//Exp1 Factorial,prime numbers,average

import java.util.\*;

public class Class\_1 {

public static int factorial(int n) {

int result = 1;

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

result = result \* i;

}

return result;

}

public static void printPrimeNumbers(int n) {

int count = 0;

int number = 2;

while (count < n) {

boolean isPrime = true;

for (int i = 2; i <= number / 2; i++) {

if (number % i == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

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

count++;

}

number++;

}

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

}

public static void calculateSumAndAverage(int N) {

int sum = 0;

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

sum = sum + i;

}

int average = sum / N;

System.***out***.println("Sum of first " + N + " numbers: " + sum);

System.***out***.println("Average of first " + N + " numbers: " + average);

}

public static void main(String[] args) {

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

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

int numberForFactorial = scanner.nextInt();

System.***out***.println("Factorial of " + numberForFactorial + " is: " + *factorial*(numberForFactorial));

System.***out***.print("Enter the number of prime numbers to display: ");

int primeCount = scanner.nextInt();

System.***out***.print("First " + primeCount + " prime numbers: ");

*printPrimeNumbers*(primeCount);

System.***out***.print("Enter N to calculate the sum and average of the first N numbers: ");

int N = scanner.nextInt();

*calculateSumAndAverage*(N);

scanner.close();

}

}

// Exp 2 Calculator

import java.util.Scanner;

public class Class\_1 {

public static int factorial(int n) {

int result = 1;

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

result = result \* i;

}

return result;

}

public static void main(String[] args) {

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

while (true) {

System.***out***.println("Simple Calculator");

System.***out***.println("Choose an operation:");

System.***out***.println("1. Add");

System.***out***.println("2. Subtract");

System.***out***.println("3. Multiply");

System.***out***.println("4. Divide");

System.***out***.println("5. Factorial");

System.***out***.println("6. Exit");

int choice = scanner.nextInt();

if (choice == 6) {

System.***out***.println("Exiting the calculator. Goodbye!");

break;

}

switch (choice) {

case 1:

System.***out***.print("Enter first number: ");

double num1 = scanner.nextDouble();

System.***out***.print("Enter second number: ");

double num2 = scanner.nextDouble();

System.***out***.println("Result: " + (num1 + num2));

break;

case 2:

System.***out***.print("Enter first number: ");

num1 = scanner.nextDouble();

System.***out***.print("Enter second number: ");

num2 = scanner.nextDouble();

System.***out***.println("Result: " + (num1 - num2));

break;

case 3:

System.***out***.print("Enter first number: ");

num1 = scanner.nextDouble();

System.***out***.print("Enter second number: ");

num2 = scanner.nextDouble();

System.***out***.println("Result: " + (num1 \* num2));

break;

case 4:

System.***out***.print("Enter first number: ");

num1 = scanner.nextDouble();

System.***out***.print("Enter second number: ");

num2 = scanner.nextDouble();

if (num2 != 0) {

System.***out***.println("Result: " + (num1 / num2));

} else {

System.***out***.println("Error: Cannot divide by zero.");

}

break;

case 5:

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

int num = scanner.nextInt();

if (num < 0) {

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

} else {

System.***out***.println("Factorial of " + num + " is: " + *factorial*(num));

}

break;

default:

System.***out***.println("Invalid choice. Please try again.");

break;

}

}

scanner.close();

}

}

//Exp 3 Rectangle Comparison  
  
import java.util.Scanner;

class Rectangle {

double width;

double length;

double area;

String colour;

public double get\_length() {

return length;

}

public double get\_width() {

return width;

}

public String get\_colour() {

return colour;

}

public void find\_area() {

area = length \* width;

}

}

public class Class\_1 {

public static void main(String[] args) {

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

Rectangle rect1 = new Rectangle();

Rectangle rect2 = new Rectangle();

System.***out***.println("Enter details for the first rectangle:");

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

rect1.length = scanner.nextDouble();

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

rect1.width = scanner.nextDouble();

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

rect1.colour = scanner.next();

rect1.find\_area();

System.***out***.println("Enter details for the second rectangle:");

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

rect2.length = scanner.nextDouble();

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

rect2.width = scanner.nextDouble();

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

rect2.colour = scanner.next();

rect2.find\_area();

if (rect1.area == rect2.area && rect1.colour.equals(rect2.colour)) {

System.***out***.println("Matching Rectangles");

