**CSE1007 –JAVA PROGRAMMING**

Programs on Looping, Array and String, Class Fundamentals

**ASSESSMENT 1**

1. **Salient Features of Java Programming Language and Real Time Applications of Java?**

|  |
| --- |
| **Salient Features of Java Programming Language:**   * **Platform Independence (Write Once, Run Anywhere):** Java programs are compiled into an intermediate form called bytecode, which can run on any platform with a Java Virtual Machine (JVM). This allows Java applications to be platform independent. * **Object-Oriented:** Java follows the object-oriented programming (OOP) paradigm, making it easier to organize and manage code through encapsulation, inheritance, and polymorphism. * **Automatic Memory Management:** Java includes a garbage collector that automatically manages memory by reclaiming unused memory resources, reducing the risk of memory leaks and manual memory management errors. * **Strongly Typed:** Java enforces strong type checking during both compile-time and runtime, which enhances code reliability and reduces errors. * **Multi-threading Support:** Java has built-in support for multithreading, allowing the creation and management of multiple threads within a single program. This is especially useful for concurrent programming. * **Exception Handling:** Java provides a robust exception handling mechanism, which helps in writing more reliable and error-tolerant code. * **Rich Standard Library:** Java comes with a comprehensive standard library (Java API) that provides prebuilt classes and methods for various common programming tasks, making development faster and more efficient. * **Security:** Java includes built-in security features like class loaders, bytecode verification, and security managers to create a secure execution environment. * **Network and I/O Support:** Java offers libraries for network and I/O operations, making it easy to create networked applications and handle input/output operations. * **Dynamic Loading:** Java allows classes to be loaded at runtime, enabling dynamic loading of classes and components. * **Rich Ecosystem:** Java has a vast and active community, along with numerous third-party libraries, frameworks, and tools that aid in various types of application development.   **Real-time Applications of Java:**   * **Web Applications:** Java is widely used for building dynamic web applications using frameworks like Spring, Java Server Faces (JSF), and Apache Struts. These applications often use technologies like Servlets, JSP, and Java EE. * **Mobile Applications:** Java is used in Android app development, as Android applications are primarily built using Java and the Android SDK. * **Desktop Applications:** Java Swing and JavaFX libraries are used to create graphical user interfaces (GUI) for desktop applications. Examples include integrated development environments (IDEs) like Eclipse and NetBeans. * **Enterprise Applications:** Java EE (Enterprise Edition) is used for developing large-scale enterprise applications, including customer relationship management (CRM) systems, financial applications, and more. * **Embedded Systems:** Java is employed in embedded systems development, especially in scenarios where platform independence is crucial, such as industrial automation and IoT devices. * **Scientific and Research Applications:** Java is used for scientific computing, simulations, and data analysis due to its performance and extensive libraries. * **Gaming:** Java is used for creating games, both browser-based and standalone, using libraries like LibGDX and LWJGL (Lightweight Java Game Library). * **Financial Applications:** Java's robustness, security features, and libraries make it suitable for developing financial software, such as trading platforms and banking applications. * **Healthcare Applications:** Java is used in healthcare systems to develop electronic health records (EHR) systems, medical billing software, and telemedicine applications. * **E-commerce Applications:** Java is used for building e-commerce platforms, handling online transactions, and managing inventory and user data.   Java's versatility, strong features, and large community support make it suitable for a wide range of application domains. |

