Lab Task: 5

Name: Sidratul Muntaha

ID: 24070112

Dept: CSE

Batch: 43rd (A)   Sem: 2nd

6. Post Lab Exercise:

a. Write a Java program to convert temperature from Fahrenheit to Celsius degree.

Test Data

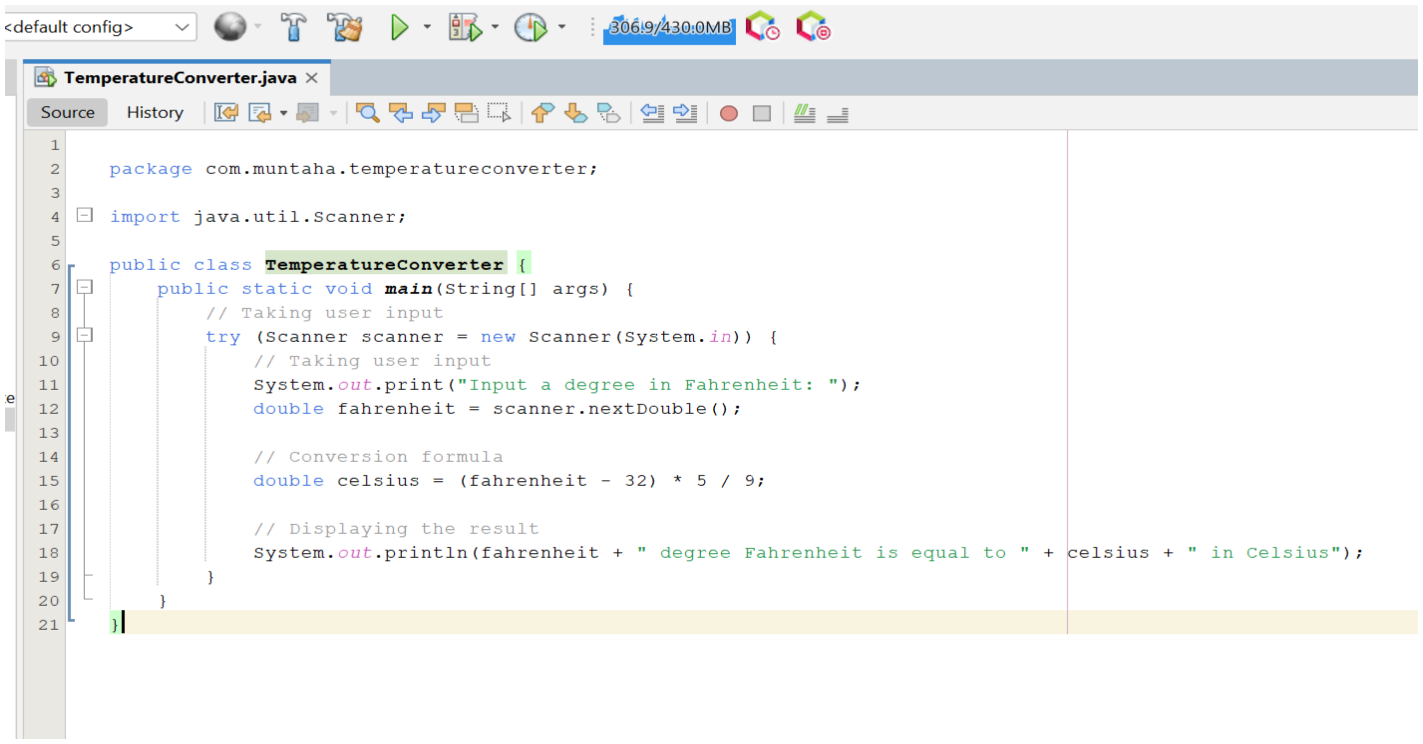
Input a degree in Fahrenheit: 212

Expected Output :

212.0 degree Fahrenheit is equal to 100.0 in Celsius.

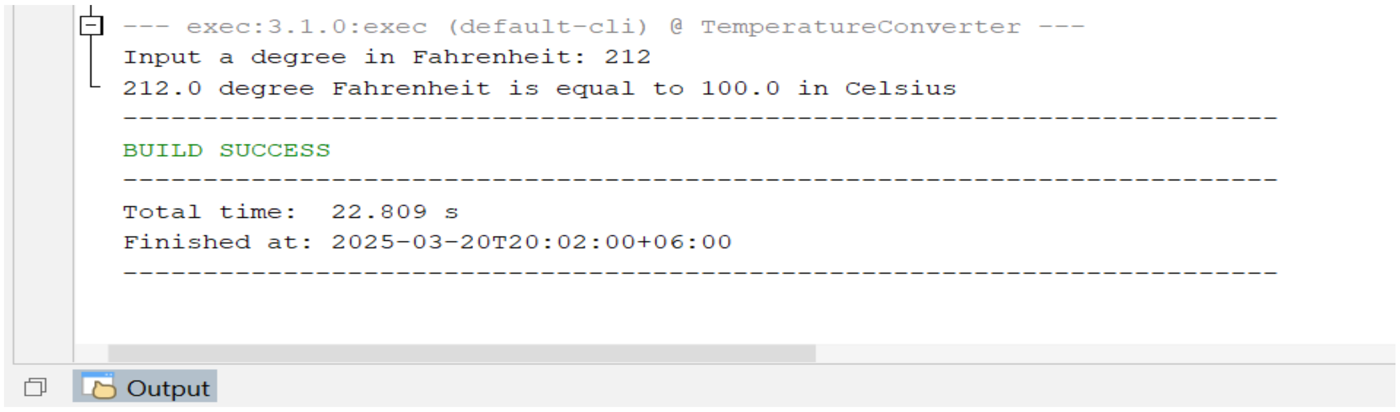
Ans:

Code:



```java

package com.muntaha.temperatureconverter;

import java.util.Scanner;

public class TemperatureConverter {
    public static void main(String[] args) {
        // Taking user input
        try (Scanner scanner = new Scanner(System.in)) {
            // Taking user input
            System.out.print("Input a degree in Fahrenheit: ");
            double fahrenheit = scanner.nextDouble();

            // Conversion formula
            double celsius = (fahrenheit - 32) * 5 / 9;

            // Displaying the result
            System.out.println(fahrenheit + " degree Fahrenheit is equal to " + celsius + " in Celsius");
        }
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ TemperatureConverter ---
Input a degree in Fahrenheit: 212
212.0 degree Fahrenheit is equal to 100.0 in Celsius
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  22.809 s
Finished at: 2025-03-20T20:02:00+06:00
------------------------------------------------------------------------
```

Explanation:

The program prompts the user to input a temperature in Fahrenheit. It applies the formula: C = (F - 32) × 5/9 to convert Fahrenheit to Celsius.

