aim: 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.

1. Cars should have an additional property: number of doors, Seating capacity.
2. 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?

1. Truck should include and additional property capacity (in tons).
2. Create a showTruck() method to display the truck’s capacity.
3. Write a constructor for truck that initializes all properties.

3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Finally display the details.

- Important Points:

1.Understand the calling of a Constructor

2.Giving class name correctly 3.Give the parameters Correctly

Program:

class Vehicle{

    String brand;

    int speed;

    Vehicle(String brand,int speed){

        this.brand=brand;

        this.speed=speed;

    }

    void Details(){

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

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

    }

}

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==true)

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

        else

        System.out.println("This bike does not have 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 truck is: "+tons);

    }

}

class Rent{

    public static void main(String[] args){

        CARS c=new CARS("Tayota",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);

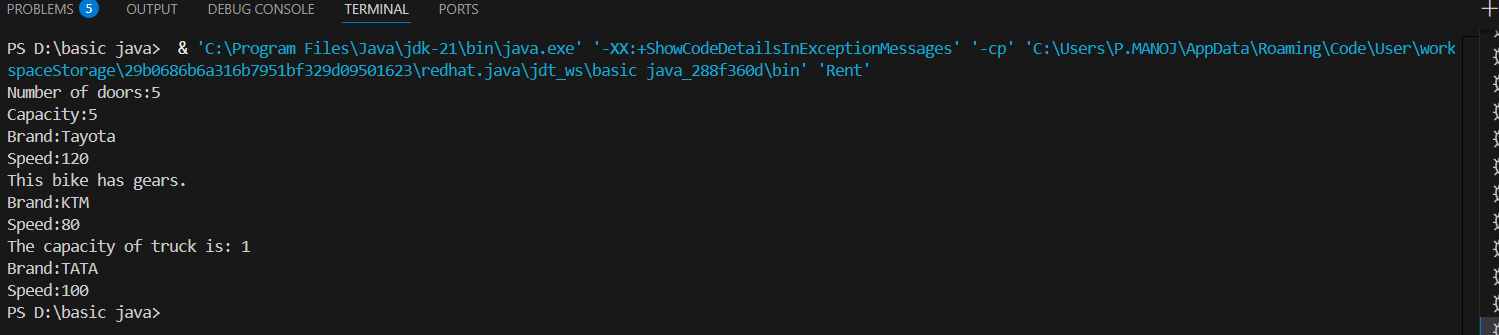
        t.truckdetails();

        t.Details();

    }

}

OUTPUT :



ERROR :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| brand: string  speed: string |
| +Vehicle(String brand, int speed)  +Details(): void |

|  |
| --- |
| **CARS** |
| doors: int  capacity: int |
| + CARS (String brand, int speed, int doors, int capacity)  +cardetails(): void |

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

WEEK – 6

1)Aim : Write a Java program to create a Vehicle class with method displayInfo().Override this method in the Car subclass to provide specific information about a car.

**Important Points:**

Understand the calling of a Constructor

Giving class name correctly

Give the parameters Correctly

Program :

class vehicle{

    String company;

    String model;

    String fuel;

    int capacity;

    void displayInfo(String company,String model,String fuel,int capacity){

        System.out.println("The details of vehicle: ");

        this.company=company;

        this.model=model;

        this.fuel=fuel;

        this.capacity=capacity;

    }

}

class car extends vehicle{

    void displayInfo(String company,String model,String fuel,int capacity){

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

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

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

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

    }

}

class cardetails{

    public static void main(String[] args){

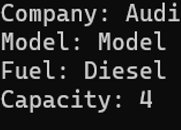
        car car1=new car();

        car1.displayInfo("Audi","Model","Diesel",4);

    }

}

OUTPUT :



ERRORS :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Syntax error | [] is missed | [] is added |
| 2 | Logical error | Incorrect logic | Correct logic |

2) Aim : A college is developing an automated admission system 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 qualifications.

(i)UG admissions require a minimum of 60%.