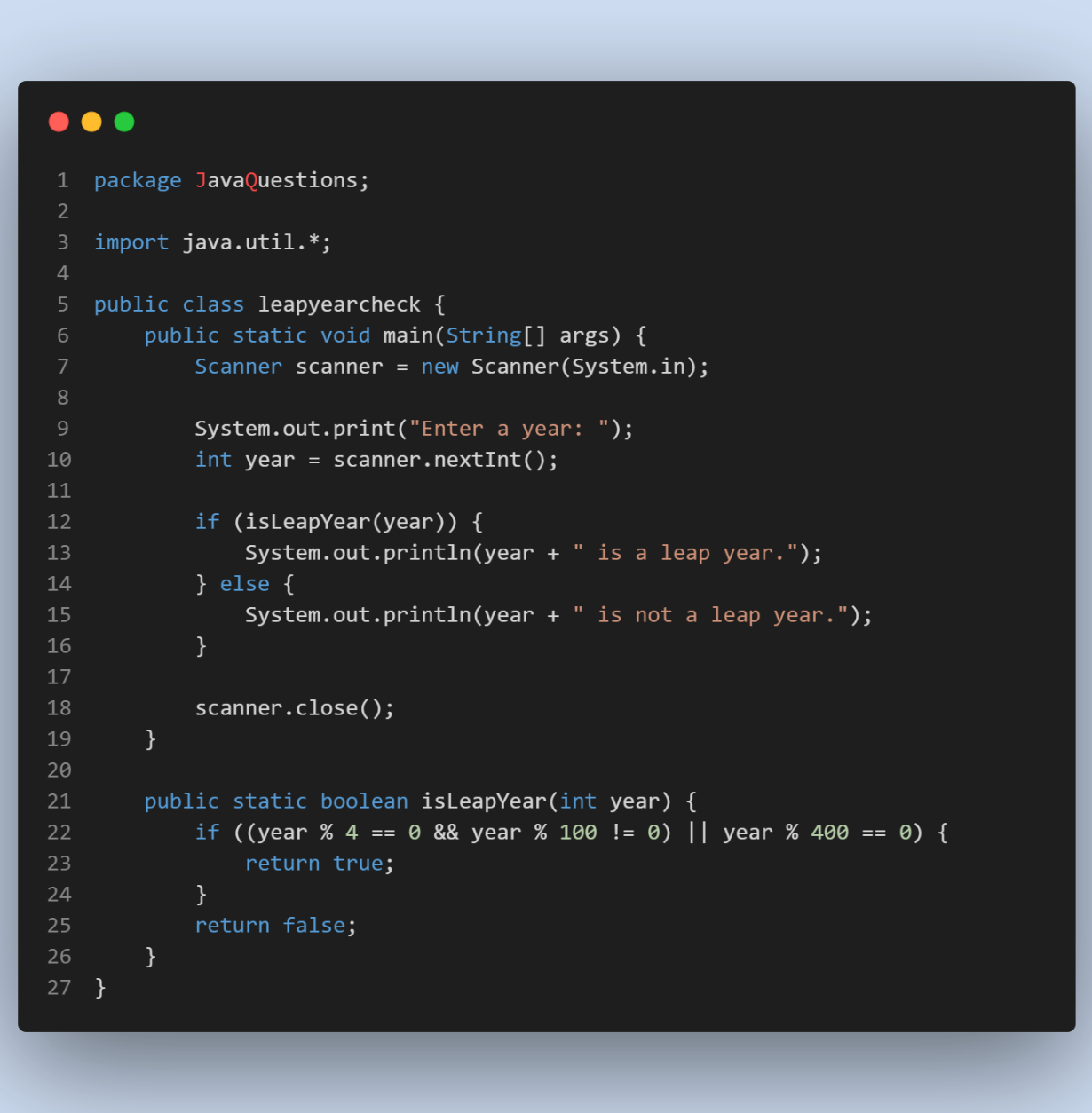
1. **Check whether a given year is leap year or not.**

**Approach:**

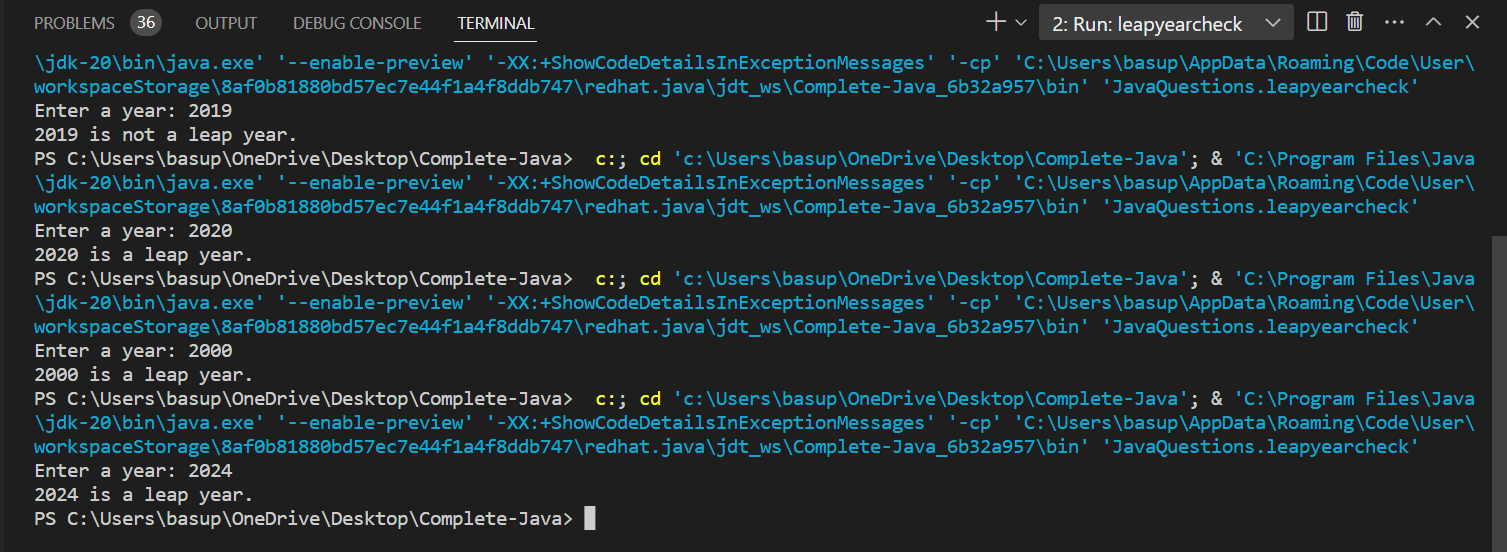
1. Import the Scanner class to get input from the user.
2. Prompt the user to enter a year.
3. Read the year input from the user.
4. Call the isLeapYear method to determine if the given year is a leap year or not.
5. The isLeapYear method checks if the year is divisible by 4 but not divisible by 100, or if it's divisible by 400. If either of these conditions is true, the year is a leap year.

