OOP Problem Statement

1. Create the CaesarCipher class with the following parts:

* Private fields for the alphabet and shiftedAlphabet
* Write a constructor CaesarCipher that has one int parameter key. This method should initialize all the private fields of the class.
* Write an encrypt method that has one String parameter named input. This method returns a String that is the input encrypted using shiftedAlphabet.
* Write a decrypt method that has one String parameter named input. This method returns a String that is the encrypted String decrypted using the key associated with this CaesarCipher object. One way to do this is to create another private field mainKey, which is initialized to be the value of key. Then you can create a CaesarCipher object within decrypt: CaesarCipher cc = new CaesarCipher(26 - mainKey); and call cc.encrypt(input).
* Create the TestCaesarCipher class with the following parts:
* Create a CaesarCipher object with key 18, encrypt the String read in using the Scanner class object, print the encrypted String, and decrypt the encrypted String using the decrypt method.

1. Create the URLFinder class with the following parts:

· Private fields for the url

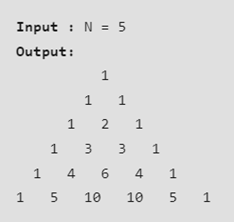
· Write a constructor URLFinder that has one int parameter url. This method should initialize all the private fields of the class.

* Write an urlChecker method that has one String parameter named inputUrl. This method returns a Boolean “true” for valid url.
* Create the TestURLFinder class with the following parts:
* create a URLFinder object with String read in using the Scanner class

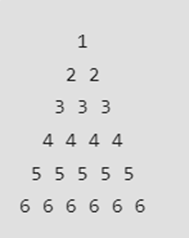
1. Perform the following operations on given matrix

| 1 | 2 | 3 |
| --- | --- | --- |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

M =

1. Transpose
2. Determinant
3. Inverse
4. A) Write a Program to Print the Pascal’s and Number Triangle in Java

4) B) Write a Java program to print the number triangle



5)

1. An ArrayList consists of 1-25 numbers. Write a Java program to remove prime numbers from an ArrayList using an iterator.
2. Write a Java program to

a. Create and traverse (or iterate) an ArrayList using a for-loop, iterator, and advance for-loop.

b. Check if the element(value) exists in the ArrayList?

C. Add element at the particular index of the ArrayList?

6)

* Write a Java program to find the largest and smallest elements in an array of integers.
* Implement a function to reverse an array in place.
* Given two arrays, write a method to merge them into a single sorted array.

7)

* + Write a program to check if a given string is a palindrome.
  + Implement a function to count the occurrences of a specific character in a string.
  + Write a program to remove all whitespace from a string.

8) Design a class BankAccount with methods for deposit, withdraw, and check balance. Implement exception handling for insufficient funds during withdrawal.

9) Write a program to read data from a text file using IO Stream class and display the number of characters and words on the console.

10)

1. Write a program to find the greatest common divisor (GCD) of two numbers.
2. Write a program to convert a decimal number to binary.

11) Create the CarAssembly class which implements Runnable interface with the following parts:

* Private fields for the componentName(String) and timeToPrepare(int)
* Write a constructor CarAssembly that has two parameters componentName and timeToPrepare . This method should initialize all the private fields of the class.
* Write an run method that has sleep method which takes timeToPrepare parameter. The sleep method is invoveked between two print statements componentName is preparing & componentName is ready.
* Components names and their preparation times are as follows
* Engine-3000, Body-4000, Wheels-5000

Create three threads namely engineThread, bodyThread, wheelThread and use Join method for Sysnchronization.

12) Design a Java program to manage an ArrayList of integers that supports dynamic insertion at any position, deletion, updating values, and efficiently computing the sum of elements between two given indices after each modification.

| Sample Input | Sample Output |
| --- | --- |
| insert 0 5  insert 1 10  insert 1 15  update 2 20  sum 0 2  delete 1  sum 0 1 | [5]  [5, 10]  [5, 15, 10]  [5, 15, 20]  25  [5, 20]  25 |

13) Write a Java program using HashMap to add, remove, and track the frequency of words. You also need to find the most frequent word, and if there’s a tie, return the smallest word alphabetically. The program should handle up to 10^5 operations efficiently.

| Sample Input | Sample Output |
| --- | --- |
| add apple  add banana  add apple  remove apple  query apple  mostFrequent | Frequency of 'apple': 1  Most frequent word: banana |

14) Design and implement a multi-threaded banking system in Java that simulates multiple users performing concurrent transactions on shared bank accounts. Each transaction can be a deposit, withdrawal, or transfer between accounts.

15)Design a shopping cart program where users can add items, apply discount codes, and check out. Use custom exceptions to handle scenarios like invalid coupon codes, out-of-stock items, and negative quantity inputs. (Exception handling)

16) Design an interface Vehicle with methods startRide(), endRide(), and calculateFare(int distance). Implement classes like Bike, Auto, and Cab, where each vehicle calculates fares differently. Use a PricingStrategy interface to dynamically adjust fares based on conditions like peak hours and holidays.