(ii)PG admissions require a minimum of 70%

**Important Points:**

Understand the calling of a Constructor

Giving class name correctly

Give the parameters Correctly

Program :

class College{

    String name;

    int percentage;

    void geteligibility(String name,int percentage){

        this.name=name;

        this.percentage=percentage;

    }

}

class UG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=60){

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

        }

        else{

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

        }

    }

}

class PG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=70){

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

        }

        else{

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

        }

    }

}

class poly2{

    public static void main(String[] args){

        UG ug=new UG();

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

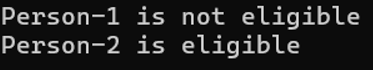
        PG pg=new PG();

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

    }

}

OUPUT :



ERROR :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

3) Aim : Create a Calculator class with overloaded methods to perform addition:

(i) Add two integers.

(ii) Add two doubles.

(iii) Add three integers.

**- Important Points:**

Understand the calling of a Constructor

Giving class name correctly

Give the parameters Correctly

Program :

class Calcee{

  public int add(int a,int b){

        return a+b;

    }

    public double add(double a,double b){

        return a+b;

    }

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

        return a+b+c;

    }

}

class poly3{

    public static void main(String[] args){

        Calcee C1=new Calcee();

        System.out.println("Sum of 2 and 5 is: "+C1.add(2,5));

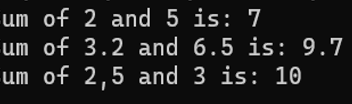
        System.out.println("Sum of 3.2 and 6.5 is: "+C1.add(3.2,6.5));

        System.out.println("Sum of 2,5 and 3 is: "+C1.add(2,5,3));

    }

}

OUTPUT :



ERROR :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

4)Aim : Create a Shape class with a method calculateArea() that is overloaded for different shapes (e.g., square, rectangle). Then, create a subclass Circle that overrides the calculateArea() method for a circle.

**- Important Points:**

Understand the calling of a Constructor

Giving class name correctly

Give the parameters Correctly

Program :

class Shape { // class shape

    void calculateArea( int a) { // method 1

        System.out.println("The area of Square is :" + (a\*a) );

    }

    void calculateArea(int a , int b) { // method 2

        System.out.println("The area of rectangle is :" + (a\*b));

    }

}

class circle extends Shape { // inheritance class

    void calculateArea(double a){ // method overloading

        System.out.println("The area of circle is :" + (3.14\*a\*a));

  } }

class main { // main program

    public static void main(String[] args) {

  Shape s = new Shape();

        circle c = new circle();

      s.calculateArea(4);

        System.out.println("    ");

        s.calculateArea(4, 5);

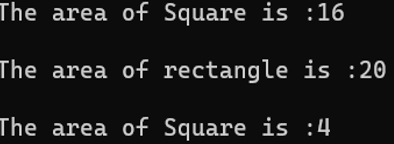
        System.out.println("    ");

        c.calculateArea(2);

    }

}

OUTPUT :



ERRORS :

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Syntax error | ; is missed | ; is added |
| 2 | Logical error | Incorrect logic | Correct logic |

WEEK-7

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

Important points :

1. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.
2. An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

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

Program :

abstract class animal{

abstract void sound();}

class lion extends animal{

void sound(){

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

class tiger extends animal{

void sound(){

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

class p17{

public static void main(String[] args) {

System.out.println("NAME : G.CHAITANYA;ROLL NO : av.sc.u4CSE24115;SEC: CSE-B");

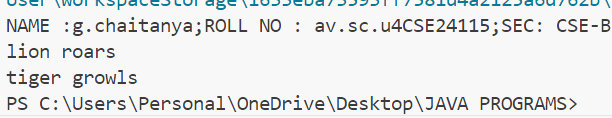
lion l = new lion();

l.sound();

tiger t = new tiger();

t.sound(); } }

Output



Error table

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

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.

Important points :

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

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

3. Understanding the calling of constructor.

Program :

abstract class Shape3D {

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

int radius;

Sphere(int radius) {

this.radius = radius;

}

double calculateVolume() {

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

}

double calculateSurfaceArea() {

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

}

}

class Cube extends Shape3D {

int edge;

Cube(int edge) {

this.edge = edge;

}

double calculateVolume() {

return Math.pow(edge, 3);

}

double calculateSurfaceArea() {

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

}

}

public class Week7{

public static void main(String[] args) {

Shape3D s = new Sphere(5);

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

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

Shape3D c = new Cube(6);

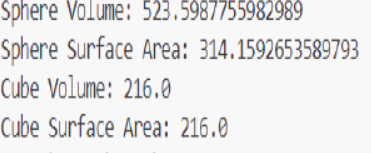
System.out.println("Cube Volume: " + c.calculateVolume());

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

}

}

Output :



Error table

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

* 1. 3) Aim : write a java program using an abstract class to define a method for pattern printing Here we used nested for loop concept the block of code is executed until the condition is false.
  2. Here the logic very important .
  3. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.
  4. Create an abstract class named pattern printer with an abstract method print pattern (int n)

and a concrete method to display the pattern tittle.