|  |
| --- |
| **Program code:**  package JavaQuestions;  import java.util.\*;  public class leapyearcheck {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a year: ");  int year = scanner.nextInt();  if (isLeapYear(year)) {  System.out.println(year + " is a leap year.");  } else {  System.out.println(year + " is not a leap year.");  }  scanner.close();  }  public static boolean isLeapYear(int year) {  if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {  return true;  }  return false;  }  } |

**Code Screenshot:**



**Output:**

****

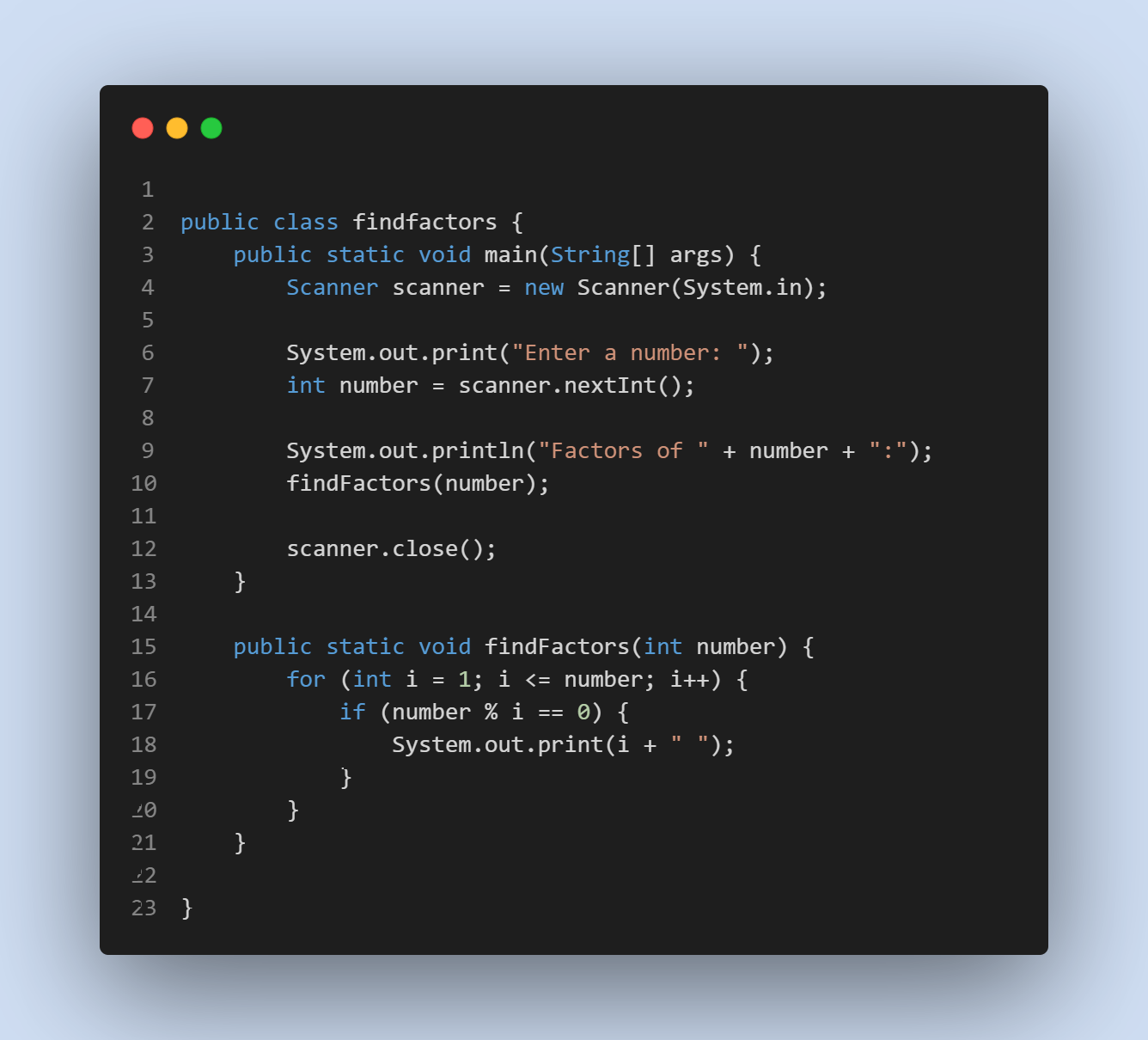
1. **Find the factors of a given number.**