17) Design a University Staff Management System using a base class Staff and derived classes Professor, AdministrativeStaff, and MaintenanceStaff.  
 Override methods like displayDetails() and calculateBonus() differently in each subclass using polymorphism.  
 Use a list of base class pointers or references to manage multiple staff objects and demonstrate runtime polymorphism.  
 (Advanced) Implement a promote() method with different behaviors in each subclass.

18)

1. Create a class Rectangle with attributes length and breadth. Write a constructor that uses this keyword to initialize the attributes. Also, create a method compareArea(Rectangle r) that compares the area of two rectangles.
2. Create a class MathOperations with:

* A static method square(int n) that returns the square of a number.
* An instance method cube(int n) that returns the cube of a number. Write a Java program that demonstrates calling both static and instance methods properly.

1. Create a Java program for a seating system in a cinema hall represented by a 2D array (rows × columns).

* Mark all seats as available (e.g., 0) initially.
* Allow the user to book seats by marking them as booked (e.g., 1).
* Display the current seat map (matrix form).
* Add functionality to check if a given seat is available before booking.

20)

Develop a Java program that takes a paragraph input from the user and:

* Remove all vowels (a, e, i, o, u) from the paragraph using StringBuilder.
* Efficiently update the paragraph after each deletion.
* Finally, display the transformed paragraph along with the count of characters removed.

import java.util.Scanner;

public class RemoveVowels {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter paragraph");

String paragraph = sc.nextLine();

StringBuilder sb = new StringBuilder(paragraph);

int count = 0;

for (int i = 0; i < sb.length(); ) {

char ch = sb.charAt(i);

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||

ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {

sb.deleteCharAt(i);

count++;

} else {

i++;

}

}

System.out.println(sb.toString());

System.out.println(count);

}

}

1. Create a base class `Employee` with attributes: name, id, and basicSalary, along with methods `displayDetails()` and `calculateSalary()`. Derive two subclasses: `Manager` with an additional bonus attribute and `Developer` with a projectAllowance attribute. Override the `calculateSalary()` method in both subclasses to include their respective additional amounts. In the main method, create objects of Manager and Developer, and call `displayDetails()` for each. Demonstrate polymorphism by invoking `calculateSalary()` using base class references and display the total salary.

import java.util.Scanner;

class Employee {

String name;

int id;

double basicSalary;

Employee(String name, int id, double basicSalary) {

this.name = name;

this.id = id;

this.basicSalary = basicSalary;

}

void displayDetails() {

System.out.println("Name: " + name);

System.out.println("ID: " + id);

System.out.println("Basic Salary: " + basicSalary);

}

double calculateSalary() {

return basicSalary;

}

}

class Manager extends Employee {

double bonus;

Manager(String name, int id, double basicSalary, double bonus) {

super(name, id, basicSalary);

this.bonus = bonus;

}

@Override

double calculateSalary() {

return basicSalary + bonus;

}

}

class Developer extends Employee {

double projectAllowance;

Developer(String name, int id, double basicSalary, double projectAllowance) {

super(name, id, basicSalary);

this.projectAllowance = projectAllowance;

}

@Override

double calculateSalary() {

return basicSalary + projectAllowance;

}

}

public class EmployeeDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Manager details: Name, ID, Basic Salary, Bonus");

String mName = sc.next();

int mId = sc.nextInt();

double mBasic = sc.nextDouble();

double mBonus = sc.nextDouble();

Employee manager = new Manager(mName, mId, mBasic, mBonus);

System.out.println("Enter Developer details: Name, ID, Basic Salary, Project Allowance");

String dName = sc.next();

int dId = sc.nextInt();

double dBasic = sc.nextDouble();

double dAllowance = sc.nextDouble();

Employee developer = new Developer(dName, dId, dBasic, dAllowance);

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

manager.displayDetails();

System.out.println("Total Salary: " + manager.calculateSalary());

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

developer.displayDetails();

System.out.println("Total Salary: " + developer.calculateSalary());

}

}

22)

Create a class BankAccount with the following attributes and methods:

Attributes:

accountNumber (String)

balance (double)

Methods:

A constructor that initializes the accountNumber and balance.

A method withdraw(double amount) that:

Throws an ArithmeticException if the withdrawal amount is greater than the current balance.

Throws an IllegalArgumentException if the withdrawal amount is less than or equal to zero.

In the main() method:

Create an object of the BankAccount class with an initial balance.

Use try-catch blocks to handle the following scenarios:

Catch the ArithmeticException and display a message "Insufficient funds for withdrawal."

Catch the IllegalArgumentException and display a message "Invalid withdrawal amount."

After handling the exception, allow the program to continue running.