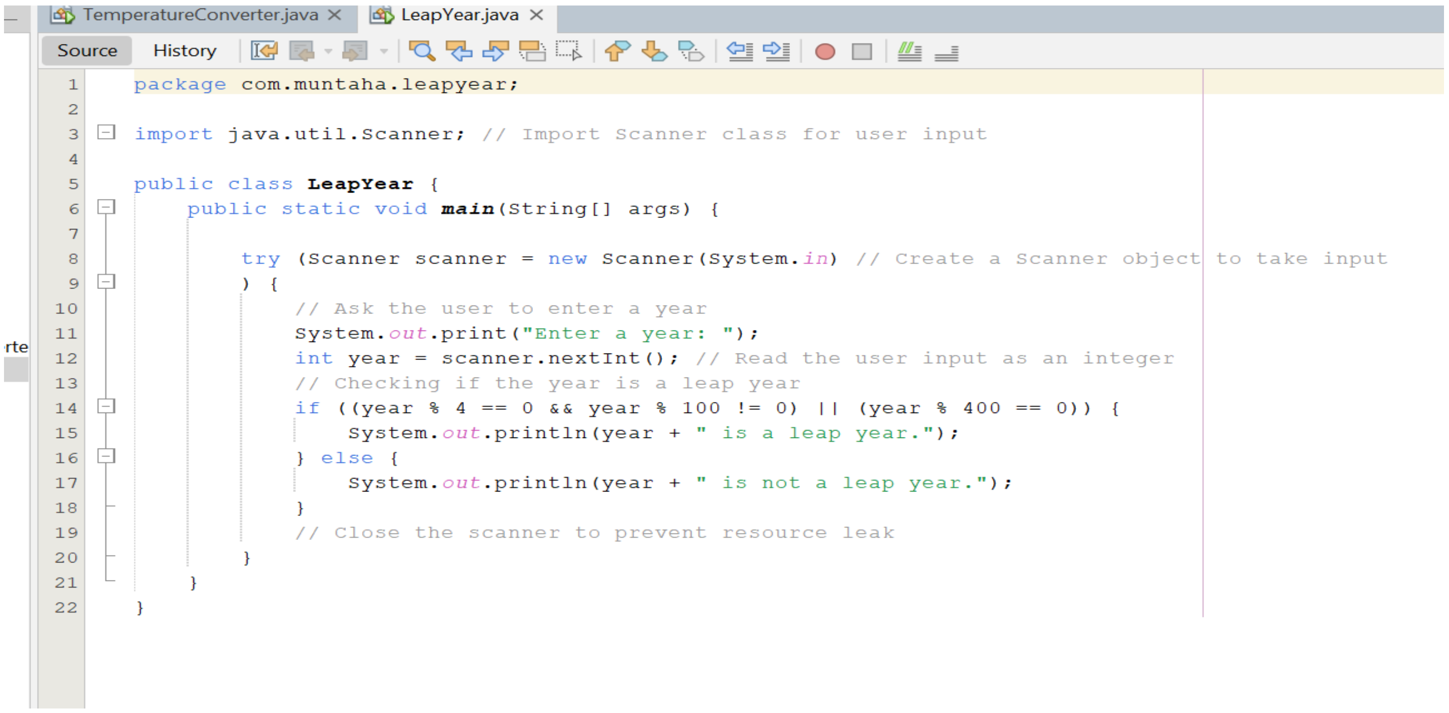
Finally, it prints the result. This program will work for any Fahrenheit value input by the user.

b. SOLVE THESE PROBLEMS USING JAVA:

1. Write a program to test a year if it is leap year or not.

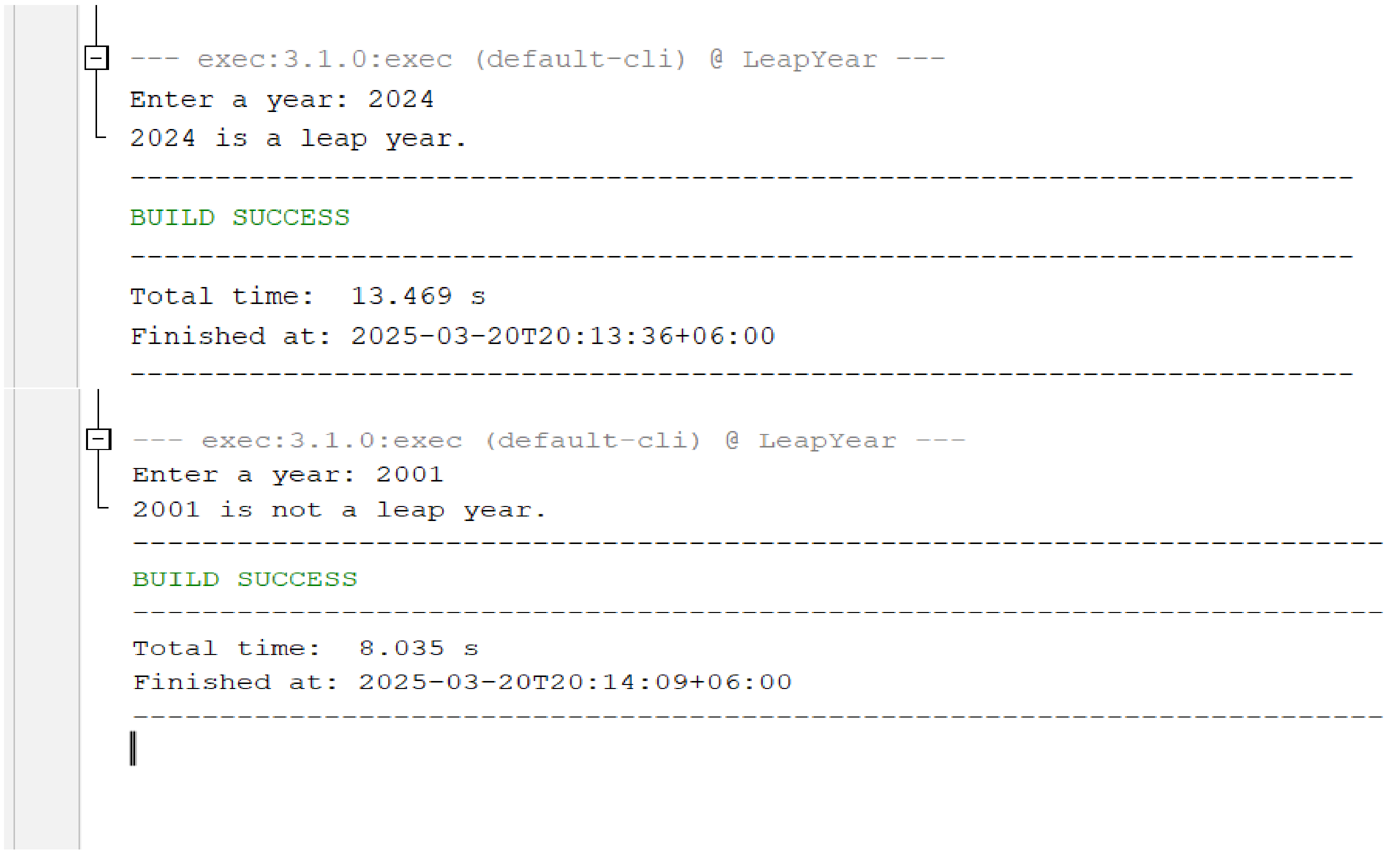
Ans:

Code:



```java
package com.muntaha.leapyear;

import java.util.Scanner; // Import Scanner class for user input

public class LeapYear {
    public static void main(String[] args) {

        try (Scanner scanner = new Scanner(System.in) // Create a Scanner object to take input
        ) {
            // Ask the user to enter a year
            System.out.print("Enter a year: ");
            int year = scanner.nextInt(); // Read the user input as an integer
            // Checking if the year is a leap year
            if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
                System.out.println(year + " is a leap year.");
            } else {
                System.out.println(year + " is not a leap year.");
            }
            // Close the scanner to prevent resource leak
        }
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ LeapYear ---
Enter a year: 2024
2024 is a leap year.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  13.469 s
Finished at: 2025-03-20T20:13:36+06:00
------------------------------------------------------------------------

--- exec:3.1.0:exec (default-cli) @ LeapYear ---
Enter a year: 2001
2001 is not a leap year.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  8.035 s
Finished at: 2025-03-20T20:14:09+06:00
------------------------------------------------------------------------
```

Explanation:

The program asks the user to input a year. It checks if the year satisfies the leap year conditions. If true, it prints that the year is a leap year; otherwise, it prints that it is not a leap year.