**Approach:**

1. Import the Scanner class to get input from the user.
2. Prompt the user to enter a number.
3. Read the number input from the user.
4. Call the findFactors method to find and print the factors of the given number.
5. The findFactors method iterates through numbers from 1 to the given number and checks if the given number is divisible by the current number. If it is divisible, the current number is a factor and is printed.

|  |
| --- |
| **Program Code:**  package JavaQuestions;  import java.util.Scanner;  public class findfactors {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a number: ");  int number = scanner.nextInt();  System.out.println("Factors of " + number + ":");  findFactors(number);  scanner.close();  }  public static void findFactors(int number) {  for (int i = 1; i <= number; i++) {  if (number % i == 0) {  System.out.print(i + " ");  }  }  }  } |

**Code Screenshot:**



**Output:**

**A screenshot of a computer program

Description automatically generated**

1. **Display sum of series 1+x2+x3+x4+x5...+xn.**

**Approach:**

1. Import the Scanner class to get input from the user.
2. Prompt the user to enter the value of x (base) and n (number of terms).
3. Read the input values for x and n.
4. Call the calculateSeriesSum method to compute the sum of the series.
5. The calculateSeriesSum method uses a loop to calculate each term of the series (x^0, x^1, x^2, ..., x^n) and accumulates them in the sum variable.
6. Finally, the calculated sum is displayed.

|  |
| --- |
| **Program Code:**  package JavaQuestions;  import java.util.Scanner;  public class SeriesSumCalculator {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the value of x: ");  double x = scanner.nextDouble();  System.out.print("Enter the value of n: ");  int n = scanner.nextInt();  double sum = calculateSeriesSum(x, n);  System.out.println("Sum of the series: " + sum);  scanner.close();  }  public static double calculateSeriesSum(double x, int n) {  double sum = 0;  for (int i = 0; i <= n; i++) {  sum += Math.pow(x, i);  }  return sum;  }  } |