Display the current balance after each operation.

import java.util.Scanner;

class BankAccount {

String accountNumber;

double balance;

BankAccount(String accountNumber, double balance) {

this.accountNumber = accountNumber;

this.balance = balance;

}

void withdraw(double amount) {

if (amount <= 0) {

throw new IllegalArgumentException();

}

if (amount > balance) {

throw new ArithmeticException();

}

balance -= amount;

}

}

public class BankAccountDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter account number:");

String accNo = sc.next();

System.out.println("Enter initial balance:");

double initBalance = sc.nextDouble();

BankAccount account = new BankAccount(accNo, initBalance);

while (true) {

System.out.println("Enter withdrawal amount (or -1 to exit):");

double amt = sc.nextDouble();

if (amt == -1) break;

try {

account.withdraw(amt);

System.out.println("Withdrawal successful.");

} catch (ArithmeticException e) {

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

} catch (IllegalArgumentException e) {

System.out.println("Invalid withdrawal amount.");

}

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

}

}

}

23)

Create a class Printer with a method printNumbers() that prints the numbers from 1 to 10 with a small delay between each number (use Thread.sleep(500) to simulate the delay).

Create two threads:

One thread will call the printNumbers() method and print numbers from 1 to 10.

The second thread will call the printNumbers() method and also print numbers from 1 to 10.

In the main() method:

Create two Printer objects.

Create two threads and start them to execute the printNumbers() method concurrently.

Ensure that the numbers from both threads are printed without any interruption.

* Additional attribute: department (String)

24) Create a Java program to demonstrate access modifiers (private, public, protected, and default).

1. Class Person:  
   * Attributes:  
     + name (private), age (public), address (protected), phoneNumber (default)
   * Methods:  
     + Constructor to initialize all attributes.
     + displayDetails() (public) to display name, age, and address.
     + updatePhoneNumber() (public) to update phoneNumber.
2. Class Employee (extends Person):  
   * Additional attribute: employeeId (public)
   * Override displayDetails() to include employeeId.
3. In the main() method:  
   * Create an Employee object and demonstrate access to attributes and methods using different access modifiers.

25)

1) Create an application for employee management with the following classes:

a) Create an Employee class with following attributes and behaviors:

i) int empId  
 ii) String empName  
 iii) String email  
 iv) String gender  
 v) float salary  
 vi) void GetEmployeeDetails() -> prints employee details

b) Create one more class EmployeeDB with the following attributes and behaviors:  
 i) ArrayList list;  
 ii) boolean addEmployee(Employee e) -> adds the employee object to the collection  
 iii) boolean deleteEmployee(int empId) -> delete the employee object from the collection with the given empid  
 iv) String showPaySlip(int empId) -> returns the payslip of the employee with the given empId

26)

You are given a sorted integer array nums in non-decreasing order. Your task is to remove the duplicate elements in-place such that each element appears only once, and return the new length of the modified array.

You must not allocate extra space for another array; you must do this by modifying the input array in-place with O(1) extra memory.

After removing the duplicates, the first part of the array should contain the unique elements, and the remaining elements can be left as any value (underscores \_ or any arbitrary values).

### Example:

Input: nums = [0,0,1,1,1,2,2,3,3,4]

Output: 5, nums = [0,1,2,3,4,\_,\_,\_,\_,\_]

import java.util.Scanner;

public class RemoveDuplicatesInPlace {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of elements:");

int n = sc.nextInt();

int[] nums = new int[n];

System.out.println("Enter array elements in sorted order:");

for (int i = 0; i < n; i++) {

nums[i] = sc.nextInt();

}

int len = removeDuplicates(nums, n);

System.out.println("New length: " + len);

System.out.print("Modified array: ");

for (int i = 0; i < len; i++) {

System.out.print(nums[i] + " ");

}

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

System.out.print("\_ ");

}

}

static int removeDuplicates(int[] nums, int n) {

if (n == 0) return 0;

int i = 0;

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

if (nums[j] != nums[i]) {

i++;

nums[i] = nums[j];

}

}

return i + 1;

}

}

27)

Write a class MathOperation which accepts 5 integers through the command line.  
Create an array using these parameters.  
Loop through the array and obtain the sum and average of all the elements and display the result.

Handle various exceptions that may arise such as:

* ArithmeticException
* NumberFormatException
* and other relevant exceptions.

import java.util.Scanner;

public class MathOperation {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[] numbers = new int[5];

int sum = 0;

try {

System.out.println("Enter 5 integers:");

for (int i = 0; i < 5; i++) {

String input = sc.next();

numbers[i] = Integer.parseInt(input);

sum += numbers[i];

}

double average = sum / 5.0;

System.out.println("Sum: " + sum);

System.out.println("Average: " + average);

} catch (NumberFormatException e) {

System.out.println("Invalid input: Please enter only integers.");

} catch (ArithmeticException e) {

System.out.println("Arithmetic error occurred.");

} catch (Exception e) {

System.out.println("An unexpected error occurred.");

}

}

}

28)

Create a base class named Fruit with the following attributes:

* name (String)
* taste (String)
* size (String)

Define a method eat() in the Fruit class that prints the name and taste of the fruit.

