1. **Write programs to explain at least 2 types of nested and inner classes.**

class CPU {

double price;

// nested class

class Processor{

// members of nested class

double cores;

String manufacturer;

double getCache(){

return 4.3;

}

}

// nested protected class

protected class RAM{

// members of protected nested class

double memory;

String manufacturer;

double getClockSpeed(){

return 5.5;

}

}

}

public class Main {

public static void main(String[] args) {

// create object of Outer class CPU

CPU cpu = new CPU();

// create an object of inner class Processor using outer class

CPU.Processor processor = cpu.new Processor();

// create an object of inner class RAM using outer class CPU

CPU.RAM ram = cpu.new RAM();

System.out.println("Processor Cache = " + processor.getCache());

System.out.println("Ram Clock speed = " + ram.getClockSpeed());

}

}

****

**2) Define a class Student (name, roll\_no, class and marks of 6 subjects). Create an array of 5 Student objects. Calculate the percentage of each student using a method per().**

class Student {

String name;

int roll\_no;

String class\_name;

int[] marks;

Student(String name, int roll\_no, String class\_name, int[] marks) {

this.name = name;

this.roll\_no = roll\_no;

this.class\_name = class\_name;

this.marks = marks;

}

float per() {

int total\_marks = 0;

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

total\_marks += marks[i];

}

return (float) total\_marks / (float) marks.length;

}

}

public class Main {

public static void main(String[] args) {

Student[] students = new Student[5];

students[0] = new Student("Alice", 1, "X", new int[]{80, 70, 90, 85, 75, 95});

students[1] = new Student("Bob", 2, "X", new int[]{70, 65, 75, 80, 90, 85});

students[2] = new Student("Charlie", 3, "X", new int[]{90, 80, 85, 95, 70, 75});

students[3] = new Student("David", 4, "X", new int[]{75, 80, 70, 85, 90, 80});

students[4] = new Student("Eve", 5, "X", new int[]{85, 90, 80, 70, 75, 95});

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

System.out.println(students[i].name + "'s percentage is " + students[i].per());

}

}

}

**3). Define a class Staff with members’ id, name, DOB, joining\_date and salary. . Define class TeachingStaff with subjects[], experience and extends Staff. Using array of objects store details of teaching staff. Find senior staff member from the teaching staff (hint.Use joining date)**

import java.time.LocalDate;

import java.util.Arrays;

class Staff {

int id;

String name;

LocalDate dob;

LocalDate joiningDate;

double salary;

Staff(int id, String name, LocalDate dob, LocalDate joiningDate, double salary) {

this.id = id;

this.name = name;

this.dob = dob;

this.joiningDate = joiningDate;

this.salary = salary;

}

public LocalDate getJoiningDate() {

return joiningDate;

}

}

class TeachingStaff extends Staff {

String[] subjects;

int experience;

TeachingStaff(int id, String name, LocalDate dob, LocalDate joiningDate, double salary, String[] subjects, int experience) {

super(id, name, dob, joiningDate, salary);

this.subjects = subjects;

this.experience = experience;

}

}

public class Main {

public static void main(String[] args) {

// create an array of teaching staff

TeachingStaff[] teachingStaff = {

new TeachingStaff(1, "John", LocalDate.of(1980, 1, 1), LocalDate.of(2010, 1, 1), 50000, new String[]{"Math", "Science"}, 10),

new TeachingStaff(2, "Mary", LocalDate.of(1975, 3, 15), LocalDate.of(2015, 3, 15), 60000, new String[]{"English", "Social Studies"}, 8),

new TeachingStaff(3, "Bob", LocalDate.of(1985, 6, 30), LocalDate.of(2005, 6, 30), 45000, new String[]{"Art", "Music"}, 12)

};

// find the senior staff member by comparing joining dates

TeachingStaff seniorStaff = teachingStaff[0];

for (int i = 1; i < teachingStaff.length; i++) {

if (teachingStaff[i].getJoiningDate().isBefore(seniorStaff.getJoiningDate())) {

seniorStaff = teachingStaff[i];

}

}

// print details of senior staff member

System.out.println("Senior staff member:");

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

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

System.out.println("DOB: " + seniorStaff.dob);