**Code Screenshot:**

A screenshot of a computer program

Description automatically generated

**Output:**

**A screenshot of a computer

Description automatically generated**

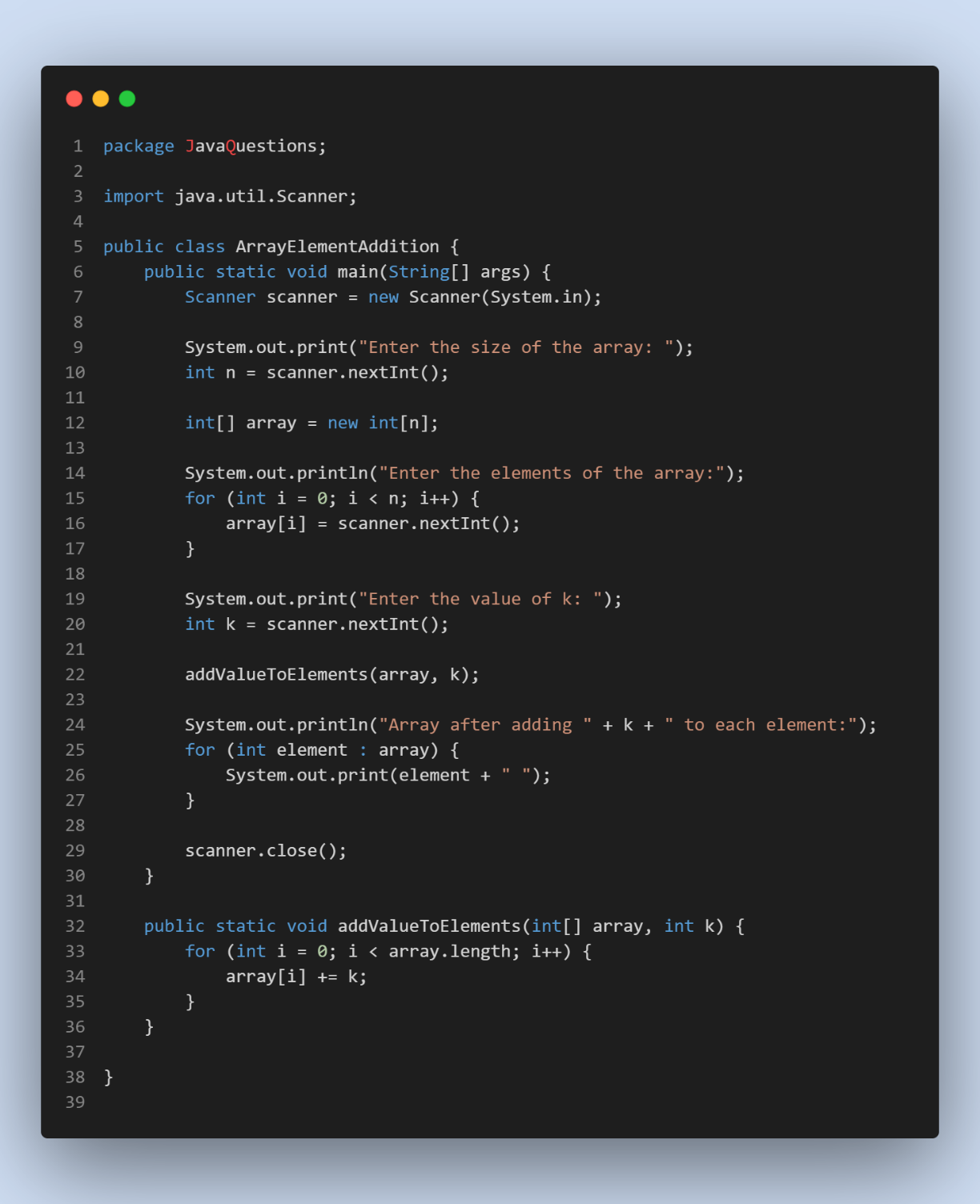
1. **Add a value ‘k’ to all elements in each array of size ‘n’. Get the value of n and k from users [Example: If A=[2,3,5, 6] and k=5 then A=[7,8,10,11]].**

**Approach:**

1. Import the Scanner class to get input from the user.
2. Prompt the user to enter the size of the array (n).
3. Create an array of integers of size n.
4. Prompt the user to enter the elements of the array.
5. Read the array elements from the user.
6. Prompt the user to enter the value of k.
7. Call the addValueToElements method to add the value of k to each element in the array.
8. The addValueToElements method iterates through the array and adds the value of k to each element.
9. Finally, the modified array is displayed.

|  |
| --- |
| **Program code:**  package JavaQuestions;  import java.util.Scanner;  public class ArrayElementAddition {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int n = scanner.nextInt();  int[] array = new int[n];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < n; i++) {  array[i] = scanner.nextInt();  }  System.out.print("Enter the value of k: ");  int k = scanner.nextInt();  addValueToElements(array, k);  System.out.println("Array after adding " + k + " to each element:");  for (int element : array) {  System.out.print(element + " ");  }  scanner.close();  }  public static void addValueToElements(int[] array, int k) {  for (int i = 0; i < array.length; i++) {  array[i] += k;  }  }  } |

**Code Screenshot:**



**Output:**

**A screenshot of a computer

Description automatically generated**

1. **Display the number of occurrences of a particular character in a string.**

**Approach:**

1. Import the Scanner class to get input from the user.
2. Prompt the user to enter a string.
3. Read the input string from the user.
4. Prompt the user to enter the character to be counted.
5. Read the character input from the user.
6. Call the countOccurrences method to count the occurrences of the target character in the input string.
7. The countOccurrences method iterates through each character in the string and checks if it matches the target character. If it does, the count is incremented.
8. Finally, the code displays the number of occurrences of the target character in the input string.