A year is considered a leap year if it satisfies the following conditions:

1. It is divisible by 4 → (year % 4 == 0)

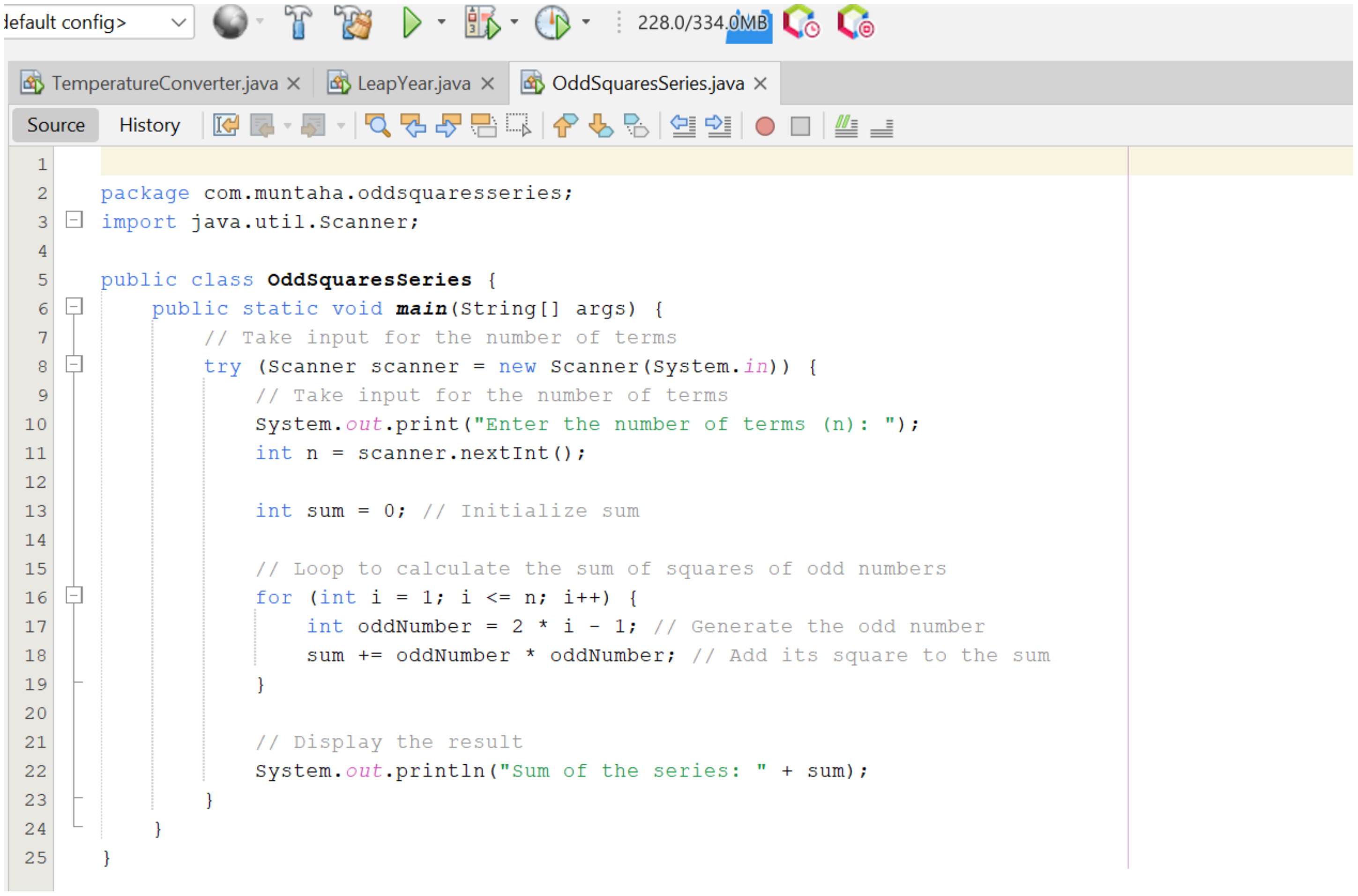
2. BUT, if it is also divisible by 100, it must be divisible by 400 → (year % 100 != 0 || year % 400 == 0)

This ensures that century years (like 1700, 1800, 1900) are not leap years unless they are divisible by 400 (like 1600, 2000).

2. Write a program to evaluate the following series $1^2+3^2+5^2+$.......................... Up to n terms.

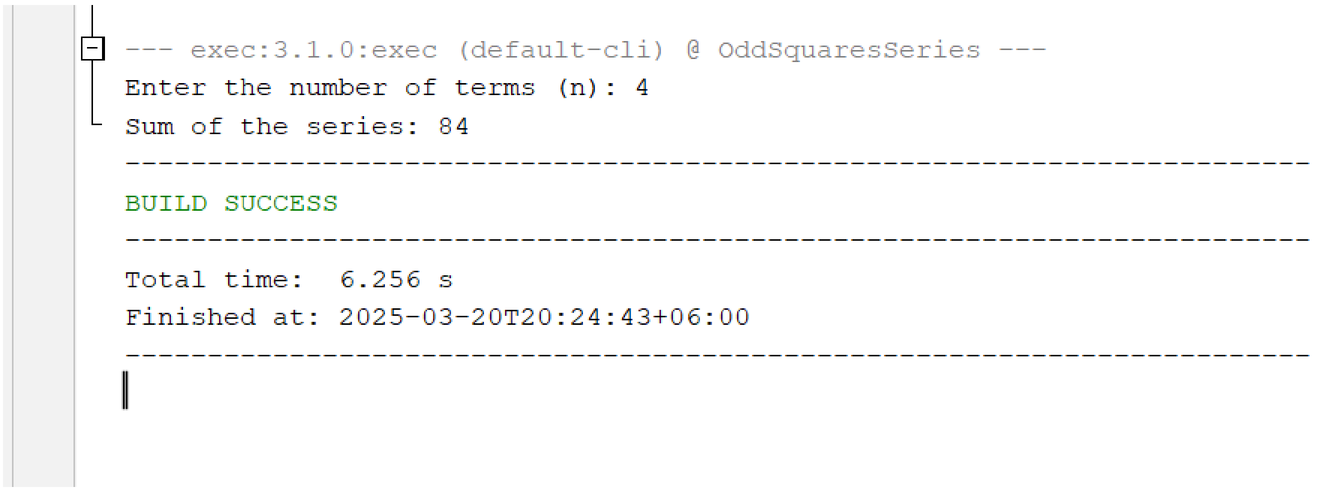
Ans:

Code:



```

package com.muntaha.oddsquaresseries;
import java.util.Scanner;

public class OddSquaresSeries {
    public static void main(String[] args) {
        // Take input for the number of terms
        try (Scanner scanner = new Scanner(System.in)) {
            // Take input for the number of terms
            System.out.print("Enter the number of terms (n): ");
            int n = scanner.nextInt();

            int sum = 0; // Initialize sum

            // Loop to calculate the sum of squares of odd numbers
            for (int i = 1; i <= n; i++) {
                int oddNumber = 2 * i - 1; // Generate the odd number
                sum += oddNumber * oddNumber; // Add its square to the sum
            }

            // Display the result
            System.out.println("Sum of the series: " + sum);
        }
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ OddSquaresSeries ---
Enter the number of terms (n): 4
Sum of the series: 84
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  6.256 s
Finished at: 2025-03-20T20:24:43+06:00
------------------------------------------------------------------------
```

Explanation:

The series consists of squares of the first odd numbers.The program calculates the sum iteratively using a loop. For n = 4:

$1^2 + 3^2 + 5^2 + 7^2 = 1 + 9 + 25 + 49 = 84$

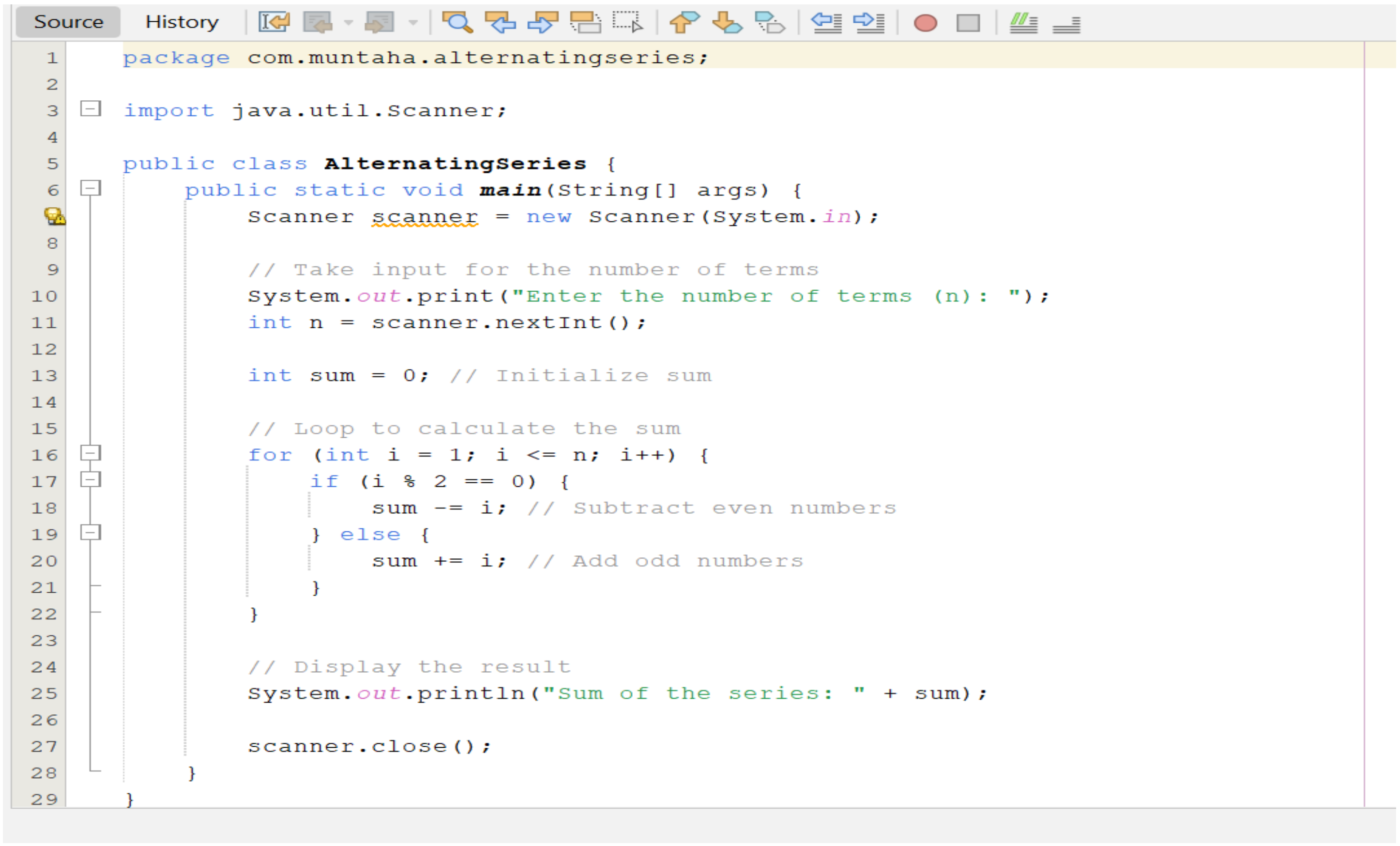
This program efficiently computes the sum by iterating through the first odd numbers, squaring each, and accumulating the sum.

3. Write a program to evaluate the following series

1-2+3-4+........................... Up to n terms.

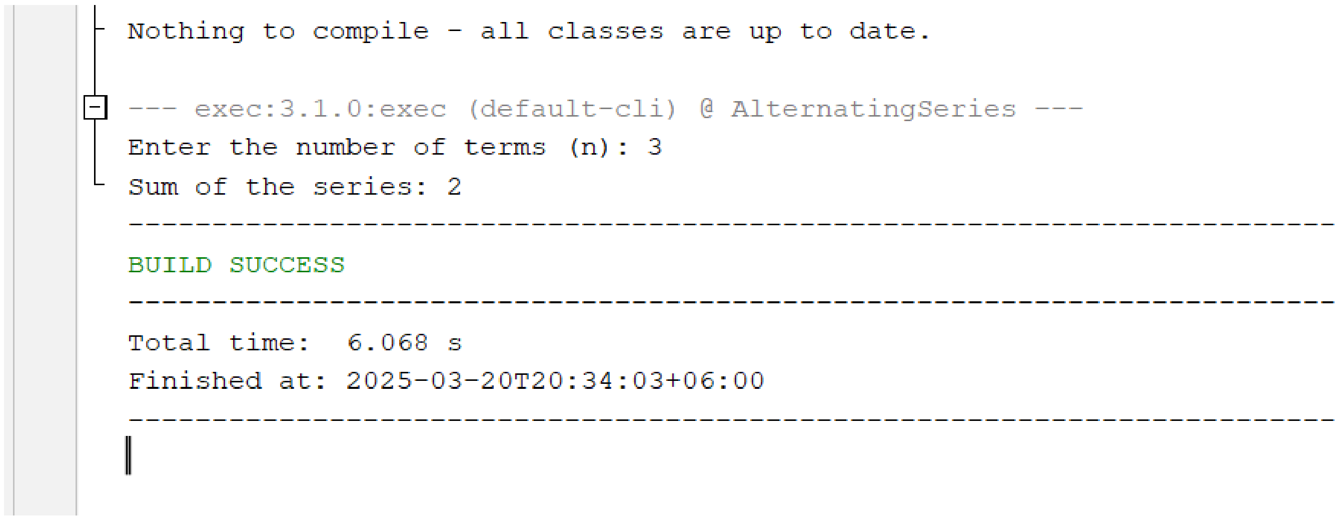
Ans:

Code:



```java
package com.muntaha.alternatingseries;

import java.util.Scanner;

public class AlternatingSeries {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Take input for the number of terms
        System.out.print("Enter the number of terms (n): ");
        int n = scanner.nextInt();

        int sum = 0; // Initialize sum

        // Loop to calculate the sum
        for (int i = 1; i <= n; i++) {
            if (i % 2 == 0) {
                sum -= i; // Subtract even numbers
            } else {
                sum += i; // Add odd numbers
            }
        }

        // Display the result
        System.out.println("Sum of the series: " + sum);

        scanner.close();
    }
}
```

Output:



```
Nothing to compile - all classes are up to date.

--- exec:3.1.0:exec (default-cli) @ AlternatingSeries ---
Enter the number of terms (n): 3
Sum of the series: 2
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  6.068 s
Finished at: 2025-03-20T20:34:03+06:00
------------------------------------------------------------------------
```

Explanation:

Odd-indexed terms are positive. Even-indexed terms are negative.