System.out.println("Joining date: " + seniorStaff.joiningDate);

System.out.println("Salary: " + seniorStaff.salary);

System.out.println("Subjects: " + Arrays.toString(seniorStaff.subjects));

System.out.println("Experience: " + seniorStaff.experience);

}

}

**4.) Write a thread program to display Perfect numbers between one to 1000 after every 5 seconds. Eg 6=3+2+1 or 28= 14+7+4+2+1**

import java.util.ArrayList;

public class PerfectNumberThread implements Runnable {

public static void main(String[] args) {

// Start the thread

Thread t = new Thread(new PerfectNumberThread());

t.start();

}

@Override

public void run() {

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

for (int i = 1; i <= 1000; i++) {

if (isPerfect(i)) {

perfectNumbers.add(i);

}

}

// Display perfect numbers every 5 seconds

while (true) {

System.out.println("Perfect Numbers: " + perfectNumbers);

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

private boolean isPerfect(int num) {

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

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

if (num % i == 0) {

divisors.add(i);

}

}

int sum = 0;

for (int divisor : divisors) {

sum += divisor;

}

return sum == num;

}

}

5).Write a program to accept senior citizens name and age from command prompt. If age is below 60, throw “InvalidAgeException” exception

import java.util.Scanner;

class InvalidAgeException extends Exception {

InvalidAgeException(String s) {

super(s);

}

}

public class SeniorCitizen {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String name = scanner.nextLine();

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

int age = scanner.nextInt();

try {

if (age < 60) {

throw new InvalidAgeException("Invalid Age: Age must be 60 or above.");

}

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

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

} catch (InvalidAgeException e) {

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

}

}

}

**6).Create a package named com Define subpackages; a.transact: with class Transaction with static methods credit() and debit() b.loan: with class LoanAccount with method doTransaction() which calls Transaction c.class mehods. Create one LoanAccount object in main to perform operations on it.**

* **Step 1:**

**package com;**

**public class Main {**

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

**LoanAccount account = new LoanAccount();**

**if (args.length > 0) {**

**String operation = args[0];**

**double amount = Double.parseDouble(args[1]);**

**if (operation.equals("credit")) {**

**account.doTransaction(Transaction.credit(amount));**

**} else if (operation.equals("debit")) {**

**account.doTransaction(Transaction.debit(amount));**

**} else {**

**System.out.println("Invalid operation: " + operation);**

**}**

**} else {**

**System.out.println("Usage: java com.Main <operation> <amount>");**

**}**

**}**

**}**

Note that in order for Q.6 program to work, you'll need to create two separate files named "Transaction.java" and "LoanAccount.java" in their respective sub-packages with the appropriate code for each class. For example, the "Transaction" class could be defined as follows:

* **Step2:**

**package com.transact;**

**public class Transaction {**

**public static double credit(double amount) {**

**// perform credit transaction and return new balance**

**return 0;**

**}**

**public static double debit(double amount) {**

**// perform debit transaction and return new balance**

**return 0;**

**}**

**}**

* **Step3:**

**package com.loan;**

**import com.transact.Transaction;**

**public class LoanAccount {**

**private double balance;**

**public void doTransaction(double newBalance) {**

**// update balance and perform any other necessary actions**

**this.balance = newBalance;**

**}**

**}**

**7). Write a java Program using Regular Expression to check password which should have at least one Capital letter, 1 digit, 1 special character and length should be more than 6 and less than or equal to 8**

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class PasswordValidator {

public static void main(String[] args) {

String password = "MyPa55#d";

boolean isValid = validatePassword(password);

System.out.println("Password is valid: " + isValid);

}

public static boolean validatePassword(String password) {

// Regular expression to check password

String regex = "^(?=.\*[A-Z])(?=.\*\\d)(?=.\*[@#$%^&+=])(?=\\S+$).{7,8}$";

// Compile the pattern

Pattern pattern = Pattern.compile(regex);

// Match the password against the pattern

Matcher matcher = pattern.matcher(password);

// Return true if the password matches the pattern, else false

return matcher.matches();

}

}

Explanation of the regular expression:

* ^ - Beginning of string
* (?=.\*[A-Z]) - Positive lookahead to check for at least one uppercase letter
* (?=.\*\d) - Positive lookahead to check for at least one digit
* (?=.\*[@#$%^&+=]) - Positive lookahead to check for at least one special character
* (?=\S+$) - Positive lookahead to check that there are no whitespace characters in the string
* .{7,8} - Match any character between 7 and 8 times
* $ - End of string

The regular expression checks for at least one uppercase letter, one digit, one special character, and a length between 7 and 8 characters. The \\d is used to escape the backslash, since it has a special meaning in regular expressions.