|  |
| --- |
| **Program code:**  package JavaQuestions;  import java.util.Scanner;  public class CharacterOccurrences {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter a string: ");  String inputString = scanner.nextLine();  System.out.print("Enter the character to count: ");  char targetCharacter = scanner.next().charAt(0);  int occurrences = countOccurrences(inputString, targetCharacter);  System.out.println("The character '" + targetCharacter + "' appears " + occurrences + " times in the string.");  scanner.close();  }  public static int countOccurrences(String str, char target) {  int count = 0;  for (char c : str.toCharArray()) {  if (c == target) {  count++;  }  }  return count;  }  } |

**Code Screenshot:**

A screenshot of a computer program

Description automatically generated

**Output:**

**A screenshot of a computer program

Description automatically generated**

1. **Sort the names in ascending order.**

**Approach:**

1. Input: Start by getting the number of names (n) from the user. Then, create an array of strings to store the names.
2. Input Names: Get the n names from the user and store them in the array.
3. Sorting Algorithm: Implement a sorting algorithm to arrange the names in ascending order. One approach is the bubble sort algorithm.
4. **Bubble Sort Algorithm:**
5. Start with the first element of the array and compare it with the next element.
6. If the first element is greater than the second element, swap them.
7. Move to the next pair of adjacent elements and repeat the comparison and swap process.
8. After one pass, the largest element will have "bubbled up" to the last position. Repeat the process for the remaining elements (excluding the last one).
9. Continue this process until the entire array is sorted.
10. **Display Sorted Names:** After sorting, display the names in ascending order.

|  |
| --- |
| Program Code:  package JavaQuestions;  import java.util.Scanner;  public class NameSorter {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the number of names: ");  int n = scanner.nextInt();  scanner.nextLine(); // Consume the newline left by nextInt()  String[] names = new String[n];  System.out.println("Enter the names:");  for (int i = 0; i < n; i++) {  names[i] = scanner.nextLine();  }  sortNames(names);  System.out.println("Names in ascending order:");  for (String name : names) {  System.out.println(name);  }  scanner.close();  }  public static void sortNames(String[] names) {  int n = names.length;  for (int i = 0; i < n - 1; i++) {  for (int j = 0; j < n - i - 1; j++) {  if (names[j].compareTo(names[j + 1]) > 0) {  // Swap names[j] and names[j + 1]  String temp = names[j];  names[j] = names[j + 1];  names[j + 1] = temp;  }  }  }  }  } |

**Code Screenshot:**

A screen shot of a computer program

Description automatically generated

**Output:**

**A screenshot of a computer program

Description automatically generated**

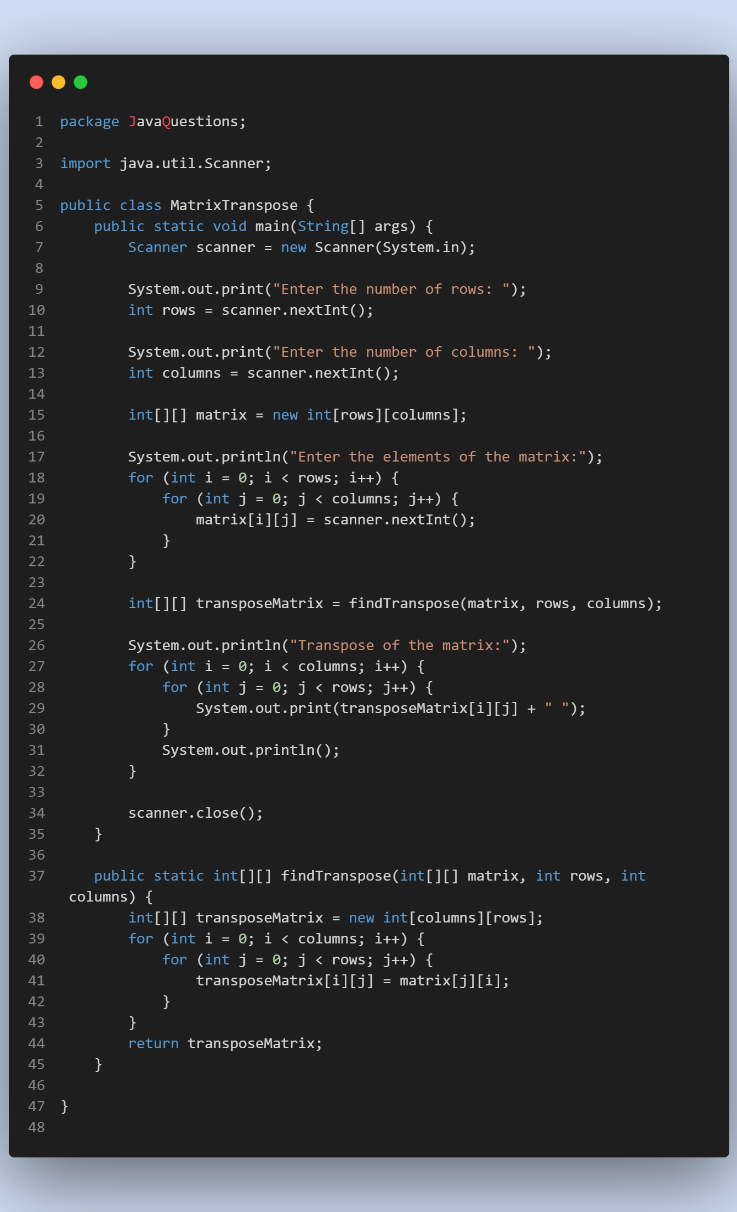
1. **Find the transpose of the given matrix.**