Implement two sub class :

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

2.number pattern-prints a right angled triangle of increasing numbers.

In the main () method,create objects of both subclasses and print the pattern for a given number of rows.

Important points :

Program :

abstract class PatternPrinter{

public abstract void printPattern(int n);

public void printTitle(String title) {

System.out.println(title);

}

}

class StarPattern extends PatternPrinter {

@Override

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 {

@Override

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 Week7\_P8 {

public static void main(String[] args) {

int rows=5;

PatternPrinter starPattern = new StarPattern();

PatternPrinter numberPattern = new NumberPattern();

starPattern.printTitle("Star Pattern:");

starPattern.printPattern(rows);

System.out.println();

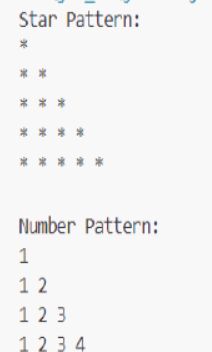
numberPattern.printTitle("Number Pattern:");

numberPattern.printPattern(rows);

}

}

Output :



Error table:

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

Week-8

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

**Important Points:**

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

CODE:

interface Shapee {

float pi = 3.14f;

int getperimeter();

}

class Rectangle implements Shapee {

int length;

int breadth;

public Rectangle(int length, int breadth) {

this.length = length;

this.breadth = breadth;

}

public int getperimeter() {

return 2 \* (length + breadth);

}

}

class Circle implements Shapee {

private int radius;

public Circle(int radius) {

this.radius = radius;

}

public int getperimeter() {

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

}

}

class Triangle implements Shapee {

private int side1;

private int side2;

private int side3;

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

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

public int getperimeter() {

return (side1 + side2 + side3);

}

}

class InShape {

public static void main(String[] args) {

Rectangle r = new Rectangle(5, 6);

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

Circle c = new Circle(7);

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

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

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

}

}

OUTPUT:



Error Table:

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

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

Program:

interface intplayable {

void play();

}

class Football implements intplayable{

public void play(){

System.out.println("Football is playing");

}

}

class Vollyball implements intplayable{

public void play(){

System.out.println("Vollyball is playing");

}

}

class Basketball implements intplayable{

public void play(){

System.out.println("Basketball is playing");

}

}

public class playable{

public static void main(String[] args) {

Football f = new Football();

f.play();

Vollyball v = new Vollyball();

v.play();

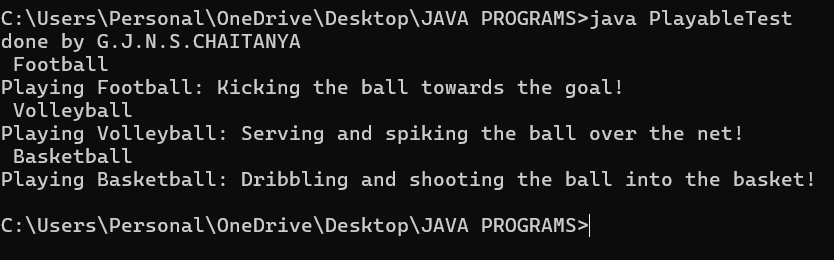
Basketball b = new Basketball();

b.play();

}

}

OUTPUT:



ERROR TABLE:

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

**Important Points:**

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

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

**Program:**

interface loginSystem {

public boolean login(String username, String password);

}

class Universitylogin implements loginSystem {

public boolean login(String username, String password) {

if(username.equals("chaitanya") && password.equals("cse")) {

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

return true;

} else {

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

return false;

}

}

}

class login {

public static void main(String[] args) {

System.out.println("chaitanya");

System.out.println("24115");