8) write JDBC application to register students for convocation in MCA (Use PreparedStatement, assume suitable table structure).

* SQL file for this program:-

CREATE TABLE Students (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255),

email VARCHAR(255),

phone VARCHAR(255),

year\_of\_passing INT,

convocation\_attending BOOLEAN

);

* Java code for same program:--

import java.sql.\*;

public class ConvocationRegistration {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/your\_database\_name\_here";

private static final String DB\_USERNAME = "your\_database\_username\_here";

private static final String DB\_PASSWORD = "your\_database\_password\_here";

public static void main(String[] args) {

Connection connection = null;

PreparedStatement preparedStatement = null;

try {

// Establish connection to the database

connection = DriverManager.getConnection(DB\_URL, DB\_USERNAME, DB\_PASSWORD);

// Prepare the SQL statement

String sql = "INSERT INTO Students (name, email, phone, year\_of\_passing, convocation\_attending) " +

"VALUES (?, ?, ?, ?, ?)";

preparedStatement = connection.prepareStatement(sql);

// Get student details from user input

String name = "John Doe";

String email = "johndoe@example.com";

String phone = "1234567890";

int yearOfPassing = 2022;

boolean convocationAttending = true;

// Set values for the PreparedStatement

preparedStatement.setString(1, name);

preparedStatement.setString(2, email);

preparedStatement.setString(3, phone);

preparedStatement.setInt(4, yearOfPassing);

preparedStatement.setBoolean(5, convocationAttending);

// Execute the SQL statement

int rowsAffected = preparedStatement.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Student registered successfully!");

} else {

System.out.println("Failed to register student.");

}

} catch (SQLException e) {

e.printStackTrace();

} finally {

// Close the connection and statement resources

try {

if (preparedStatement != null) {

preparedStatement.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

**9) Define Employee class (name, designation, salary). Define a default and parameterized constructor. Override the toString method. Keep a count of objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method). Also display the contents of each object.**

public class Employee {

private String name;

private String designation;

private double salary;

private static int count = 0;

public Employee() {

this.name = "";

this.designation = "";

this.salary = 0.0;

count++;

}

public Employee(String name, String designation, double salary) {

this.name = name;

this.designation = designation;

this.salary = salary;

count++;

System.out.println("Object created. Object count = " + count);

System.out.println(toString());

}

public static int getCount() {

return count;

}

@Override

public String toString() {

return "Employee{name='" + name + "', designation='" + designation + "', salary=" + salary + "}";

}

public static void main(String[] args) {

Employee e1 = new Employee();

Employee e2 = new Employee("John Doe", "Manager", 5000.0);

Employee e3 = new Employee("Jane Smith", "Developer", 3000.0);

System.out.println("Total objects created = " + Employee.getCount());

}

}

**10) Write servlet program to accept student details to store in database and display in tabular format. Assume suitable table structure (Write HTML interface, web.xml & servlet class)**

1. Create a suitable table structure in the database to store student details.
2. Create an HTML interface to accept student details.
3. Create a web.xml file to map the servlet class to a URL.
4. Create a servlet class to handle the request and response.
5. Creating a table structure in the database:

Step 1:

CREATE TABLE student (

id INT NOT NULL AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

phone VARCHAR(15) NOT NULL,

PRIMARY KEY (id)

);

Step 2:

<!DOCTYPE html>

<html>

<head>

<title>Add Student</title>

</head>

<body>

<h1>Add Student</h1>

<form action="addStudent" method="post">

<label>Name:</label>

<input type="text" name="name" required><br><br>

<label>Email:</label>

<input type="email" name="email" required><br><br>

<label>Phone:</label>

<input type="text" name="phone" required><br><br>

<input type="submit" value="Add">

</form>

</body>

</html>

Step 3:

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd">

<servlet>

<servlet-name>AddStudentServlet</servlet-name>

<servlet-class>com.example.AddStudentServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>AddStudentServlet</servlet-name>

<url-pattern>/addStudent</url-pattern>

</servlet-mapping>

</web-app>

Step 4:

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class StudentServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String name = request.getParameter("name");

String age = request.getParameter("age");

String grade = request.getParameter("grade");

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

PrintWriter out = response.getWriter();

try {

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDatabaseName", "username", "password");

String query = "INSERT INTO students (name, age, grade) VALUES (?, ?, ?)";

stmt = conn.prepareStatement(query);

stmt.setString(1, name);

stmt.setString(2, age);

stmt.setString(3, grade);

stmt.executeUpdate();

out.println("<html><body><h2>Student details added successfully!</h2></body></html>");

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

} finally {

try {

if (rs != null)

rs.close();

if (stmt != null)

stmt.close();

if (conn != null)

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

Step 5:

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class DisplayStudentsServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Connection conn = null;

PreparedStatement stmt = null;

ResultSet rs = null;

PrintWriter out = response.getWriter();

try {

Class.forName("com.mysql.jdbc.Driver");

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/yourDatabaseName", "username", "password");

String query = "SELECT \* FROM students";

stmt = conn.prepareStatement(query);

rs = stmt.executeQuery();

out.println("<html><body><table>");

out.println("<tr><th>ID</th><th>Name</th><th>Age</th><th>Grade</th></tr>");

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String age = rs.getString("age");

String grade = rs.getString("grade");

out.println("<tr><td>" + id + "</td><td>" + name + "</td><td>" + age + "</td><td>" + grade + "</td></tr>");

}

out.println("</table></body></html>");

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

} finally {

try {

if (rs != null)

rs.close();

if (stmt != null)

stmt.close();

if (conn != null)

conn.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

**11) Create thread to display prime numbers between 1 to 500 after every 3 second.**

**import java.util.ArrayList;**

**public class PrimeNumberThread extends Thread {**

**@Override**

**public void run() {**

**ArrayList<Integer> primes = new ArrayList<Integer>();**

**for (int i = 1; i <= 500; i++) {**

**if (isPrime(i)) {**

**primes.add(i);**

**}**

**try {**

**Thread.sleep(3000);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

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

**}**

**private boolean isPrime(int num) {**

**if (num <= 1) {**

**return false;**

**}**

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

**if (num % i == 0) {**

**return false;**

**}**

**}**

**return true;**

**}**

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

**PrimeNumberThread thread = new PrimeNumberThread();**

**thread.start();**

**}**

**}**

**12) Create class called sports\_accessories with attributes Accessory\_id, description, quantity, rate, used\_in\_game. Accept details of 10 accessories from user (5 records), store it in array of objects. Display details of all accessories used in game cricket.**