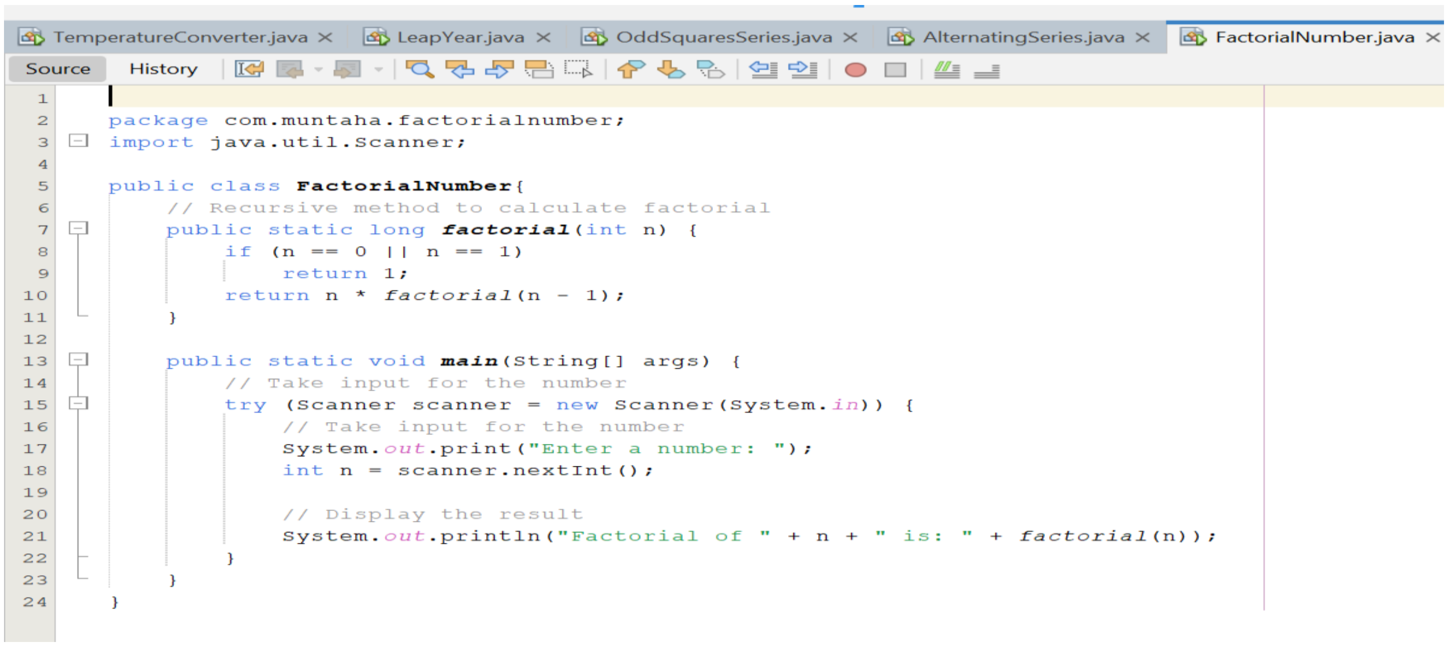
Explanation for n = 3: 1 - 2 + 3 = 2

This program efficiently computes the sum by checking whether the index is odd or even and adjusting the sign accordingly.

4. Write a program to find the factorial of a number.

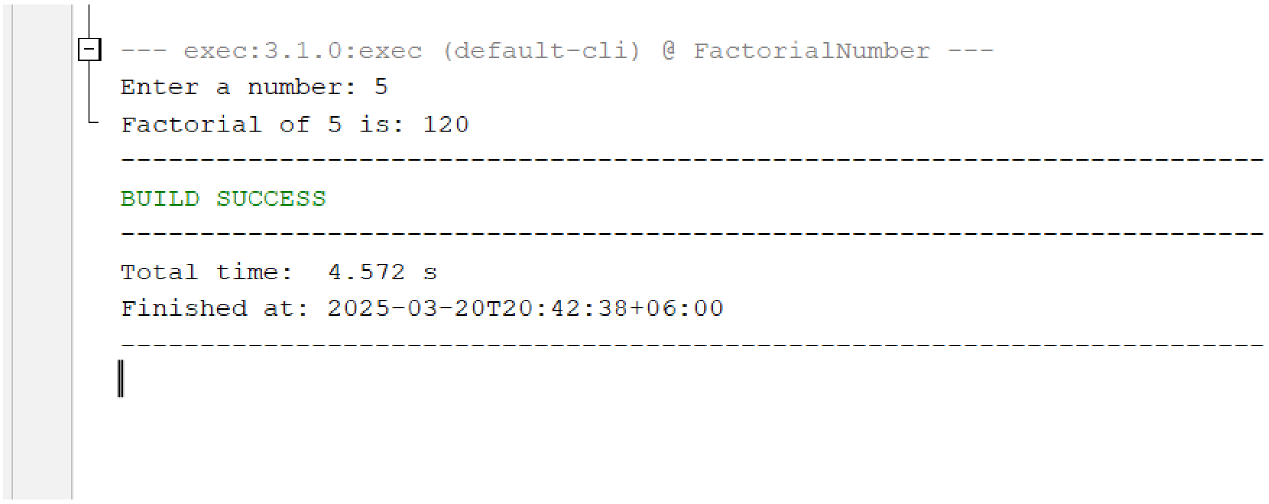
Ans:

Code:



```java

package com.muntaha.factorialnumber;
import java.util.Scanner;

public class FactorialNumber{
    // Recursive method to calculate factorial
    public static long factorial(int n) {
        if (n == 0 || n == 1)
            return 1;
        return n * factorial(n - 1);
    }

    public static void main(String[] args) {
        // Take input for the number
        try (Scanner scanner = new Scanner(System.in)) {
            // Take input for the number
            System.out.print("Enter a number: ");
            int n = scanner.nextInt();

            // Display the result
            System.out.println("Factorial of " + n + " is: " + factorial(n));
        }
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ FactorialNumber ---
Enter a number: 5
Factorial of 5 is: 120
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  4.572 s
Finished at: 2025-03-20T20:42:38+06:00
------------------------------------------------------------------------
```

Explanation:

Factorial Formula:

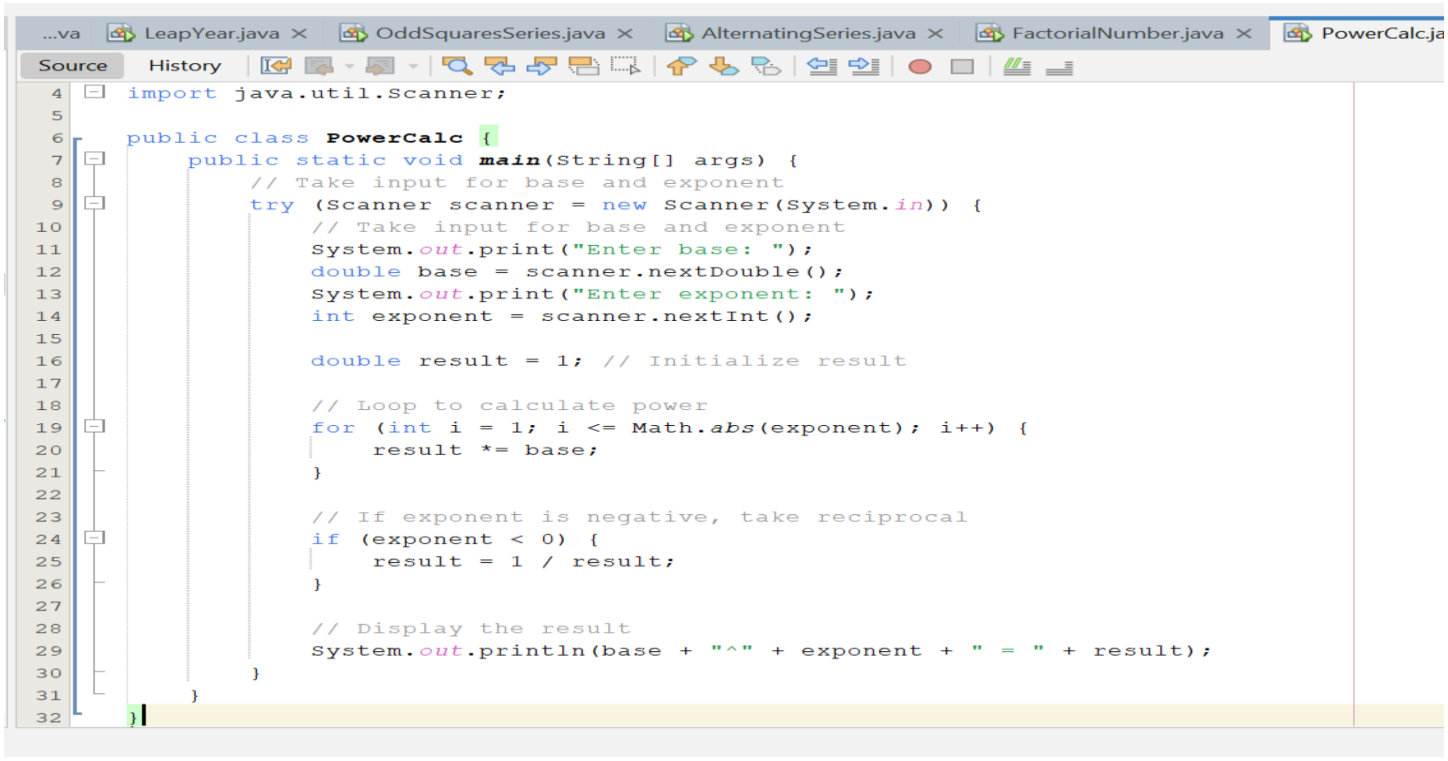
n! = n * (n - 1) *(n - 2)*………*1

5!= 5*4*3*2*1=120

5. Write a program to find the power for a given base and exponent.

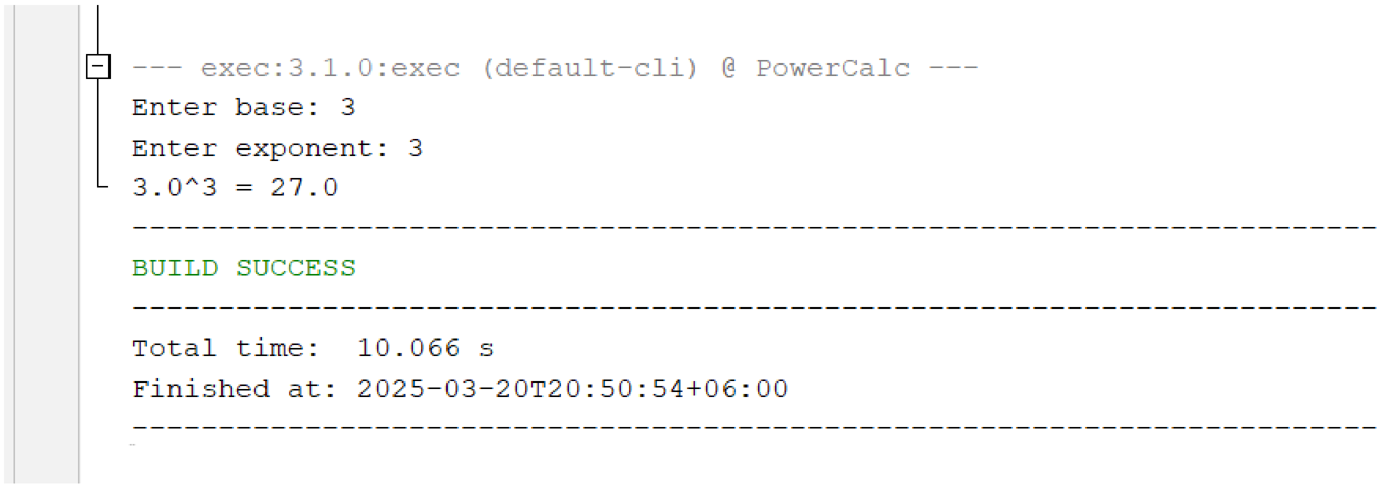
Ans:

Code:



```java
import java.util.Scanner;

public class PowerCalc {
    public static void main(String[] args) {
        // Take input for base and exponent
        try (Scanner scanner = new Scanner(System.in)) {
            // Take input for base and exponent
            System.out.print("Enter base: ");
            double base = scanner.nextDouble();
            System.out.print("Enter exponent: ");
            int exponent = scanner.nextInt();

            double result = 1; // Initialize result

            // Loop to calculate power
            for (int i = 1; i <= Math.abs(exponent); i++) {
                result *= base;
            }

            // If exponent is negative, take reciprocal
            if (exponent < 0) {
                result = 1 / result;
            }

            // Display the result
            System.out.println(base + "^" + exponent + " = " + result);
        }
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ PowerCalc ---
Enter base: 3
Enter exponent: 3
3.0^3 = 27.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  10.066 s
Finished at: 2025-03-20T20:50:54+06:00
------------------------------------------------------------------------
```

Explanation:

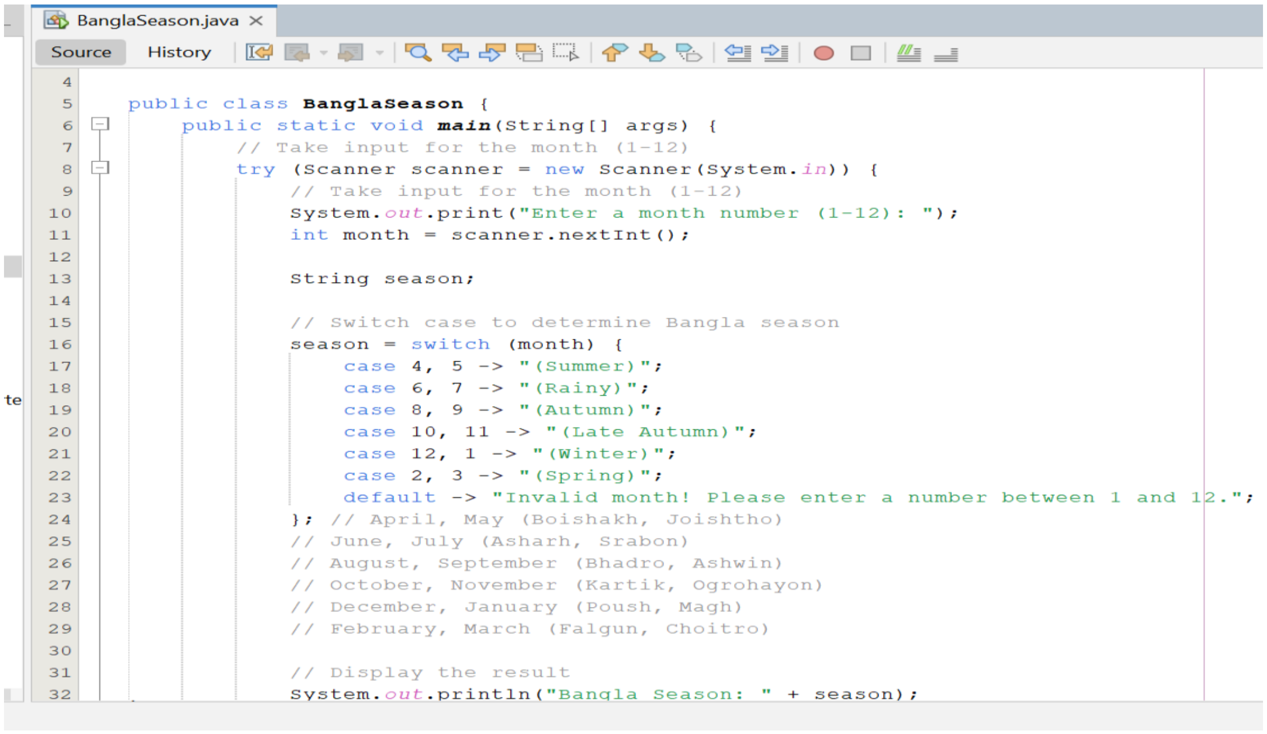
Result= $\text{base}^{\text{exponent}}$

Hence, $3^3 = 27$

6. Write a program to find the Bangla season form a given month using if/switch.

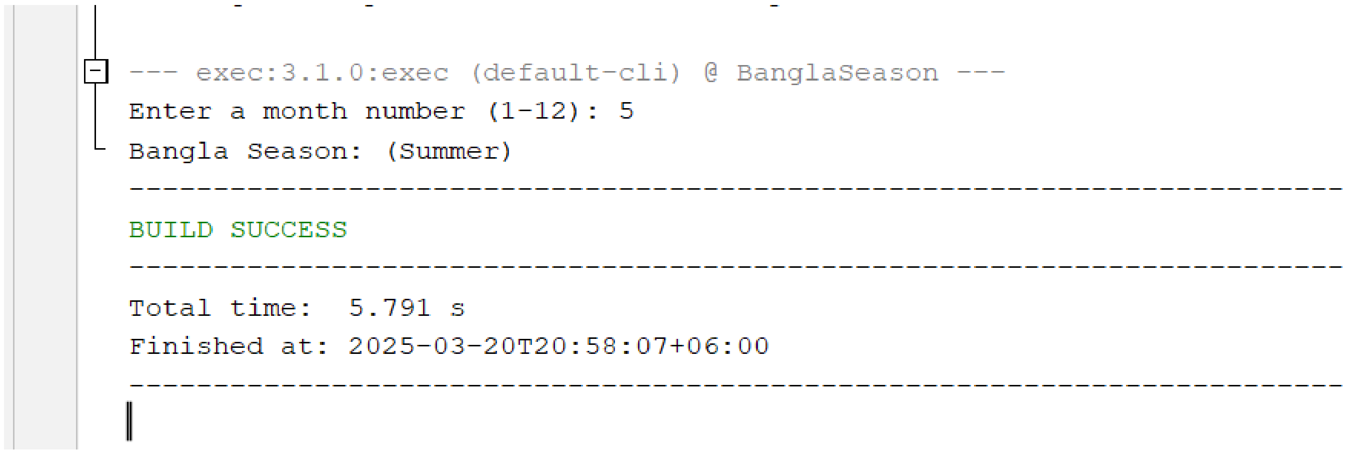
Ans: Using switch

Code:



```java

public class BanglaSeason {
    public static void main(String[] args) {
        // Take input for the month (1-12)
        try (Scanner scanner = new Scanner(System.in)) {
            // Take input for the month (1-12)
            System.out.print("Enter a month number (1-12): ");
            int month = scanner.nextInt();

            String season;

            // Switch case to determine Bangla season
            season = switch (month) {
                case 4, 5 -> "(Summer)";
                case 6, 7 -> "(Rainy)";
                case 8, 9 -> "(Autumn)";
                case 10, 11 -> "(Late Autumn)";
                case 12, 1 -> "(Winter)";
                case 2, 3 -> "(Spring)";
                default -> "Invalid month! Please enter a number between 1 and 12.";
            }; // April, May (Boishakh, Joishtho)
            // June, July (Asharh, Srabon)
            // August, September (Bhadro, Ashwin)
            // October, November (Kartik, Ogrohayon)
            // December, January (Poush, Magh)
            // February, March (Falgun, Choitro)

            // Display the result
            System.out.println("Bangla Season: " + season);
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ BanglaSeason ---
Enter a month number (1-12): 5
Bangla Season: (Summer)
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  5.791 s
Finished at: 2025-03-20T20:58:07+06:00
------------------------------------------------------------------------
```

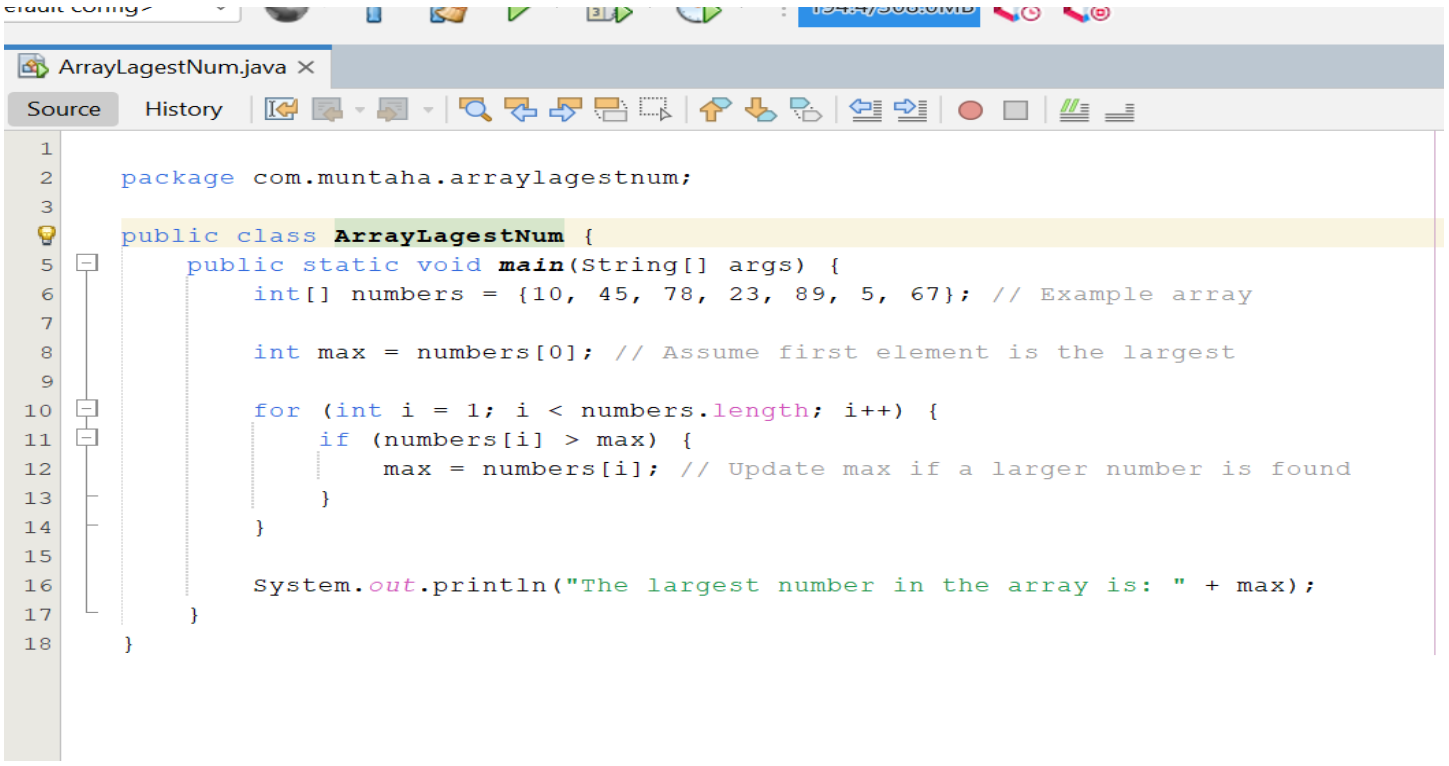
Explanation:

This program takes the **Gregorian month** number (1-12) as input and determines the corresponding **Bangla season** using a switch statement.