Now, create two subclasses, Apple and Orange, that inherit from the Fruit class.  
 Override the eat() method in each subclass to display the specific taste of that fruit.

import java.util.Scanner;

class Fruit {

String name;

String taste;

String size;

Fruit(String name, String taste, String size) {

this.name = name;

this.taste = taste;

this.size = size;

}

void eat() {

System.out.println("Fruit: " + name + ", Taste: " + taste);

}

}

class Apple extends Fruit {

Apple(String name, String taste, String size) {

super(name, taste, size);

}

@Override

void eat() {

System.out.println("Fruit: " + name + ", Taste: Sweet and " + taste);

}

}

class Orange extends Fruit {

Orange(String name, String taste, String size) {

super(name, taste, size);

}

@Override

void eat() {

System.out.println("Fruit: " + name + ", Taste: Citrus and " + taste);

}

}

public class FruitDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter Apple details (name, taste, size):");

String aName = sc.next();

String aTaste = sc.next();

String aSize = sc.next();

Fruit apple = new Apple(aName, aTaste, aSize);

System.out.println("Enter Orange details (name, taste, size):");

String oName = sc.next();

String oTaste = sc.next();

String oSize = sc.next();

Fruit orange = new Orange(oName, oTaste, oSize);

apple.eat();

orange.eat();

}

}

29)

Given a number N, the task is to count the number of unique digits in the given number.

Examples:

*Input: N = 22342 Output: 2*

*Explanation: The digits 3 and 4 occurs only once. Hence, the output is 2.*

*Input: N = 99677 Output: 1  
 Explanation: The digit 6 occurs only once. Hence, the output is 1.*

import java.util.Scanner;

public class UniqueDigitCount {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String num = sc.next();

int[] freq = new int[10];

for (int i = 0; i < num.length(); i++) {

char ch = num.charAt(i);

if (Character.isDigit(ch)) {

freq[ch - '0']++;

}

}

int count = 0;

for (int i = 0; i < 10; i++) {

if (freq[i] == 1) {

count++;

}

}

System.out.println("Number of unique digits: " + count);

}

}

30)

Given an array of positive integers nums and a positive integer target, return the minimal length of a subarray whose sum is greater than or equal to target. If there is no such subarray, return 0 instead.

Example 1: Input: target = 7, nums = [2,3,1,2,4,3] Output: 2

Example 2: Input: target = 11, nums = [1,1,1,1,1,1,1,1] Output: 0

import java.util.Scanner;

public class MinSubarrayLength {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter target value:");

int target = sc.nextInt();

System.out.println("Enter number of elements in array:");

int n = sc.nextInt();

int[] nums = new int[n];

System.out.println("Enter array elements:");

for (int i = 0; i < n; i++) {

nums[i] = sc.nextInt();

}

int minLen = Integer.MAX\_VALUE;

int sum = 0, left = 0;

for (int right = 0; right < n; right++) {

sum += nums[right];

while (sum >= target) {

minLen = Math.min(minLen, right - left + 1);

sum -= nums[left];

left++;

}

}

if (minLen == Integer.MAX\_VALUE) {

System.out.println(0);

} else {

System.out.println(minLen);

}

}

}

31)

Write a Java program to receive an integer number as a command-line argument, and print the binary, octal, and hexadecimal equivalents of the given number.

Given Number : 20

Binary equivalent : 10100

Octal equivalent : 24

Hexadecimal equivalent : 14

import java.util.Scanner;

public class NumberConversionManual {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter an integer number:");

int num = sc.nextInt();

System.out.println("Given Number : " + num);

System.out.print("Binary equivalent : ");

printBinary(num);

System.out.print("Octal equivalent : ");

printOctal(num);

System.out.print("Hexadecimal equivalent : ");

printHex(num);

}

static void printBinary(int num) {

if (num == 0) {

System.out.println("0");

return;

}

String res = "";

int n = num;

while (n > 0) {

res = (n % 2) + res;

n /= 2;

}

System.out.println(res);

}

static void printOctal(int num) {

if (num == 0) {

System.out.println("0");

return;

}

String res = "";

int n = num;

while (n > 0) {

res = (n % 8) + res;

n /= 8;

}

System.out.println(res);

}

static void printHex(int num) {

if (num == 0) {

System.out.println("0");

return;

}

String res = "";

int n = num;

char[] hexChars = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

while (n > 0) {

res = hexChars[n % 16] + res;

n /= 16;

}

System.out.println(res);

}

}

32) Develop an online payment system that supports different payment methods.