**import java.util.Scanner;**

**public class SportsAccessories {**

**private int accessory\_id;**

**private String description;**

**private int quantity;**

**private double rate;**

**private String used\_in\_game;**

**public SportsAccessories(int accessory\_id, String description, int quantity, double rate, String used\_in\_game) {**

**this.accessory\_id = accessory\_id;**

**this.description = description;**

**this.quantity = quantity;**

**this.rate = rate;**

**this.used\_in\_game = used\_in\_game;**

**}**

**public int getAccessoryId() {**

**return accessory\_id;**

**}**

**public String getDescription() {**

**return description;**

**}**

**public int getQuantity() {**

**return quantity;**

**}**

**public double getRate() {**

**return rate;**

**}**

**public String getUsedInGame() {**

**return used\_in\_game;**

**}**

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

**SportsAccessories[] accessories = new SportsAccessories[10];**

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

**// Accepting details of 10 accessories from user (5 records)**

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

**System.out.println("Enter details of accessory " + (i + 1) + ":");**

**System.out.print("Accessory ID: ");**

**int accessory\_id = scanner.nextInt();**

**scanner.nextLine();**

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

**String description = scanner.nextLine();**

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

**int quantity = scanner.nextInt();**

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

**double rate = scanner.nextDouble();**

**scanner.nextLine();**

**System.out.print("Used in game: ");**

**String used\_in\_game = scanner.nextLine();**

**accessories[i] = new SportsAccessories(accessory\_id, description, quantity, rate, used\_in\_game);**

**}**

**// Display details of all accessories used in game cricket**

**System.out.println("\nDetails of accessories used in game cricket:");**

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

**if (accessories[i].getUsedInGame().equalsIgnoreCase("cricket")) {**

**System.out.println("Accessory " + (i + 1) + ":");**

**System.out.println("Accessory ID: " + accessories[i].getAccessoryId());**

**System.out.println("Description: " + accessories[i].getDescription());**

**System.out.println("Quantity: " + accessories[i].getQuantity());**

**System.out.println("Rate: " + accessories[i].getRate());**

**System.out.println("Used in game: " + accessories[i].getUsedInGame() + "\n");**

**}**

**}**

**}**

**}**

Explanation:

1. We define a class called SportsAccessories with the attributes Accessory\_id, description, quantity, rate, used\_in\_game.
2. We define a constructor that initializes the attributes of the SportsAccessories object.
3. We define getter methods for each of the attributes to access them from outside the class.
4. In the main method, we create an array of SportsAccessories objects to store the details of 10 accessories.
5. We use a Scanner object to accept the details of each accessory from the user and store it in the array.
6. We then iterate through the array and display the details of all accessories used in the game of cricket by checking the value of the used\_in\_game attribute using the getUsedInGame() method.

**13) Write a thread program to display Perfect numbers between one to 1000 after every 5 seconds.**

**public class PerfectNumberThread extends Thread {**

**public void run() {**

**for (int i = 1; i <= 1000; i++) {**

**if (isPerfectNumber(i)) {**

**System.out.print(i + " ");**

**}**

**if (i % 50 == 0) { // display every 50th number**

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

**}**

**try {**

**Thread.sleep(5000); // wait for 5 seconds**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**private boolean isPerfectNumber(int n) {**

**int sum = 0;**

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

**if (n % i == 0) {**

**sum += i;**

**}**

**}**

**return sum == n;**

**}**

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

**PerfectNumberThread thread = new PerfectNumberThread();**

**thread.start();**

**}**

**}**

**14) Write a program to demonstrate Arrays class any 10 methods with examples (hint: Java.util.Arrays).**

import java.util.Arrays;

public class ArraysDemo {

public static void main(String[] args) {

// Example 1: Sorting an array

int[] arr1 = {3, 2, 1, 5, 4};

Arrays.sort(arr1);

System.out.println("Sorted array: " + Arrays.toString(arr1));

// Example 2: Filling an array with a value

int[] arr2 = new int[5];

Arrays.fill(arr2, 42);

System.out.println("Filled array: " + Arrays.toString(arr2));

// Example 3: Comparing two arrays

int[] arr3 = {1, 2, 3};

int[] arr4 = {1, 2, 4};

int cmp = Arrays.compare(arr3, arr4);

System.out.println("Comparison result: " + cmp);

// Example 4: Copying an array

int[] arr5 = {1, 2, 3};

int[] arr6 = Arrays.copyOf(arr5, 5);

System.out.println("Copied array: " + Arrays.toString(arr6));

// Example 5: Finding an element in an array

int[] arr7 = {1, 2, 3, 4, 5};

int idx = Arrays.binarySearch(arr7, 3);

System.out.println("Index of 3: " + idx);

// Example 6: Checking if two arrays are equal

int[] arr8 = {1, 2, 3};

int[] arr9 = {1, 2, 3};

boolean eq = Arrays.equals(arr8, arr9);

System.out.println("Are arr8 and arr9 equal? " + eq);

// Example 7: Creating a string representation of an array

int[] arr10 = {1, 2, 3};

String str = Arrays.toString(arr10);

System.out.println("Array as a string: " + str);

// Example 8: Creating a hash code for an array

int[] arr11 = {1, 2, 3};

int hash = Arrays.hashCode(arr11);

System.out.println("Array hash code: " + hash);

// Example 9: Finding the maximum value in an array

int[] arr12 = {3, 2, 1, 5, 4};

int max = Arrays.stream(arr12).max().getAsInt();

System.out.println("Maximum value: " + max);