**Approach:**

1. Input: Get the dimensions (number of rows and columns) of the matrix from the user.
2. Input Elements: Input the elements of the matrix.
3. Transpose Operation: For each element at position (i, j), swap it with the element at position (j, i) to find the transpose.
4. Display Transposed Matrix: Display the transposed matrix.

|  |
| --- |
| **Program code:**  package JavaQuestions;  import java.util.Scanner;  public class MatrixTranspose {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the number of rows: ");  int rows = scanner.nextInt();  System.out.print("Enter the number of columns: ");  int columns = scanner.nextInt();  int[][] matrix = new int[rows][columns];  System.out.println("Enter the elements of the matrix:");  for (int i = 0; i < rows; i++) {  for (int j = 0; j < columns; j++) {  matrix[i][j] = scanner.nextInt();  }  }  int[][] transposeMatrix = findTranspose(matrix, rows, columns);  System.out.println("Transpose of the matrix:");  for (int i = 0; i < columns; i++) {  for (int j = 0; j < rows; j++) {  System.out.print(transposeMatrix[i][j] + " ");  }  System.out.println();  }  scanner.close();  }  public static int[][] findTranspose(int[][] matrix, int rows, int columns) {  int[][] transposeMatrix = new int[columns][rows];  for (int i = 0; i < columns; i++) {  for (int j = 0; j < rows; j++) {  transposeMatrix[i][j] = matrix[j][i];  }  }  return transposeMatrix;  }  } |

**Code Screenshot:**



**Output:**

**A computer screen shot of a black screen

Description automatically generated**

1. **Create two classes placed into a package geometry.twodim representing a Square and Triangle with methods to find area and perimeter. Write an application class which imports these packages to create objects and test the methods.**

**Approach:**

1. The Square and Triangle classes are placed inside the geometry.twodim package. They have methods to calculate area and perimeter based on their respective formulas.
2. The packageimporting.java class is outside the geometry.twodim package. It imports the Square and Triangle classes from the package and creates objects to test their methods.
3. Compile and run the packageimporting.java class to see the output of area and perimeter calculations for a square and a triangle.