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

Universitylogin ul = new Universitylogin();

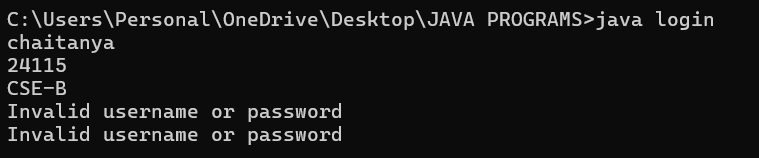
ul.login("chaitanya", "cse24115"); // Invalid

ul.login("chaitanya", "chaitanya01"); // Invalid

}

}

**Output:**

****

**Errors table:**

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

**Important Points:**

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

**WEEK-9**

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

Program:

public class Exceptionnumber {

    public static void checknum(int num) throws Exception {

        if (num%2 == 0){

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

        else{

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

    public static void main(String[] args) {

        Exceptionnumber en = new Exceptionnumber();

        int num = 5;

        try {

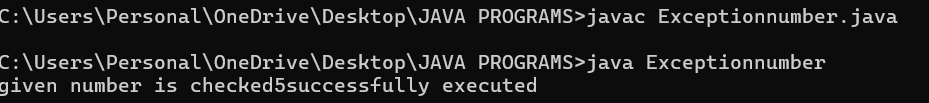
            checknum(5);}

        catch (Exception e) {

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

        }}}

Output:



Class Diagram:

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

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

ERROR TABLE:

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Unhandled exception type EvenNumberException | If you call checkOddNumber() without using try-catch or without declaring throws. |
| 2 | Syntax Error | If missing curly braces {} or wrong method syntax |

**Important points:**

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

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

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

4) Demonstrates user-defined exception handling.

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

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

imp points buffered reader

program:

import java.io.\*;

public class File {

    public static void main(String[] args) {

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

        try {

            BufferedReader reader = new BufferedReader

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

            String line;

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

                System.out.println(line);  }

            reader.close();

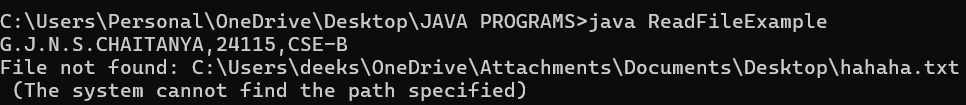
        } catch (FileNotFoundException e) {

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

        } catch (IOException e) {

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

Output:



CLASS DIAGRAM:

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

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

Error table:

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | FileNotFoundException | Occurs if the specified file path is wrong or file does not exist. |
| 2 | IOException | Occurs while reading file if an input/output error happens. |
| 3 | Syntax Error | If missing semicolon ;, wrong try-catch block syntax. |

**Important Points:**

1) Used BufferedReader and FileReader to read text files.

2) FileNotFoundException occurs if the file is missing.

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

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

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

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

Program:

import java.util.Scanner;

class Arthematic {

    public static void main(String[] args) {

        try {

            Scanner sc = new Scanner(System.in);

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

            int a = sc.nextInt();

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

            int b = sc.nextInt();

            int result = a/b;

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

            System.out.println("G.J.N.S.CHAITANYA,24115,CSE-b");

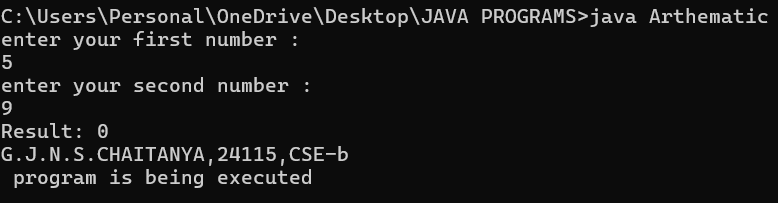
        } catch (Exception e) {

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

        finally{

            System.out.println(" program is being executed"); }}}

Output:



CLASS DIAGRAM:

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

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

Error table:

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | FileNotFoundException | Occurs if the specified file path is wrong or file does not exist. |
| 2 | IOException | Occurs while reading file if an input/output error happens. |
| 3 | Syntax Error | If missing semicolon ;, wrong try-catch block syntax. |

**Important points:**

1) Used BufferedReader and FileReader to read text files.