} else {

System.***out***.println("Non-matching Rectangle");

}

scanner.close();

}

}

//Exp 4:Method Overloading  
  
import java.util.\*;

class Class\_1 {

// Method Overloading - same method name but different parameters

// Method with one integer parameter

public void display(int a) {

System.***out***.println("Integer: " + a);

}

// Method with two integer parameters

public void display(int a, int b) {

System.***out***.println("Integer 1: " + a + ", Integer 2: " + b);

}

// Method with one double parameter

public void display(double a) {

System.***out***.println("Double: " + a);

}

// Method with one string parameter

public void display(String a) {

System.***out***.println("String: " + a);

}

// Method with mixed types: one integer and one string parameter

public void display(int a, String b) {

System.***out***.println("Integer: " + a + ", String: " + b);

}

public static void main(String[] args) {

Class\_1 obj = new Class\_1();

// Calling the overloaded methods with different parameters

obj.display(10); // Calls the method with one integer

obj.display(10, 20); // Calls the method with two integers

obj.display(3.14); // Calls the method with one double

obj.display("Hello World"); // Calls the method with one string

obj.display(42, "Answer"); // Calls the method with one integer and one string

}

}

//Exp 4:Constructor Overloading

import java.util.\*;

class Class\_1 {

// Data fields

int length;

int width;

// Default Constructor (Constructor Overloading 1)

public Class\_1() {

this.length = 0;

this.width = 0;

System.***out***.println("Default Constructor: Length = " + length + ", Width = " + width);

}

// Parameterized Constructor (Constructor Overloading 2)

public Class\_1(int length, int width) {

this.length = length;

this.width = width;

System.***out***.println("Parameterized Constructor: Length = " + length + ", Width = " + width);

}

// Constructor with one parameter (Constructor Overloading 3)

public Class\_1(int side) {

this.length = side;

this.width = side;

System.***out***.println("Square Constructor: Length = " + length + ", Width = " + width);

}

public static void main(String[] args) {

// Creating objects using different constructors

Class\_1 rect1 = new Class\_1(); // Calls default constructor

Class\_1 rect2 = new Class\_1(10, 20); // Calls parameterized constructor

Class\_1 square = new Class\_1(15); // Calls constructor for square (one parameter)

}

}

//Exp 5 :Create and Sort array for List of integers

import java.util.\*;

public class Class\_1 {

public static void main(String[] args) {

// Create an array of integers

int[] numbers = {32, 5, 43, 12, 9, 67, 23};

// Display the original array

System.***out***.println("Original Array:");

*printArray*(numbers);

// Menu for user choice

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

System.***out***.println("\nChoose sorting method:");

System.***out***.println("1. Built-in Sorting (Arrays.sort())");

System.***out***.println("2. Manual Sorting (Bubble Sort)");

System.***out***.print("Enter your choice (1 or 2): ");

int choice = scanner.nextInt();

// Switch case to choose between sorting methods

switch (choice) {

case 1:

// Built-in sorting using Arrays.sort()

Arrays.*sort*(numbers);

System.***out***.println("\nSorted Array using Built-in Sorting (Arrays.sort()):");

*printArray*(numbers);

break;

case 2:

// Manual sorting using Bubble Sort

*bubbleSort*(numbers);

System.***out***.println("\nSorted Array using Manual Sorting:");

*printArray*(numbers);

break;

default:

System.***out***.println("Invalid choice. Please enter 1 or 2.");

break;

}

// Close the scanner

scanner.close();

}

// Method to perform Bubble Sort

public static void bubbleSort(int[] arr) {

int n = arr.length;

boolean swapped;

// Outer loop for each pass through the array

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

swapped = false;

// Inner loop for comparing adjacent elements

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

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

// Swap the elements if they are in the wrong order

int temp = arr[j];

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

arr[j + 1] = temp;

swapped = true;

}

}

// If no elements were swapped in this pass, the array is sorted

if (!swapped) break;

}

}

// Method to print the array

public static void printArray(int[] arr) {

for (int i : arr) {

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

}

System.***out***.println(); // Newline after printing the array

}

}

//Exp 5:Create and sort array for list of names

import java.util.\*;

public class Class\_1 {

public static void main(String[] args) {

// Create an array of names

String[] names = { "John", "Alice", "Bob", "Charlie", "David" };