// Example 10: Finding the minimum value in an array

int[] arr13 = {3, 2, 1, 5, 4};

int min = Arrays.stream(arr13).min().getAsInt();

System.out.println("Minimum value: " + min);

}

}

**15) Write a program to accept senior citizens name age from command prompt if age is below 60 throw “invalid age exception” exception.**

import java.util.Scanner;

class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

public class SeniorCitizen {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter senior citizen's name: ");

String name = scanner.nextLine();

System.out.print("Enter senior citizen's age: ");

int age = scanner.nextInt();

try {

if (age < 60) {

throw new InvalidAgeException("Invalid age. Age must be 60 or above.");

}

System.out.println("Senior citizen's name: " + name);

System.out.println("Senior citizen's age: " + age);

} catch (InvalidAgeException e) {

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

}

}

}

Explanation:

1. We first import the Scanner class, which is used to accept user input from the command prompt.
2. We define a custom exception class called "InvalidAgeException" that extends the built-in "Exception" class. This exception will be thrown if the senior citizen's age is below 60.
3. In the main method, we create an instance of the Scanner class and use it to accept the senior citizen's name and age from the command prompt.
4. We use a try-catch block to handle any exceptions that may be thrown. If the age is less than 60, we throw an "InvalidAgeException" with a message "Invalid age. Age must be 60 or above." Otherwise, we print the senior citizen's name and age.
5. If an "InvalidAgeException" is caught, we simply print its error message using the getMessage() method.

**16) Write JSP code to accept registration details for placement portal from candidate and insert the details in database (assume suitable table structure).**

**Below File is JSP program:**

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ page import="java.sql.\*" %>

<%@ page import="javax.sql.\*" %>

<!DOCTYPE html>

<html>

<head>

<title>Placement Portal Registration</title>

</head>

<body>

<h1>Placement Portal Registration Form</h1>

<form method="post" action="register.jsp">

<label for="fullname">Full Name:</label>

<input type="text" name="fullname" required><br>

<label for="email">Email:</label>

<input type="email" name="email" required><br>

<label for="phone">Phone Number:</label>

<input type="tel" name="phone" required><br>

<label for="college">College Name:</label>

<input type="text" name="college" required><br>

<label for="branch">Branch:</label>

<input type="text" name="branch" required><br>

<input type="submit" value="Register">

</form>

<%

// Connect to the database

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/placement\_portal", "root", "");

// Insert the registration details into the database

if(request.getMethod().equals("POST")) {

String fullname = request.getParameter("fullname");

String email = request.getParameter("email");

String phone = request.getParameter("phone");

String college = request.getParameter("college");

String branch = request.getParameter("branch");

PreparedStatement ps = con.prepareStatement("INSERT INTO candidates (fullname, email, phone, college, branch) VALUES (?, ?, ?, ?, ?)");

ps.setString(1, fullname);

ps.setString(2, email);

ps.setString(3, phone);

ps.setString(4, college);

ps.setString(5, branch);

ps.executeUpdate();

out.println("<p>Registration successful!</p>");

}

// Close the database connection

con.close();

%>

</body>

</html>

**Below file is SQL file for the above Program:**

**candidates**

**----------**

**id INT(11) NOT NULL AUTO\_INCREMENT**

**fullname VARCHAR(255) NOT NULL**

**email VARCHAR(255) NOT NULL**

**phone VARCHAR(15) NOT NULL**

**college VARCHAR(255) NOT NULL**

**branch VARCHAR(255) NOT NULL**

**PRIMARY KEY (id)**

**17) Write a JSP program to accept patient details from HTML and display patient details in proper format to update current details**

**This is html file:**

<!DOCTYPE html>

<html>

<head>

<title>Patient Details</title>

</head>

<body>

<h1>Patient Details</h1>

<form method="post" action="patient-details.jsp">

<label for="name">Name:</label>

<input type="text" id="name" name="name"><br><br>

<label for="age">Age:</label>

<input type="number" id="age" name="age"><br><br>

<label for="gender">Gender:</label>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label><br><br>

<label for="address">Address:</label>

<textarea id="address" name="address"></textarea><br><br>

<label for="phone">Phone Number:</label>

<input type="tel" id="phone" name="phone"><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