2) FileNotFoundException occurs if the file is missing.

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

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

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

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

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

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

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

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

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

**ex: public class university{**

**string universityname="amrita Vishwa Vidyapeetham**

**program:**

class University {

String universityName = "Amrita Vishwa Vidyapeetham";

class Department {

String deptName = "CSE";

int deptcode = 101;

}

class Student {

String studentName = "G.J.N.S.CHAITANYA";

int rollNum = 24115;

}

public static void main(String[] args) {

University un = new University();

System.err.println(un.universityName);

Department dp = un.new Department();

System.err.println(dp.deptName);

System.err.println(dp.deptcode);

Student st = un.new Student();

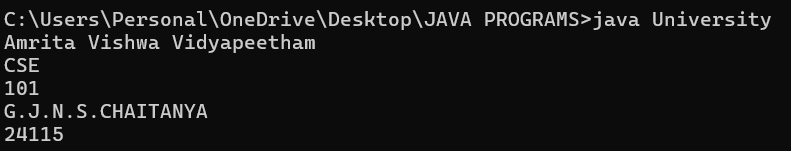
System.err.println(st.studentName);

System.err.println(st.rollNum);

}

}

**Output:**



**CLASS DIAGRAM:**

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

**Inner Classes:**

- Department

- deptName: String

- deptCode: String

+ displayDepartment(): void

- Student

- studentName: String

- rollNumber: int

+ displayStudent(): void

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

**Error table:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax Error | Wrong object creation for inner class |
| 2 | IOException | Occurs while reading file if an input/output error happens. |
| 3 | Syntax Error | If missing semicolon ;, wrong try-catch block syntax. |

**IMPORTANT POINTS:**

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

Good example of encapsulation.

**WEEK-10**

**1)write a java program to generate a password for a student using his/her initials and age the password displayed should be string consists of first character of first name middle name last name with age**

**CODE:**

import java.util.Scanner;

public class PasswordGenerator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt the user for their first, middle, and last names

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

String firstName = scanner.nextLine();

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

String middleName = scanner.nextLine();

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

String lastName = scanner.nextLine();

// Prompt the user for their age

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

int age = scanner.nextInt();

// Generate the password

String password = generatePassword(firstName, middleName, lastName, age);

// Display the generated password

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

}

public static String generatePassword(String firstName, String middleName, String lastName, int age) {

// Extract the first character of each name and convert to uppercase

char firstInitial = Character.toUpperCase(firstName.charAt(0));

char middleInitial = Character.toUpperCase(middleName.charAt(0));

char lastInitial = Character.toUpperCase(lastName.charAt(0));

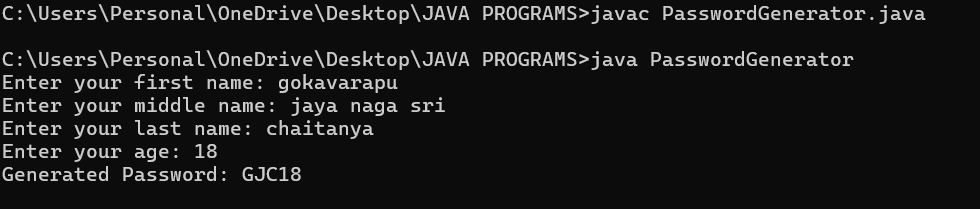
// Concatenate initials and age to form the password

return "" + firstInitial + middleInitial + lastInitial + age;

}

}

OUTPUT:



ERROR TABLE:

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

2)design and implement a java program that will do the following operations to this string "welcome! you are practicing strings concepts"

\*convert all alphabets to capital letters and print out the result

\*convert all aphabets into lower case letters and print out the resukt

\*print out the length of string

\*print out the index of course

CODE:

public class StringOperations {

public static void main(String[] args) {

// Original string

String str = "welcome! you are practicing strings concepts";

// Convert all alphabets to uppercase and print

String upperCaseStr = str.toUpperCase();

System.out.println("Uppercase: " + upperCaseStr);

// Convert all alphabets to lowercase and print

String lowerCaseStr = str.toLowerCase();

System.out.println("Lowercase: " + lowerCaseStr);

// Print the length of the string

int length = str.length();