// Display the original array of names

System.***out***.println("Original Array of Names:");

*printArray*(names);

// Menu for user choice

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

System.***out***.println("\nChoose sorting method:");

System.***out***.println("1. Built-in Sorting (Arrays.sort())");

System.***out***.println("2. Manual Sorting (Bubble Sort)");

System.***out***.print("Enter your choice (1 or 2): ");

int choice = scanner.nextInt();

// Switch case to choose between sorting methods

switch (choice) {

case 1:

// Built-in sorting using Arrays.sort()

Arrays.*sort*(names);

System.***out***.println("\nSorted Array of Names using Built-in Sorting (Arrays.sort()):");

*printArray*(names);

break;

case 2:

// Manual sorting using Bubble Sort

*bubbleSort*(names);

System.***out***.println("\nSorted Array of Names using Manual Sorting (Bubble Sort):");

*printArray*(names);

break;

default:

System.***out***.println("Invalid choice. Please enter 1 or 2.");

break;

}

// Close the scanner

scanner.close();

}

// Method to perform Bubble Sort on an array of strings

public static void bubbleSort(String[] arr) {

int n = arr.length;

boolean swapped;

// Outer loop for each pass through the array

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

swapped = false;

// Inner loop for comparing adjacent elements

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

// Compare adjacent elements using compareTo for lexicographical order

if (arr[j].compareTo(arr[j + 1]) > 0) {

// Swap the elements if they are in the wrong order

String temp = arr[j];

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

arr[j + 1] = temp;

swapped = true;

}

}

// If no elements were swapped in this pass, the array is already sorted

if (!swapped) break;

}

}

// Method to print the array of names

public static void printArray(String[] arr) {

for (String name : arr) {

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

}

System.***out***.println(); // Print a new line after the array

}

}

//Exp 6:Matrix Addition  
  
import java.util.\*;

public class Class\_1 {

public static void main(String[] args) {

// Initialize Scanner to take user input

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

// Taking matrix dimensions input

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

int rows = scanner.nextInt();

System.***out***.print("Enter the number of columns: ");

int cols = scanner.nextInt();

// Create two matrices of the given size

int[][] matrix1 = new int[rows][cols];

int[][] matrix2 = new int[rows][cols];

int[][] resultMatrix = new int[rows][cols]; // Resultant matrix for storing the sum

// Input matrix 1

System.***out***.println("Enter elements of matrix 1:");

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

for (int j = 0; j < cols; j++) {

System.***out***.print("Element at [" + i + "][" + j + "]: ");

matrix1[i][j] = scanner.nextInt();

}

}

// Input matrix 2

System.***out***.println("Enter elements of matrix 2:");

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

for (int j = 0; j < cols; j++) {

System.***out***.print("Element at [" + i + "][" + j + "]: ");

matrix2[i][j] = scanner.nextInt();

}

}

// Perform matrix addition

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

for (int j = 0; j < cols; j++) {

resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

// Displaying the result matrix

System.***out***.println("\nResult of Matrix Addition:");

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

for (int j = 0; j < cols; j++) {

System.***out***.print(resultMatrix[i][j] + " ");

}

System.***out***.println(); // Move to the next line for each row

}

// Close the scanner

scanner.close();

}

}

//Exp 7: Demonstrate the concept of inheritance in

//JAVA by designing a Player class. Inherit

//the Player class to Cricket\_player,

//Football\_player and Hockey\_player.

import java.util.\*;

class Player {

String name;

int age;

String sportType;

// Constructor

public Player(String name, int age, String sportType) {

this.name = name;

this.age = age;

this.sportType = sportType;

}

// Method to display player details

public void displayDetails() {

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

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

System.***out***.println("Sport Type: " + sportType);

}

// Method to describe the player’s role in their sport (can be overridden)

public void play() {

System.***out***.println(name + " is playing the sport: " + sportType);

}

}

// Subclass: Cricket\_player (inherits from Player)

class Cricket\_player extends Player {

String battingStyle;

// Constructor

public Cricket\_player(String name, int age, String battingStyle) {

super(name, age, "Cricket"); // Call to the base class constructor

this.battingStyle = battingStyle;

}

// Overriding the play method for Cricket Player

*@Override*

public void play() {

System.***out***.println(name + " is playing Cricket with a " + battingStyle + " batting style.");

