**Problem Statements:**

1. **Number Guessing Game**:

○ Write a Java program where the user thinks of a number between 1 and 100, and the computer tries to guess the number by generating random guesses.

○ The user provides feedback by indicating whether the guess is **high**, **low**, or **correct**. The program should be modular, with different functions for generating guesses, receiving user feedback, and determining the next guess.

| **import** java.util.Scanner; **import** java.util.Random;  **public** **class** **NumberGuessingGame** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  Random random = **new** Random();  **int** low = 1, high = 100, guess;  String feedback;   System.out.println("Think of a number between 1 and 100. The computer will guess it.");   **do** {  guess = generateGuess(low, high, random);  System.out.println("Is the number " + guess + "? (higher/lower/correct)");  feedback = sc.nextLine().toLowerCase();   **if** (feedback.equals("higher")) {  low = guess + 1;  } **else** **if** (feedback.equals("lower")) {  high = guess - 1;  }  } **while** (!feedback.equals("correct"));   System.out.println("The computer guessed your number!");  sc.close();  }   **public** **static** **int** **generateGuess**(**int** low, **int** high, Random random) {  **return** random.nextInt(high - low + 1) + low;  } } |
| --- |

2. **Maximum of Three Numbers**:

○ Write a program that takes three integer inputs from the user and finds the maximum of the three numbers.

○ Ensure your program follows best practices for organizing code into modular functions, such as separate functions for taking input and calculating the maximum value.

| **import** java.util.Scanner;  **public** **class** **MaxOfThree** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  **int** a = getInput(sc, "Enter first number: ");  **int** b = getInput(sc, "Enter second number: ");  **int** c = getInput(sc, "Enter third number: ");   **int** max = findMax(a, b, c);  System.out.println("The maximum number is: " + max);   sc.close();  }   **public** **static** **int** **getInput**(Scanner sc, String prompt) {  System.out.print(prompt);  **return** sc.nextInt();  }   **public** **static** **int** **findMax**(**int** a, **int** b, **int** c) {  **return** Math.max(a, Math.max(b, c));  } } |
| --- |

3. **Prime Number Checker**:

○ Create a program that checks whether a given number is a prime number. ○ The program should use a separate function to perform the prime check and return the result.

| **import** java.util.Scanner;  **public** **class** **PrimeChecker** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter a number: ");  **int** num = sc.nextInt();   **if** (isPrime(num)) {  System.out.println(num + " is a prime number.");  } **else** {  System.out.println(num + " is not a prime number.");  }   sc.close();  }   **public** **static** **boolean** **isPrime**(**int** num) {  **if** (num < 2)  **return** **false**;  **for** (**int** i = 2; i \* i <= num; i++) {  **if** (num % i == 0)  **return** **false**;  }  **return** **true**;  } } |
| --- |

**Additional Problem Statements:**

4. **Fibonacci Sequence Generator**:

○ Write a program that generates the Fibonacci sequence up to a specified number of terms entered by the user.

○ Organize the code by creating a function that calculates and prints the Fibonacci sequence.

| **import** java.util.Scanner;  **public** **class** **FibonacciGenerator** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter the number of terms: ");  **int** terms = sc.nextInt();   generateFibonacci(terms);   sc.close();  }   **public** **static** **void** **generateFibonacci**(**int** terms) {  **int** a = 0, b = 1;  **for** (**int** i = 0; i < terms; i++) {  System.out.print(a + " ");  **int** next = a + b;  a = b;  b = next;  }  System.out.println();  } } |
| --- |

5. **Palindrome Checker**:

○ Write a program that checks if a given string is a palindrome (a word, phrase, or sequence that reads the same backward as forward).

○ Break the program into functions for input, checking the palindrome condition, and displaying the result.

| **import** java.util.Scanner;  **public** **class** **PalindromeChecker** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter a string: ");  String input = sc.nextLine();   **if** (isPalindrome(input)) {  System.out.println("The string is a palindrome.");  } **else** {  System.out.println("The string is not a palindrome.");  }   sc.close();  }   **public** **static** **boolean** **isPalindrome**(String str) {  str = str.replaceAll("\\s+", "").toLowerCase();  **int** left = 0, right = str.length() - 1;   **while** (left < right) {  **if** (str.charAt(left) != str.charAt(right)) {  **return** **false**;  }  left++;  right--;  }  **return** **true**;  } } |
| --- |

6. **Factorial Using Recursion**:

○ Write a program that calculates the factorial of a number using a recursive function.