**This is JSP file :**

**<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Patient Details</title>**

**</head>**

**<body>**

**<h1>Patient Details</h1>**

**<%**

**String name = request.getParameter("name");**

**int age = Integer.parseInt(request.getParameter("age"));**

**String gender = request.getParameter("gender");**

**String address = request.getParameter("address");**

**String phone = request.getParameter("phone");**

**%>**

**<p><strong>Name:</strong> <%=name%></p>**

**<p><strong>Age:</strong> <%=age%></p>**

**<p><strong>Gender:</strong> <%=gender%></p>**

**<p><strong>Address:</strong> <%=address%></p>**

**<p><strong>Phone Number:</strong> <%=phone%></p>**

**<form method="post" action="update-details.jsp">**

**<input type="hidden" name="name" value="<%=name%>">**

**<input type="hidden" name="age" value="<%=age%>">**

**<input type="hidden" name="gender" value="<%=gender%>">**

**<input type="hidden" name="address" value="<%=address%>">**

**<input type="hidden" name="phone" value="<%=phone%>">**

**<input type="submit" value="Update Details">**

**</form>**

**</body>**

**</html>**

**18) Write an application to define a user define exception “Insufficientfund Exception”.read the amount from the console and if the amount is available in your account ,then draw the amount.if the amount is not available,throw “Insufficientfund Exception”and display the amount available for withdrawal**

import java.util.Scanner;

class InsufficientfundException extends Exception {

public InsufficientfundException(String message) {

super(message);

}

}

class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

balance = initialBalance;

}

public void withdraw(double amount) throws InsufficientfundException {

if (balance >= amount) {

balance -= amount;

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

} else {

throw new InsufficientfundException("Insufficient funds. Available balance: " + balance);

}

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double initialBalance = scanner.nextDouble();

BankAccount account = new BankAccount(initialBalance);

System.out.print("Enter withdrawal amount: ");

double amount = scanner.nextDouble();

try {

account.withdraw(amount);

} catch (InsufficientfundException e) {

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

}

}

}

**19) Write a java program to copy the content of two files into one file, read the file names from the users**

import java.io.\*;

public class FileCopy {

public static void main(String[] args) {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

try {

// Get the names of the input files from the user

System.out.print("Enter the name of the first input file: ");

String inputFile1 = br.readLine();

System.out.print("Enter the name of the second input file: ");

String inputFile2 = br.readLine();

// Get the name of the output file from the user

System.out.print("Enter the name of the output file: ");

String outputFile = br.readLine();

// Create input streams for the two input files

FileInputStream fis1 = new FileInputStream(inputFile1);

FileInputStream fis2 = new FileInputStream(inputFile2);

// Create an output stream for the output file

FileOutputStream fos = new FileOutputStream(outputFile);

// Copy the contents of the first input file to the output file

int c;

while ((c = fis1.read()) != -1) {

fos.write(c);

}

// Copy the contents of the second input file to the output file

while ((c = fis2.read()) != -1) {

fos.write(c);

}

// Close the input and output streams

fis1.close();

fis2.close();

fos.close();

System.out.println("Files copied successfully.");

} catch (IOException e) {

System.out.println("An error occurred: " + e.getMessage());

}

}

}

**20) Write a java program to create the following GUI screen using appropriate layout managers using AWT components**

Program:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class Swing2 extends JFrame implements ActionListener

{

JLabel l1,l2,l3;

JButton b;

JRadioButton r1,r2,r3;

JCheckBox c1,c2,c3;

JTextField t1,t2;

ButtonGroup b1;

JPanel p1,p2;

static int cnt;

private StringBuffer s1=new StringBuffer();

Swing2()