7. Write a program to find the largest number in a list of Array.

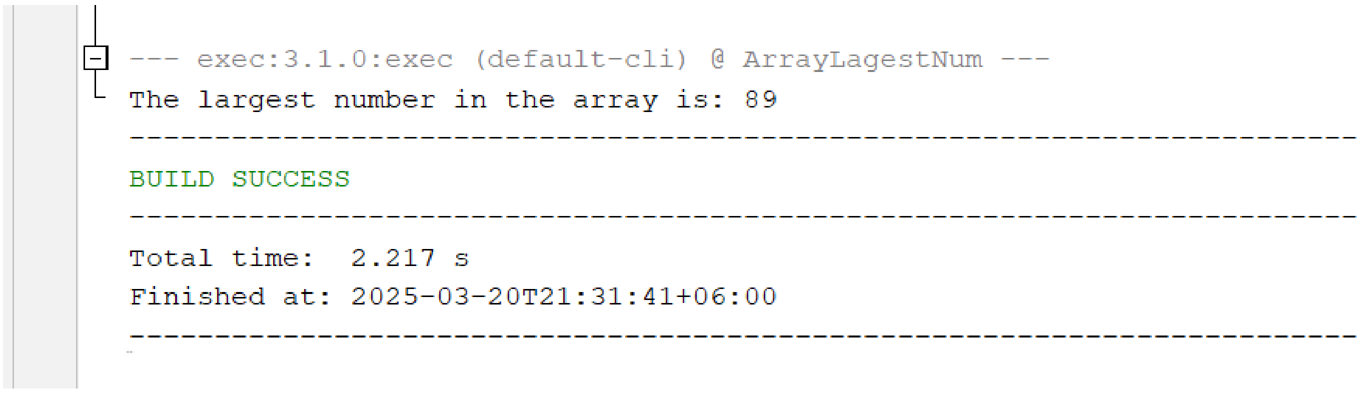
Ans:

Code:



```java

package com.muntaha.arraylagestnum;

public class ArrayLagestNum {
    public static void main(String[] args) {
        int[] numbers = {10, 45, 78, 23, 89, 5, 67}; // Example array

        int max = numbers[0]; // Assume first element is the largest

        for (int i = 1; i < numbers.length; i++) {
            if (numbers[i] > max) {
                max = numbers[i]; // Update max if a larger number is found
            }
        }

        System.out.println("The largest number in the array is: " + max);
    }
}
```

Output:



```
--- exec:3.1.0:exec (default-cli) @ ArrayLagestNum ---
The largest number in the array is: 89
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.217 s
Finished at: 2025-03-20T21:31:41+06:00
------------------------------------------------------------------------
```

Explanation:

1. We initialize max with the first element of the array.

2. We iterate through the array starting from index 1 and compare each element with max.

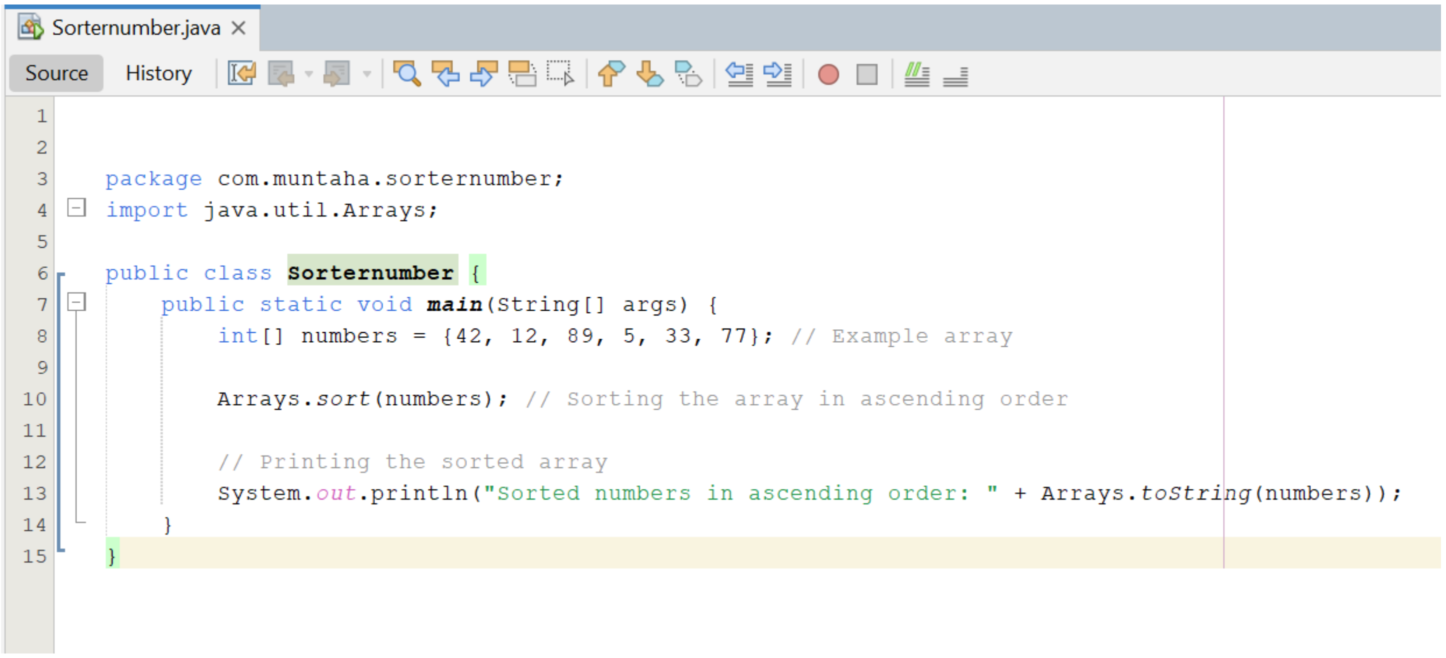
3. If a larger number is found, we update max.

4. Finally, we print the largest number.

8. Write a program to sort some number in ascending order.

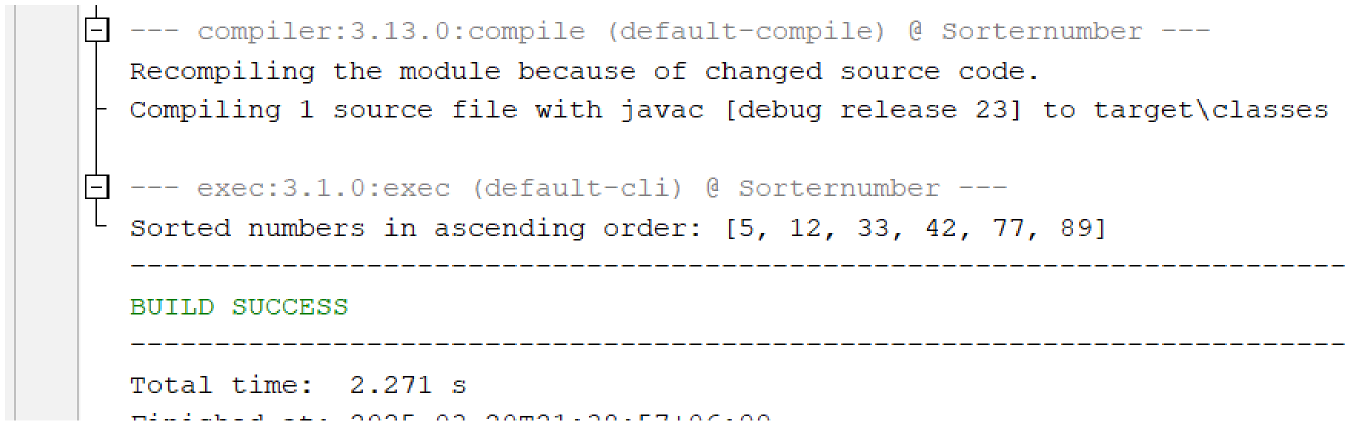
Ans:

Code:



```java


package com.muntaha.sortenumber;
import java.util.Arrays;

public class Sortenumber {
    public static void main(String[] args) {
        int[] numbers = {42, 12, 89, 5, 33, 77}; // Example array

        Arrays.sort(numbers); // Sorting the array in ascending order

        // Printing the sorted array
        System.out.println("Sorted numbers in ascending order: " + Arrays.toString(numbers));
    }
}
```

Output:



```
--- compiler:3.13.0:compile (default-compile) @ Sorternumber ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 23] to target\classes

--- exec:3.1.0:exec (default-cli) @ Sorternumber ---
Sorted numbers in ascending order: [5, 12, 33, 42, 77, 89]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.271 s
Finished at: 2025-03-20T21:38:57+06:00
```

Explanation:

1. We define an array of numbers.
2. We use Arrays.sort() to sort the array in ascending order.
3. Finally, we print the sorted array using **Arrays.toString().**