○ Include modular code to separate input, calculation, and output processes.

| **import** java.util.Scanner;  **public** **class** **FactorialRecursion** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter a number: ");  **int** num = sc.nextInt();   System.out.println("Factorial of " + num + " is: " + factorial(num));   sc.close();  }   **public** **static** **int** **factorial**(**int** n) {  **return** (n <= 1) ? 1 : n \* factorial(n - 1);  } } |
| --- |

7. **GCD and LCM Calculator**:

○ Create a program that calculates the Greatest Common Divisor (GCD) and Least Common Multiple (LCM) of two numbers using functions.

○ Use separate functions for GCD and LCM calculations, showcasing how modular code works.

| **import** java.util.Scanner;  **public** **class** **GCD\_LCM\_Calculator** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter first number: ");  **int** a = sc.nextInt();  System.out.print("Enter second number: ");  **int** b = sc.nextInt();   System.out.println("GCD: " + gcd(a, b));  System.out.println("LCM: " + lcm(a, b));   sc.close();  }   **public** **static** **int** **gcd**(**int** a, **int** b) {  **return** (b == 0) ? a : gcd(b, a % b);  }   **public** **static** **int** **lcm**(**int** a, **int** b) {  **return** (a \* b) / gcd(a, b);  } } |
| --- |

8. **Temperature Converter**:

○ Write a program that converts temperatures between Fahrenheit and Celsius. ○ The program should have separate functions for converting from Fahrenheit to Celsius and from Celsius to Fahrenheit.

| **import** java.util.Scanner;  **public** **class** **TemperatureConverter** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter temperature: ");  **double** temp = sc.nextDouble();  System.out.print("Convert to (C/F): ");  **char** choice = sc.next().toUpperCase().charAt(0);   **if** (choice == 'C') {  System.out.println("Temperature in Celsius: " + toCelsius(temp));  } **else** **if** (choice == 'F') {  System.out.println("Temperature in Fahrenheit: " + toFahrenheit(temp));  } **else** {  System.out.println("Invalid choice.");  }   sc.close();  }   **public** **static** **double** **toCelsius**(**double** fahrenheit) {  **return** (fahrenheit - 32) \* 5 / 9;  }   **public** **static** **double** **toFahrenheit**(**double** celsius) {  **return** (celsius \* 9 / 5) + 32;  } } |
| --- |

9. **Basic Calculator**:

○ Write a program that performs basic mathematical operations (addition, subtraction, multiplication, division) based on user input.

○ Each operation should be performed in its own function, and the program should prompt the user to choose which operation to perform.

| **import** java.util.Scanner;  **public** **class** **BasicCalculator** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  System.out.print("Enter first number: ");  **double** num1 = sc.nextDouble();  System.out.print("Enter second number: ");  **double** num2 = sc.nextDouble();  System.out.print("Choose operation (+, -, \*, /): ");  **char** op = sc.next().charAt(0);   **double** result = calculate(num1, num2, op);  System.out.println("Result: " + result);   sc.close();  }   **public** **static** **double** **calculate**(**double** a, **double** b, **char** op) {  **switch** (op) {  **case** '+':  **return** add(a, b);  **case** '-':  **return** subtract(a, b);  **case** '\*':  **return** multiply(a, b);  **case** '/':  **return** divide(a, b);  **default**:  **return** Double.NaN;  }  }   **public** **static** **double** **add**(**double** a, **double** b) {  **return** a + b;  }   **public** **static** **double** **subtract**(**double** a, **double** b) {  **return** a - b;  }   **public** **static** **double** **multiply**(**double** a, **double** b) {  **return** a \* b;  }   **public** **static** **double** **divide**(**double** a, **double** b) {  **return** (b != 0) ? a / b : Double.NaN;  } } |
| --- |