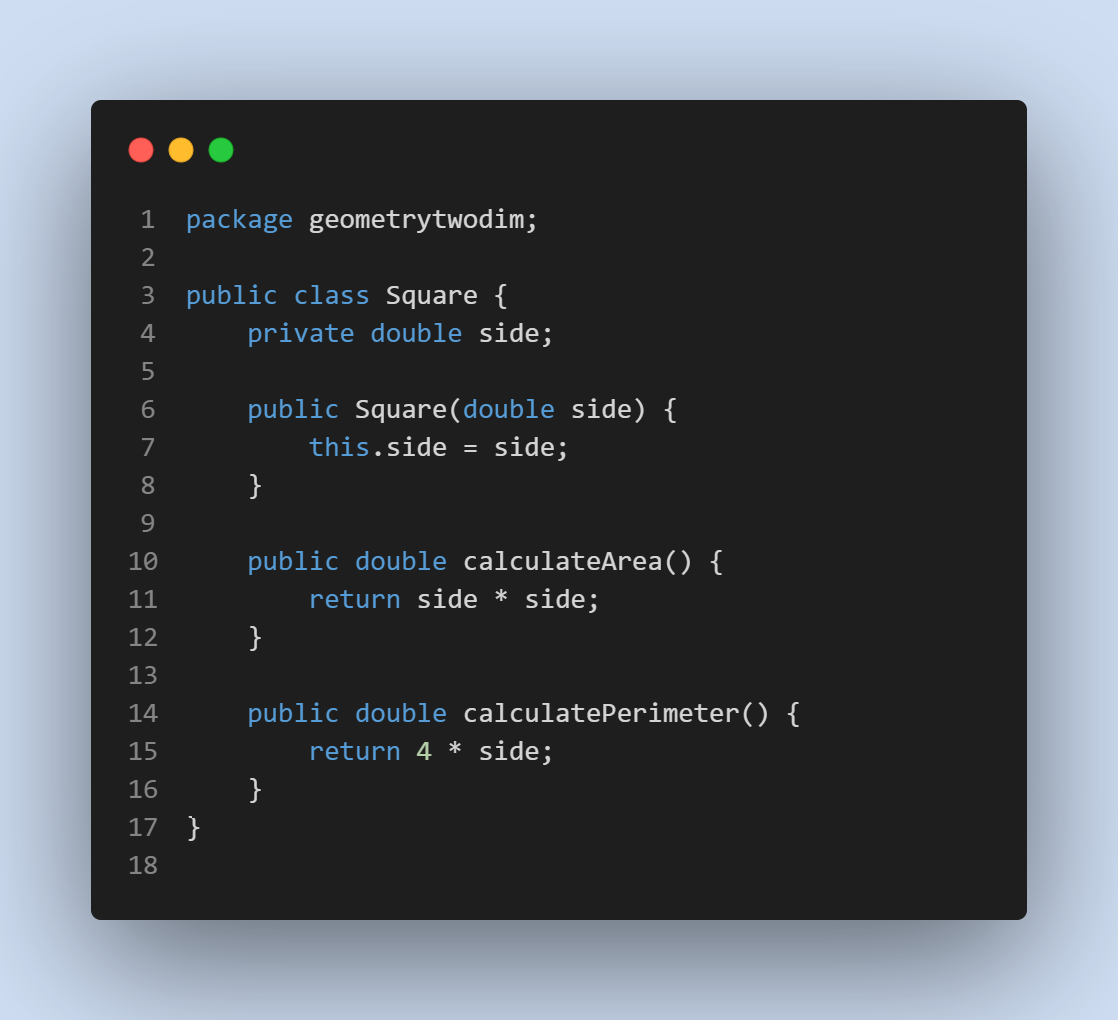
**Package Screenshots:**

**A screenshot of a computer

Description automatically generated**

**Square.java (in package geometrytwodim):**

|  |
| --- |
| **Program code:**  package geometrytwodim;  public class Square {  private double side;  public Square(double side) {  this.side = side;  }  public double calculateArea() {  return side \* side;  }  public double calculatePerimeter() {  return 4 \* side;  }  } |



**Triangle.java (in package geometrytwodim):**

|  |
| --- |
| **Program Code:**  package geometrytwodim;  public class Triangle {  private double base;  private double height;  private double side1;  private double side2;  private double side3;  public Triangle(double base, double height, double side1, double side2, double side3) {  this.base = base;  this.height = height;  this.side1 = side1;  this.side2 = side2;  this.side3 = side3;  }  public double calculateArea() {  return 0.5 \* base \* height;  }  public double calculatePerimeter() {  return side1 + side2 + side3;  }  } |

A screen shot of a computer

Description automatically generated

**GeometryApp.java (outside the package):**

|  |
| --- |
| **Program Code:**  import geometrytwodim.Square;  import geometrytwodim.Triangle;  public class packageimporting {  public static void main(String[] args) {  Square square = new Square(5.0);  System.out.println("Square Area: " + square.calculateArea());  System.out.println("Square Perimeter: " + square.calculatePerimeter());  Triangle triangle = new Triangle(6.0, 4.0, 5.0, 4.0, 3.0);  System.out.println("Triangle Area: " + triangle.calculateArea());  System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());  }  } |

A computer screen shot of a black screen with white text

Description automatically generated

**Output:**

**A computer screen shot of a computer

Description automatically generated**