Requirements:

1. Create an abstract class Payment with:
   * Attributes: amount, transactionID.
   * Abstract method processPayment().
   * Concrete method showTransactionDetails().
2. Create subclasses:
   * CreditCardPayment (cardNumber, CVV, expiryDate).
   * PayPalPayment (email, password).
   * UPIPayment (UPI ID).
3. Implement the processPayment() method in each subclass to handle payments uniquely.
4. Create a PaymentGateway class to process transactions dynamically.

import java.util.Scanner;

abstract class Payment {

double amount;

String transactionID;

Payment(double amount, String transactionID) {

this.amount = amount;

this.transactionID = transactionID;

}

abstract void processPayment();

void showTransactionDetails() {

System.out.println("Transaction ID: " + transactionID);

System.out.println("Amount: " + amount);

}

}

class CreditCardPayment extends Payment {

String cardNumber;

String cvv;

String expiryDate;

CreditCardPayment(double amount, String transactionID, String cardNumber, String cvv, String expiryDate) {

super(amount, transactionID);

this.cardNumber = cardNumber;

this.cvv = cvv;

this.expiryDate = expiryDate;

}

void processPayment() {

System.out.println("Processing Credit Card Payment...");

System.out.println("Card Number: " + cardNumber);

System.out.println("CVV: " + cvv);

System.out.println("Expiry Date: " + expiryDate);

System.out.println("Payment Successful via Credit Card.");

}

}

class PayPalPayment extends Payment {

String email;

String password;

PayPalPayment(double amount, String transactionID, String email, String password) {

super(amount, transactionID);

this.email = email;

this.password = password;

}

void processPayment() {

System.out.println("Processing PayPal Payment...");

System.out.println("Email: " + email);

System.out.println("Payment Successful via PayPal.");

}

}

class UPIPayment extends Payment {

String upiID;

UPIPayment(double amount, String transactionID, String upiID) {

super(amount, transactionID);

this.upiID = upiID;

}

void processPayment() {

System.out.println("Processing UPI Payment...");

System.out.println("UPI ID: " + upiID);

System.out.println("Payment Successful via UPI.");

}

}

class PaymentGateway {

void processTransaction(Payment payment) {

payment.processPayment();

payment.showTransactionDetails();

}

}

public class OnlinePaymentSystem {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Payment payment = null;

PaymentGateway gateway = new PaymentGateway();

System.out.println("Enter amount:");

double amount = sc.nextDouble();

sc.nextLine();

System.out.println("Enter transaction ID:");

String transactionID = sc.nextLine();

System.out.println("Select Payment Method: 1. Credit Card 2. PayPal 3. UPI");

int choice = sc.nextInt();

sc.nextLine();

if (choice == 1) {

System.out.println("Enter Card Number:");

String cardNumber = sc.nextLine();

System.out.println("Enter CVV:");

String cvv = sc.nextLine();

System.out.println("Enter Expiry Date:");

String expiryDate = sc.nextLine();

payment = new CreditCardPayment(amount, transactionID, cardNumber, cvv, expiryDate);

} else if (choice == 2) {

System.out.println("Enter PayPal Email:");

String email = sc.nextLine();

System.out.println("Enter PayPal Password:");

String password = sc.nextLine();

payment = new PayPalPayment(amount, transactionID, email, password);

} else if (choice == 3) {

System.out.println("Enter UPI ID:");

String upiID = sc.nextLine();

payment = new UPIPayment(amount, transactionID, upiID);

} else {

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

return;

}

gateway.processTransaction(payment);

}

}

33) Design a simple system to calculate the area of different 2D shapes using interfaces in Java.

Define an interface named Shape with a method:

Create the following classes that implement the Shape interface:

* Circle with a field radius
* Rectangle with fields length and width
* Triangle with fields base and height

Each class must implement the calculateArea() method according to the respective formula:

* Circle: π × radius²
* Rectangle: length × width
* Triangle: 0.5 × base × height

In the main() method, use polymorphism to create an array of Shape references and call calculateArea() on each.

import java.util.Scanner;

interface Shape {

double calculateArea();

}

class Circle implements Shape {

double radius;

Circle(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Rectangle implements Shape {

double length, width;

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double calculateArea() {

return length \* width;

}

}

class Triangle implements Shape {

double base, height;

Triangle(double base, double height) {

this.base = base;

this.height = height;

}

public double calculateArea() {

return 0.5 \* base \* height;

}

}

public class ShapeAreaDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Shape[] shapes = new Shape[3];

System.out.println("Enter radius of circle:");

double radius = sc.nextDouble();

shapes[0] = new Circle(radius);

System.out.println("Enter length and width of rectangle:");

double length = sc.nextDouble();

double width = sc.nextDouble();

shapes[1] = new Rectangle(length, width);

System.out.println("Enter base and height of triangle:");

double base = sc.nextDouble();

double height = sc.nextDouble();

shapes[2] = new Triangle(base, height);

for (int i = 0; i < shapes.length; i++) {

System.out.println("Area of shape " + (i+1) + ": " + shapes[i].calculateArea());

}

}

}

34). simple banking system that handles user withdrawals, including proper use of exception handling and custom exceptions.