**Problem Statements for Practice**

1. **Problem 1: Time Zones and ZonedDateTime** Write a program that displays the current time in different time zones:

➢ GMT (Greenwich Mean Time)

➢ IST (Indian Standard Time)

➢ PST (Pacific Standard Time)

**Hint**: Use ZonedDateTime and ZoneId to work with different time zones.

| **import** java.time.ZoneId; **import** java.time.ZonedDateTime; **import** java.time.format.DateTimeFormatter;  **public** **class** **TimeZones** {  **public** **static** **void** **main**(String[] args) {  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss z");   System.out.println("Current Time in GMT: " + getTimeInZone("GMT", formatter));  System.out.println("Current Time in IST: " + getTimeInZone("Asia/Kolkata", formatter));  System.out.println("Current Time in PST: " + getTimeInZone("America/Los\_Angeles", formatter));  }   **public** **static** String **getTimeInZone**(String zone, DateTimeFormatter formatter) {  **return** ZonedDateTime.now(ZoneId.of(zone)).format(formatter);  } } |
| --- |

2. **Problem 2: Date Arithmetic** Create a program that:

➢ Takes a date input and adds 7 days, 1 month, and 2 years to it.

➢ Then subtracts 3 weeks from the result.

**Hint**: Use LocalDate.plusDays(), plusMonths(), plusYears(), and minusWeeks() methods.

| **import** java.time.LocalDate; **import** java.time.format.DateTimeFormatter; **import** java.util.Scanner;  **public** **class** **DateArithmetic** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");   System.out.print("Enter a date (yyyy-MM-dd): ");  LocalDate date = LocalDate.parse(sc.next(), formatter);   LocalDate modifiedDate = date.plusDays(7).plusMonths(1).plusYears(2).minusWeeks(3);   System.out.println("Final date after modifications: " + modifiedDate.format(formatter));   sc.close();  } } |
| --- |

3. **Problem 3: Date Formatting** Write a program that:

➢ Displays the current date in three different formats:

■ dd/MM/yyyy

■ yyyy-MM-dd

■ EEE, MMM dd, yyyy

**Hint**: Use DateTimeFormatter with custom patterns for date formatting.

| **import** java.time.LocalDate; **import** java.time.format.DateTimeFormatter;  **public** **class** **DateFormatting** {  **public** **static** **void** **main**(String[] args) {  LocalDate currentDate = LocalDate.now();   System.out.println("Format 1 (dd/MM/yyyy): " + formatDate(currentDate, "dd/MM/yyyy"));  System.out.println("Format 2 (yyyy-MM-dd): " + formatDate(currentDate, "yyyy-MM-dd"));  System.out.println("Format 3 (EEE, MMM dd, yyyy): " + formatDate(currentDate, "EEE, MMM dd, yyyy"));  }   **public** **static** String **formatDate**(LocalDate date, String pattern) {  **return** date.format(DateTimeFormatter.ofPattern(pattern));  } } |
| --- |

4. **Problem 4: Date Comparison** Write a program that:

➢ Takes two date inputs and compares them to check if the first date is before, after, or the same as the second date.

**Hint**: Use isBefore(), isAfter(), and isEqual() methods from the LocalDate

Class.

| **import** java.time.LocalDate; **import** java.time.format.DateTimeFormatter; **import** java.util.Scanner;  **public** **class** **DateComparison** {  **public** **static** **void** **main**(String[] args) {  Scanner sc = **new** Scanner(System.in);  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");   System.out.print("Enter first date (yyyy-MM-dd): ");  LocalDate date1 = LocalDate.parse(sc.next(), formatter);  System.out.print("Enter second date (yyyy-MM-dd): ");  LocalDate date2 = LocalDate.parse(sc.next(), formatter);   compareDates(date1, date2);   sc.close();  }   **public** **static** **void** **compareDates**(LocalDate date1, LocalDate date2) {  **if** (date1.isBefore(date2)) {  System.out.println("The first date is before the second date.");  } **else** **if** (date1.isAfter(date2)) {  System.out.println("The first date is after the second date.");  } **else** {  System.out.println("Both dates are the same.");  }  } } |
| --- |