}

// Method to display cricket-specific details

public void displayCricketDetails() {

System.***out***.println("Batting Style: " + battingStyle);

}

}

// Subclass: Football\_player (inherits from Player)

class Football\_player extends Player {

String position;

// Constructor

public Football\_player(String name, int age, String position) {

super(name, age, "Football"); // Call to the base class constructor

this.position = position;

}

// Overriding the play method for Football Player

*@Override*

public void play() {

System.***out***.println(name + " is playing Football as a " + position + ".");

}

// Method to display football-specific details

public void displayFootballDetails() {

System.***out***.println("Position: " + position);

}

}

// Subclass: Hockey\_player (inherits from Player)

class Hockey\_player extends Player {

String stickHand;

// Constructor

public Hockey\_player(String name, int age, String stickHand) {

super(name, age, "Hockey"); // Call to the base class constructor

this.stickHand = stickHand;

}

// Overriding the play method for Hockey Player

*@Override*

public void play() {

System.***out***.println(name + " is playing Hockey with a " + stickHand + " hand grip.");

}

// Method to display hockey-specific details

public void displayHockeyDetails() {

System.***out***.println("Stick Hand: " + stickHand);

}

}

// Main class to test inheritance

public class Class\_1 {

public static void main(String[] args) {

// Creating instances of each type of player

Cricket\_player cricketPlayer = new Cricket\_player("Sachin Tendulkar", 47, "Right-handed");

Football\_player footballPlayer = new Football\_player("Lionel Messi", 33, "Forward");

Hockey\_player hockeyPlayer = new Hockey\_player("Wayne Gretzky", 59, "Left");

// Displaying details of each player

System.***out***.println("Cricket Player Details:");

cricketPlayer.displayDetails();

cricketPlayer.displayCricketDetails();

cricketPlayer.play();

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

System.***out***.println("Football Player Details:");

footballPlayer.displayDetails();

footballPlayer.displayFootballDetails();

footballPlayer.play();

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

System.***out***.println("Hockey Player Details:");

hockeyPlayer.displayDetails();

hockeyPlayer.displayHockeyDetails();

hockeyPlayer.play();

}

}

//Exp 8: Build the concept of multiple inheritance by

//implementing interface features of JAVA

//programming

import java.util.\*;

// Interface 1: Sports

interface Sports {

void playSport();

}

// Interface 2: Music

interface Music {

void playInstrument();

}

// Concrete class: Person implementing both Sports and Music interfaces

class Person implements Sports, Music {

String name;

// Constructor to initialize name

public Person(String name) {

this.name = name;

}

// Implementing method from Sports interface

*@Override*

public void playSport() {

System.***out***.println(name + " is playing football.");

}

// Implementing method from Music interface

*@Override*

public void playInstrument() {

System.***out***.println(name + " is playing the guitar.");

}

}

// Main class to test the multiple inheritance using interfaces

public class Class\_1 {

public static void main(String[] args) {

// Creating an object of Person

Person person = new Person("John");

// Calling methods from both interfaces

person.playSport(); // From Sports interface

person.playInstrument(); // From Music interface

}

}

//Exp 9: Implement the exception handling using try

//and catch statements to solve runtime

//errors.

public class Class\_1 {

public static void main(String[] args) {

// Example 1: Handling ArithmeticException (Divide by zero)

try {

int num1 = 10;

int num2 = 0; // This will cause an ArithmeticException (division by zero)

int result = num1 / num2; // Division by zero

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

} catch (ArithmeticException e) {

System.***out***.println("Error: Cannot divide by zero.");

}

// Example 2: Handling ArrayIndexOutOfBoundsException

try {

int[] arr = new int[3]; // Array of size 3

arr[5] = 10; // This will cause an ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

System.***out***.println("Error: Array index is out of bounds.");

}

// Example 3: Handling Multiple Exceptions

try {

String input = null;

System.***out***.println(input.length()); // This will cause a NullPointerException

} catch (NullPointerException e) {

System.***out***.println("Error: Null pointer exception occurred.");

}

// Example 4: Finally block to execute cleanup code

try {

System.***out***.println("Try block is executing...");

int[] nums = new int[2];

nums[3] = 10; // This will throw ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

System.***out***.println("Error: Array index is out of bounds.");

} finally {

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

}

System.***out***.println("Program continues after handling exceptions.");

}

}

//Exp 10:File handling

import java.io.FileWriter;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

public class Main {

public static void main(String[] args) {

File inputfile= new File("input.txt");

File outputfile= new File("output.txt");

FileWriter output =null;

FileReader input =null;

try{

if(!inputfile.exists()){

inputfile.createNewFile();

}

input = new FileReader(inputfile);

output = new FileWriter(outputfile);

int c;

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

output.write(c);

}

input.close();

output.close();

}

catch(IOException e){

}

}

}

//Exp 9: Implement the exception handling using try

//and catch statements to solve runtime

//errors.