#### Requirements:

1. Create a class BankAccount with the following:  
   * Field: double balance
   * Constructor to initialize balance
   * Method:  
       
      If the withdrawal amount is greater than the balance, throw a custom exception named InsufficientFundsException.
   * Otherwise, deduct the amount from the balance.

2. Define a custom exception class:

* Include a constructor that accepts a custom error message.

In the main() method:

* Create a BankAccount object with an initial balance.
* Try to withdraw different amounts (some valid, some invalid).
* Catch the exception and display appropriate error messages.

import java.util.Scanner;

class InsufficientFundsException extends Exception {

InsufficientFundsException(String message) {

super(message);

}

}

class BankAccount {

double balance;

BankAccount(double balance) {

this.balance = balance;

}

void withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException("Insufficient funds. Withdrawal denied.");

}

balance -= amount;

System.out.println("Withdrawal successful. Remaining balance: " + balance);

}

}

public class SimpleBankingSystem {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter initial balance:");

double initialBalance = sc.nextDouble();

BankAccount account = new BankAccount(initialBalance);

while (true) {

System.out.println("Enter withdrawal amount (or -1 to exit):");

double amount = sc.nextDouble();

if (amount == -1) break;

try {

account.withdraw(amount);

} catch (InsufficientFundsException e) {

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

}

}

}

}

35)

The Citizen class should have following attributes name, id, country, sex, maritalStatus, anualIncome, and economyStatus. Validate the fields if the age is below 18 and country is not ‘India’ throw NonEligibleException and give proper message. Use toString method to display the citizen object in proper format. Use separate packages for Exception and application classes

36)

A. Write a Java program to remove prime numbers between 1 to 25 from ArrayList using an iterator.

import java.util.ArrayList;

import java.util.Iterator;

public class RemovePrimes {

public static void main(String[] args) {

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

for (int i = 1; i <= 25; i++) {

list.add(i);

}

Iterator<Integer> it = list.iterator();

while (it.hasNext()) {

int num = it.next();

if (isPrime(num)) {

it.remove();

}

}

System.out.println("ArrayList after removing primes: " + list);

}

static boolean isPrime(int n) {

if (n < 2) return false;

for (int i = 2; i <= Math.sqrt(n); i++) {

if (n % i == 0) return false;

}

return true;

}

}

B. Write a Java program to

a. create and traverse (or iterate) ArrayList using for-loop, iterator, and advance for-loop.

b. check if element(value) exists in ArrayList?

c. add element at particular index of ArrayList?

import java.util.ArrayList;

import java.util.Iterator;

import java.util.Scanner;

public class ArrayListDemo {

public static void main(String[] args) {

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

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of elements to add:");

int n = sc.nextInt();

sc.nextLine();

System.out.println("Enter elements:");

for (int i = 0; i < n; i++) {

String value = sc.nextLine();

list.add(value);

}

System.out.println("Traverse using for-loop:");

for (int i = 0; i < list.size(); i++) {

System.out.println(list.get(i));

}

System.out.println("Traverse using Iterator:");

Iterator<String> it = list.iterator();

while (it.hasNext()) {

System.out.println(it.next());

}

System.out.println("Traverse using advanced for-loop:");

for (String val : list) {

System.out.println(val);

}

System.out.println("Enter element to search:");

String search = sc.nextLine();

if (list.contains(search)) {

System.out.println("Element exists in ArrayList.");

} else {

System.out.println("Element does not exist in ArrayList.");

}

System.out.println("Enter element to add:");

String newElem = sc.nextLine();

System.out.println("Enter index to add at:");

int idx = sc.nextInt();

if (idx >= 0 && idx <= list.size()) {

list.add(idx, newElem);

System.out.println("Element added. Updated ArrayList:");

for (String val : list) {

System.out.println(val);

}

} else {

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

}

}

}

37).

Write a Java program that handles various types of exceptions while performing different operations. The application should read data from a file specified by the user, handling potential `FileNotFoundException` and `IOException`. It should also allow the user to input values for arithmetic operations and handle division by zero using `ArithmeticException`. Additionally, implement exception handling for `InputMismatchException` when the user provides invalid input, `ArrayIndexOutOfBoundsException` for accessing invalid indices in arrays, and `NullPointerException` when performing operations on `null` values. The program should provide user-friendly error messages and ensure smooth execution even when exceptions occur.