{

b1=new ButtonGroup();

p1=new JPanel();

p2=new JPanel();

b=new JButton("Clear");

b.addActionListener(this);

r1=new JRadioButton("FY");

r2=new JRadioButton("SY");

r3=new JRadioButton("TY");

b1.add(r1);

b1.add(r2);

b1.add(r3);

r1.addActionListener(this);

r2.addActionListener(this);

r3.addActionListener(this);

c1=new JCheckBox("Music");

c2=new JCheckBox("Dance");

c3=new JCheckBox("Sports");

c1.addActionListener(this);

c2.addActionListener(this);

c3.addActionListener(this);

l1=new JLabel("Your Name");

l2=new JLabel("Your Class");

l3=new JLabel("Your Hobbies");

t1=new JTextField(20);

t2=new JTextField(30);

p1.setLayout(new GridLayout(5,2));

p1.add(l1);p1.add(t1);

p1.add(l2);p1.add(l3);

p1.add(r1);p1.add(c1);

p1.add(r2); p1.add(c2);

p1.add(r3);p1.add(c3);

p2.setLayout(new FlowLayout());

p2.add(b);

p2.add(t2);

setLayout(new BorderLayout());

add(p1,BorderLayout.NORTH);

add(p2,BorderLayout.EAST);

setSize(400,200);

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void actionPerformed(ActionEvent e)

{

if(e.getSource()==r1)

{

cnt++;

if(cnt==1)

{

String s =t1.getText();

s1.append("Name = ");

s1.append(s);

}

s1.append(" Class = FY");

}

else if(e.getSource()==r2)

{

cnt++;

if(cnt==1)

{

String s =t1.getText();

s1.append("Name = ");

s1.append(s);

}

s1.append(" Class = SY");

}

else if(e.getSource()==r3)

{

cnt++;

if(cnt==1)

{

String s =t1.getText();

s1.append("Name = ");

s1.append(s);

}

s1.append(" Class = TY");

}

else if(e.getSource()==c1)

{

s1.append(" Hobbies = Music");

}

else if(e.getSource()==c2)

{

s1.append(" Hobbies = Dance");

}

else if(e.getSource()==c3)

{

s1.append(" Hobbies = Sports");

}

t2.setText(new String(s1));

// t2.setText(s2);

if(e.getSource()==b)

{

t2.setText(" ");

t1.setText(" ");

}

}

public static void main(String arg[])

{

Swing2 s=new Swing2();

}

}

23) write JDBC application to register students for convocation in MCA (Use PreparedStatement, assume suitable table structure).

Table Name: students Columns:

* id (int, auto-incremented primary key)
* name (varchar)
* email (varchar)
* phone (varchar)
* cgpa (double)

Table Name: convocation Columns:

* id (int, auto-incremented primary key)
* student\_id (int, foreign key referencing students.id)
* registration\_date (date)
* is\_paid (boolean)

Note: In the code, replace **db\_url**, **db\_user**, and **db\_password** with the actual values for your database connection.

PROGRAM:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.sql.Date;

import java.util.Scanner;

public class ConvocationRegistration {

public static void main(String[] args) {

// Get student details from user

Scanner scanner = new Scanner(System.in);

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

String name = scanner.nextLine();

System.out.print("Enter student email: ");

String email = scanner.nextLine();

System.out.print("Enter student phone: ");

String phone = scanner.nextLine();

System.out.print("Enter student CGPA: ");

double cgpa = scanner.nextDouble();

Connection connection = null;

PreparedStatement stmt = null;

try {

// Connect to database

String db\_url = "jdbc:mysql://localhost:3306/mydatabase";

String db\_user = "username";

String db\_password = "password";

connection = DriverManager.getConnection(db\_url, db\_user, db\_password);

// Insert student details into students table

String studentsQuery = "INSERT INTO students (name, email, phone, cgpa) VALUES (?, ?, ?, ?)";

stmt = connection.prepareStatement(studentsQuery);

stmt.setString(1, name);

stmt.setString(2, email);

stmt.setString(3, phone);

stmt.setDouble(4, cgpa);

int studentId = stmt.executeUpdate();

// Register student for convocation in MCA

String convocationQuery = "INSERT INTO convocation (student\_id, registration\_date, is\_paid) VALUES (?, ?, ?)";

stmt = connection.prepareStatement(convocationQuery);

stmt.setInt(1, studentId);

stmt.setDate(2, new Date(System.currentTimeMillis()));

stmt.setBoolean(3, false);

int convocationId = stmt.executeUpdate();

System.out.println("Student registered for convocation with ID " + convocationId);

} catch (SQLException e) {

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

} finally {

try {

if (stmt != null) {

stmt.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

System.out.println("Error closing connection: " + e.getMessage());

}

}

}

}