//Chatgpt

import java.io.FileWriter;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.io.\*;

import java.io.\*;

public class Class\_1 {

public static void main(String[] args) {

// Source file path and destination file path

String sourceFile = "source.txt";

String destinationFile = "destination.txt";

// File reader and writer objects

BufferedReader reader = null;

BufferedWriter writer = null;

try {

// Initialize BufferedReader to read from the source file

reader = new BufferedReader(new FileReader(sourceFile));

// Initialize BufferedWriter to write to the destination file

writer = new BufferedWriter(new FileWriter(destinationFile));

String line;

// Read line by line from the source file

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

// Write each line to the destination file

writer.write(line);

writer.newLine(); // Add a new line after each line

}

System.***out***.println("File has been copied successfully.");

} catch (IOException e) {

System.***out***.println("An error occurred while reading or writing the file.");

e.printStackTrace();

} finally {

try {

// Close the resources

if (reader != null) reader.close();

if (writer != null) writer.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

//Multithreading

// Thread 1: Extending the Thread class

class Thread1 extends Thread {

public void run() {

System.***out***.println("Thread 1 is running");

System.***out***.println("Thread 1 has finished");

}

}

// Thread 2: Implementing the Runnable interface

class Thread2 implements Runnable {

public void run() {

System.***out***.println("Thread 2 is running");

System.***out***.println("Thread 2 has finished");

}

}

public class Class\_1 {

public static void main(String[] args) {

// Creating and starting Thread1 (extending Thread class)

Thread1 t1 = new Thread1();

t1.start();

// Creating and starting Thread2 (implementing Runnable interface)

Thread2 t2 = new Thread2();

Thread t2Thread = new Thread(t2);

t2Thread.start();

}

}

//Draw different geometrical figures like

//oval, rectangle, line, text using graphics

//class.

import java.awt.\*;

import java.awt.event.\*;

public class Class\_2 extends Frame {

// Constructor to set up the frame

public Class\_2() {

// Set the title of the frame

setTitle("Geometric Shapes using AWT");

// Set the size of the frame

setSize(600, 400);

// Close the application when the user clicks the close button

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we) {

System.*exit*(0);

}

});

// Make the frame visible

setVisible(true);

}

// Overriding the paint() method to draw shapes

public void paint(Graphics g) {

// Set font for the text

Font customFont = new Font("Arial", Font.***BOLD*** | Font.***ITALIC***, 18); // Create a new font (Arial, bold and italic, size 18)

g.setFont(customFont); // Set the font for drawing text

// Drawing a Rectangle (Outline)

g.setColor(Color.***RED***); // Set color to red

g.drawRect(50, 50, 200, 100); // Draw a rectangle at (50, 50) with width 200 and height 100

// Drawing a Filled Rectangle

g.setColor(Color.***YELLOW***); // Set color to yellow

g.fillRect(300, 50, 200, 100); // Draw a filled rectangle at (300, 50) with width 200 and height 100

// Drawing an Oval (Outline)

g.setColor(Color.***BLUE***); // Set color to blue

g.drawOval(50, 200, 200, 100); // Draw an oval at (50, 200) with width 200 and height 100

// Drawing a Filled Oval

g.setColor(Color.***GREEN***); // Set color to green

g.fillOval(300, 200, 200, 100); // Draw a filled oval at (300, 200) with width 200 and height 100

// Drawing a Line

g.setColor(Color.***BLACK***); // Set color to black

g.drawLine(50, 350, 550, 350); // Draw a line from (50, 350) to (550, 350)

// Drawing Text

g.setColor(Color.***MAGENTA***); // Set color to magenta

g.drawString("Hello, AWT Graphics!", 200, 380); // Draw text at (200, 380)

}

public static void main(String[] args) {

// Create an instance of the GeometricShapesAWT class

new Class\_2();

}

}