38)

Create an abstract class Person with:

* Fields: name, age
* Constructor to initialize the fields
* Abstract method:
* Create a class Student that inherits from Person:  
  + Additional fields: rollNumber, course
  + Override the displayDetails() method to print all student details
* Create another class Teacher that also extends Person:  
  + Additional fields: employeeId, subject
  + Override the displayDetails() method to print all teacher details
* Demonstrate encapsulation by making all fields private and using getter and setter methods.
* In the main() method:  
  + Create an array of Person references (use polymorphism).
  + Store both Student and Teacher objects.
  + Call the displayDetails() method for each object using a loop.

import java.util.Scanner;

abstract class Person {

private String name;

private int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

// Getters and setters

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public int getAge() { return age; }

public void setAge(int age) { this.age = age; }

public abstract void displayDetails();

}

class Student extends Person {

private String rollNumber;

private String course;

public Student(String name, int age, String rollNumber, String course) {

super(name, age);

this.rollNumber = rollNumber;

this.course = course;

}

// Getters and setters

public String getRollNumber() { return rollNumber; }

public void setRollNumber(String rollNumber) { this.rollNumber = rollNumber; }

public String getCourse() { return course; }

public void setCourse(String course) { this.course = course; }

@Override

public void displayDetails() {

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

System.out.println("Name: " + getName());

System.out.println("Age: " + getAge());

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

System.out.println("Course: " + course);

}

}

class Teacher extends Person {

private String employeeId;

private String subject;

public Teacher(String name, int age, String employeeId, String subject) {

super(name, age);

this.employeeId = employeeId;

this.subject = subject;

}

// Getters and setters

public String getEmployeeId() { return employeeId; }

public void setEmployeeId(String employeeId) { this.employeeId = employeeId; }

public String getSubject() { return subject; }

public void setSubject(String subject) { this.subject = subject; }

@Override

public void displayDetails() {

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

System.out.println("Name: " + getName());

System.out.println("Age: " + getAge());

System.out.println("Employee ID: " + employeeId);

System.out.println("Subject: " + subject);

}

}

public class PersonDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input for Student

System.out.println("Enter Student details:");

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

String sName = sc.nextLine();

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

int sAge = sc.nextInt();

sc.nextLine();

System.out.print("Roll Number: ");

String rollNumber = sc.nextLine();

System.out.print("Course: ");

String course = sc.nextLine();

// Input for Teacher

System.out.println("Enter Teacher details:");

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

String tName = sc.nextLine();

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

int tAge = sc.nextInt();

sc.nextLine();

System.out.print("Employee ID: ");

String empId = sc.nextLine();

System.out.print("Subject: ");

String subject = sc.nextLine();

// Array of Person references

Person[] people = new Person[2];

people[0] = new Student(sName, sAge, rollNumber, course);

people[1] = new Teacher(tName, tAge, empId, subject);

// Display details using polymorphism

System.out.println("\nDisplaying details of all persons:");

for (Person p : people) {

p.displayDetails();

System.out.println();

}

}

}

1. A) Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().

import java.util.Scanner;

class Employee {

private String name;

public Employee(String name) {

this.name = name;

}

public void work() {

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

}

public double getSalary() {

return 30000.0; // Example base salary

}

public String getName() {

return name;

}

}

class HRManager extends Employee {

public HRManager(String name) {

super(name);

}

@Override

public void work() {

System.out.println(getName() + " (HR Manager) is managing HR tasks.");

}

public void addEmployee(String newEmpName) {

System.out.println("HR Manager added a new employee: " + newEmpName);

}

}

public class EmployeeDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Employee

System.out.print("Enter Employee name: ");

String empName = sc.nextLine();

Employee emp = new Employee(empName);

emp.work();

System.out.println(emp.getName() + "'s Salary: " + emp.getSalary());

// HR Manager

System.out.print("\nEnter HR Manager name: ");

String hrName = sc.nextLine();

HRManager hr = new HRManager(hrName);

hr.work();

System.out.println(hr.getName() + "'s Salary: " + hr.getSalary());

// Add new employee via HR

System.out.print("Enter name of new employee to add: ");

String newEmp = sc.nextLine();

hr.addEmployee(newEmp);

}

}

1. Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.

import java.util.Scanner;

class Shape {

public void draw() {

System.out.println("Drawing a shape.");

}

public void erase() {

System.out.println("Erasing a shape.");

}

}

class Circle extends Shape {

@Override

public void draw() {

System.out.println("Drawing a Circle.");

}

@Override

public void erase() {

System.out.println("Erasing a Circle.");

}

}

class Triangle extends Shape {

@Override

public void draw() {

System.out.println("Drawing a Triangle.");

}

@Override

public void erase() {

System.out.println("Erasing a Triangle.");

}

}

class Square extends Shape {

@Override

public void draw() {

System.out.println("Drawing a Square.");

}

@Override

public void erase() {

System.out.println("Erasing a Square.");

}

}

public class ShapeDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Choose a shape to draw and erase (1. Circle 2. Triangle 3. Square):");

int choice = sc.nextInt();

Shape chosenShape;

switch (choice) {

case 1: chosenShape = new Circle(); break;

case 2: chosenShape = new Triangle(); break;

case 3: chosenShape = new Square(); break;

default: System.out.println("Invalid choice."); return;

}

chosenShape.draw();

chosenShape.erase();

}

}

1. Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

import java.util.Scanner;

abstract class Shape {

abstract double calculateArea();

abstract double calculatePerimeter();

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

double calculateArea() {

return Math.PI \* radius \* radius;