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

// Find the index of the word "course"

int indexOfCourse = str.indexOf("course");

if (indexOfCourse != -1) {

System.out.println("Index of 'course': " + indexOfCourse);

} else {

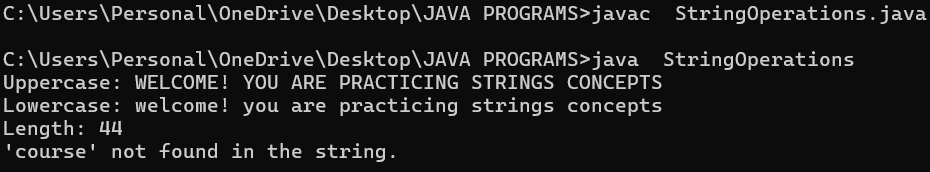
System.out.println("'course' not found in the string.");

}

}

}

OUTPUT:



ERROR TABLE:

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

3)Implement a java program using the below array methods

\*sorting the elements(numbers &strings) of an array

\*convert the array elements into string

\*fill the part of an array

\*copy the elememts of one array into another

CODE:

import java.util.Arrays;

public class ArrayOperations {

public static void main(String[] args) {

// 1. Sorting the elements (numbers and strings)

int[] numbers = { 42, 15, 8, 23, 4 };

String[] words = { "banana", "apple", "cherry", "date" };

Arrays.sort(numbers);

Arrays.sort(words);

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

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

// 2. Converting array elements into a string

String numbersStr = Arrays.toString(numbers);

String wordsStr = Arrays.toString(words);  
System.out.println("Numbers as string: " + numbersStr);

System.out.println("Words as string: " + wordsStr);

// 3. Filling a part of an array

int[] filledArray = new int[10];

Arrays.fill(filledArray, 2, 7, 99); // Fill indices 2 to 6 with 99

System.out.println("Array after filling part: " + Arrays.toString(filledArray));

// 4. Copying elements from one array to another

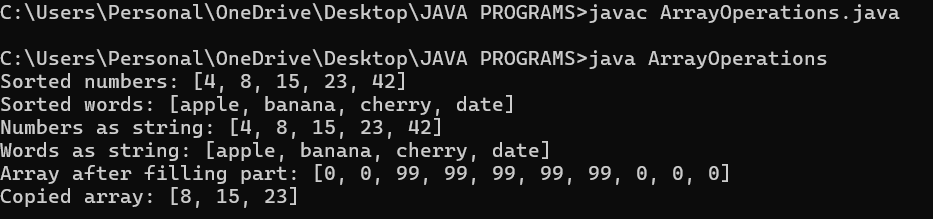
int[] copiedArray = Arrays.copyOfRange(numbers, 1, 4); // Copy elements from index 1 to 3

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

}

}

OUTPUT:



ERROR TABLE:

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

4)Implement a java program using the below arraylist 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

CODE:

import java.util.ArrayList;

public class ArrayListOperations {

public static void main(String[] args) {

// Initialize an ArrayList of strings

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

list.add("Apple");

list.add("Banana");

list.add("Cherry");

// 1. Insert an element at a particular index

list.add(1, "Blueberry"); // Inserts "Blueberry" at index 1

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

// 2. Modify an element

list.set(2, "Blackberry"); // Replaces element at index 2 with "Blackberry"

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

// 3. Access an element

String fruit = list.get(2); // Retrieves the element at index 2

System.out.println("Element at index 2: " + fruit);

// 4. Remove an element

list.remove(1); // Removes the element at index 1

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

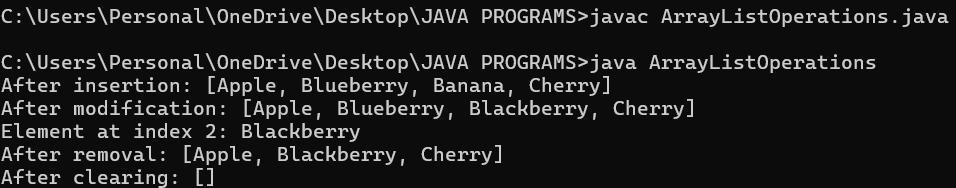
// 5. Clear all elements

list.clear(); // Removes all elements from the list

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

}

}

OUTPUT: 

ERROR TABLE:

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