}

double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle extends Shape {

private double a, b, c;

public Triangle(double a, double b, double c) {

this.a = a;

this.b = b;

this.c = c;

}

double calculateArea() {

double s = (a + b + c) / 2;

return Math.sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

double calculatePerimeter() {

return a + b + c;

}

}

public class ShapeDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Circle:");

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

double radius = sc.nextDouble();

Circle circle = new Circle(radius);

System.out.println("Area of Circle: " + circle.calculateArea());

System.out.println("Perimeter of Circle: " + circle.calculatePerimeter());

System.out.println("\nTriangle:");

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

double a = sc.nextDouble();

System.out.print("Enter side b: ");

double b = sc.nextDouble();

System.out.print("Enter side c: ");

double c = sc.nextDouble();

Triangle triangle = new Triangle(a, b, c);

System.out.println("Area of Triangle: " + triangle.calculateArea());

System.out.println("Perimeter of Triangle: " + triangle.calculatePerimeter());

}

}

1. A) Write a java program to Move all zeroes to end of array

Input: arr[] = {1, 2, 0, 4, 3, 0, 5, 0};

Output: arr[] = {1, 2, 4, 3, 5, 0, 0, 0};

import java.util.Scanner;

public class MoveZeroesToEnd {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter array size:");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter array elements:");

for(int i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

int index = 0;

for(int i = 0; i < n; i++) {

if(arr[i] != 0) {

arr[index++] = arr[i];

}

}

while(index < n) {

arr[index++] = 0;

}

System.out.print("Array after moving zeroes to end: ");

for(int i = 0; i < n; i++) {

System.out.print(arr[i] + " ");

}

}

}

B)Write a Java program to create an interface Sortable with a method sort() that sorts an array of integers in ascending order. Create two classes BubbleSort and SelectionSort that implement the Sortable interface and provide their own implementations of the sort() method.

import java.util.Scanner;

interface Sortable {

void sort(int[] arr);

}

class BubbleSort implements Sortable {

public void sort(int[] arr) {

int n = arr.length;

for(int i = 0; i < n-1; i++) {

for(int j = 0; j < n-i-1; j++) {

if(arr[j] > arr[j+1]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

System.out.print("Array after Bubble Sort: ");

for(int num : arr) {

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

}

System.out.println();

}

}

class SelectionSort implements Sortable {

public void sort(int[] arr) {

int n = arr.length;

for(int i = 0; i < n-1; i++) {

int minIdx = i;

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

if(arr[j] < arr[minIdx]) {

minIdx = j;

}

}

int temp = arr[minIdx];

arr[minIdx] = arr[i];

arr[i] = temp;

}

System.out.print("Array after Selection Sort: ");

for(int num : arr) {

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

}

System.out.println();

}

}

public class SortDemo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter array size: ");

int n = sc.nextInt();

int[] arr = new int[n];

System.out.println("Enter array elements:");

for(int i = 0; i < n; i++) {

arr[i] = sc.nextInt();

}

System.out.println("Choose sorting method: 1. Bubble Sort 2. Selection Sort");

int choice = sc.nextInt();

Sortable sorter;

if(choice == 1) {

sorter = new BubbleSort();

} else if(choice == 2) {

sorter = new SelectionSort();

} else {

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

return;

}

sorter.sort(arr);

}

}

1. A) Write a Java program to create a class called "Book" with instance variables title, author, and price. Implement a default constructor and two parameterized constructors:

One constructor takes the title and author as parameters.

The other constructor takes title, author, and price as parameters.

Print the values of the variables for each constructor.

import java.util.Scanner;

public class Book {

private String title;

private String author;

private double price;

// Default constructor

public Book() {

this.title = "Unknown";

this.author = "Unknown";

this.price = 0.0;

System.out.println("Default Constructor:");

printDetails();

}

// Constructor with title and author

public Book(String title, String author) {

this.title = title;

this.author = author;

this.price = 0.0;

System.out.println("Constructor with Title and Author:");

printDetails();

}

// Constructor with title, author, and price

public Book(String title, String author, double price) {

this.title = title;

this.author = author;

this.price = price;

System.out.println("Constructor with Title, Author, and Price:");

printDetails();

}

public void printDetails() {

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

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

System.out.println("Price : " + price);

System.out.println();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Default constructor

Book book1 = new Book();

// Constructor with title and author

System.out.println("Enter title for Book 2:");

String title2 = sc.nextLine();

System.out.println("Enter author for Book 2:");

String author2 = sc.nextLine();

Book book2 = new Book(title2, author2);

// Constructor with title, author, and price

System.out.println("Enter title for Book 3:");

String title3 = sc.nextLine();

System.out.println("Enter author for Book 3:");

String author3 = sc.nextLine();

System.out.println("Enter price for Book 3:");

double price3 = sc.nextDouble();

Book book3 = new Book(title3, author3, price3);

}

}

B) Write a Java program to create a class called "TrafficLight" with attributes for color and duration, and methods to change the color and check for red or green.