LAB # 03

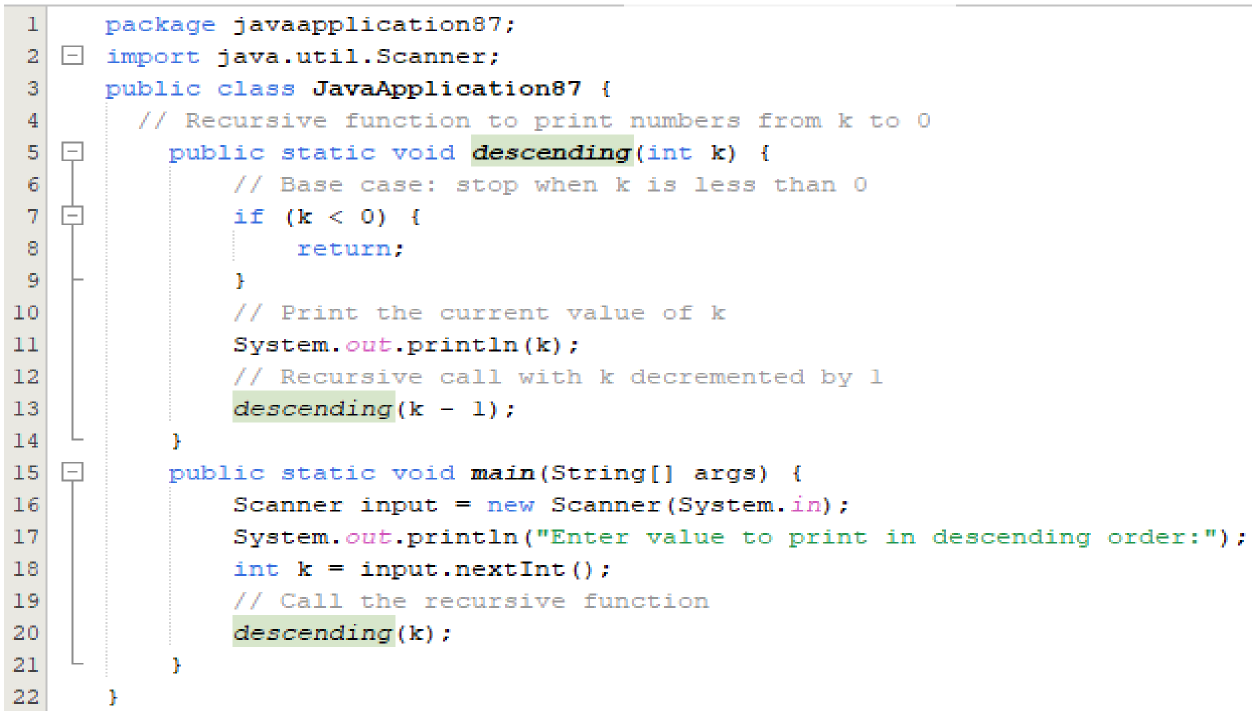
RECURSION

OBJECTIVE: To understand the complexities of the recursive functions and a way to reduce these complexities.

LAB TASK

1. Write a program which takes an integer value (k) as input and prints the sequence of numbers from k to 0 in descending order.

CODE:



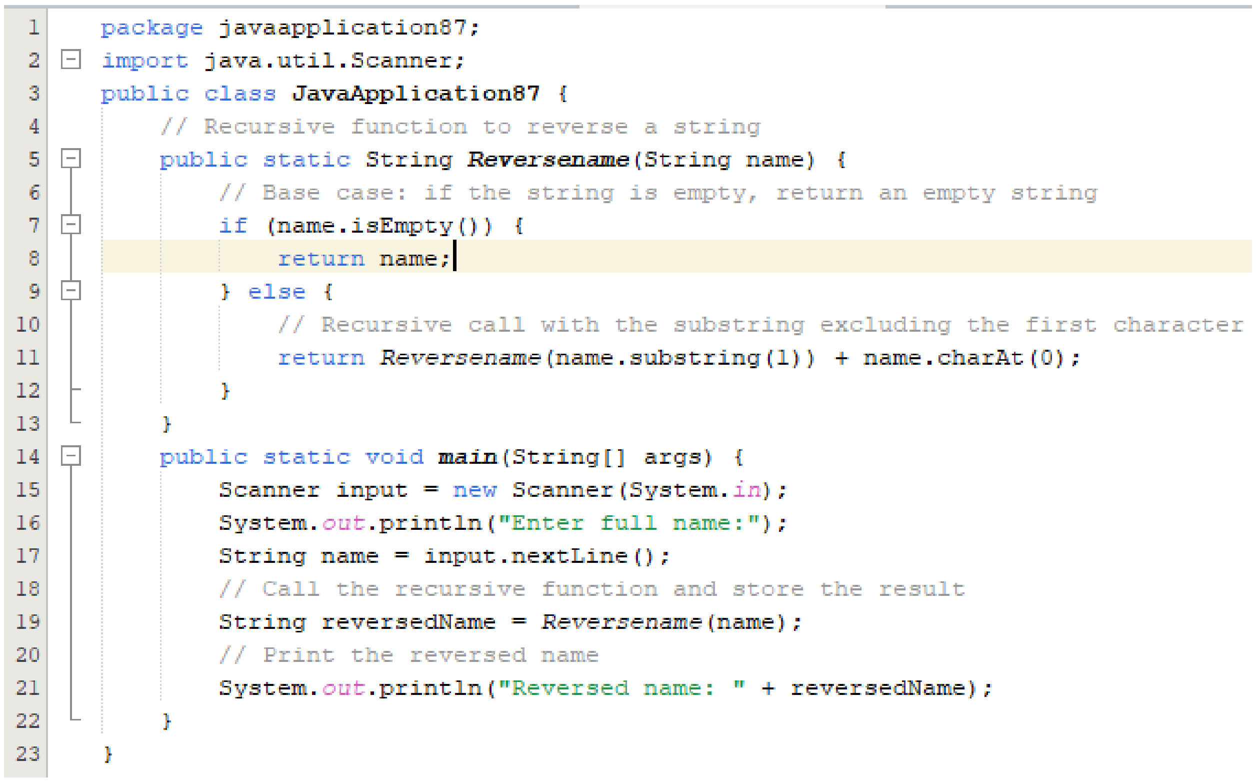
```java
package javaapplication87;
import java.util.Scanner;
public class JavaApplication87 {
  // Recursive function to print numbers from k to 0
    public static void descending(int k) {
        // Base case: stop when k is less than 0
        if (k < 0) {
            return;
        }
        // Print the current value of k
        System.out.println(k);
        // Recursive call with k decremented by 1
        descending(k - 1);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter value to print in descending order:");
        int k = input.nextInt();
        // Call the recursive function
        descending(k);
    }
}
```

OUTPUT:

```
run:
Enter value to print in descending order:
12
12
11
10
9
8
7
6
5
4
3
2
1
0
BUILD SUCCESSFUL (total time: 7 seconds)
```

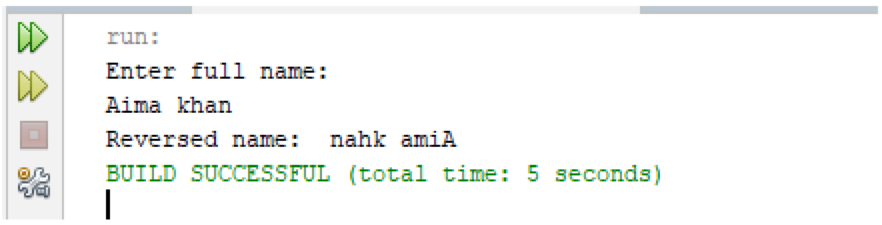
2. Write a program to reverse your full name using Recursion.

CODE:



```java
package javaapplication87;
import java.util.Scanner;
public class JavaApplication87 {
    // Recursive function to reverse a string
    public static String Reversename(String name) {
        // Base case: if the string is empty, return an empty string
        if (name.isEmpty()) {
            return name;
        } else {
            // Recursive call with the substring excluding the first character
            return Reversename(name.substring(1)) + name.charAt(0);
        }
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter full name:");
        String name = input.nextLine();
        // Call the recursive function and store the result
        String reversedName = Reversename(name);
        // Print the reversed name
        System.out.println("Reversed name: " + reversedName);
    }
}
```

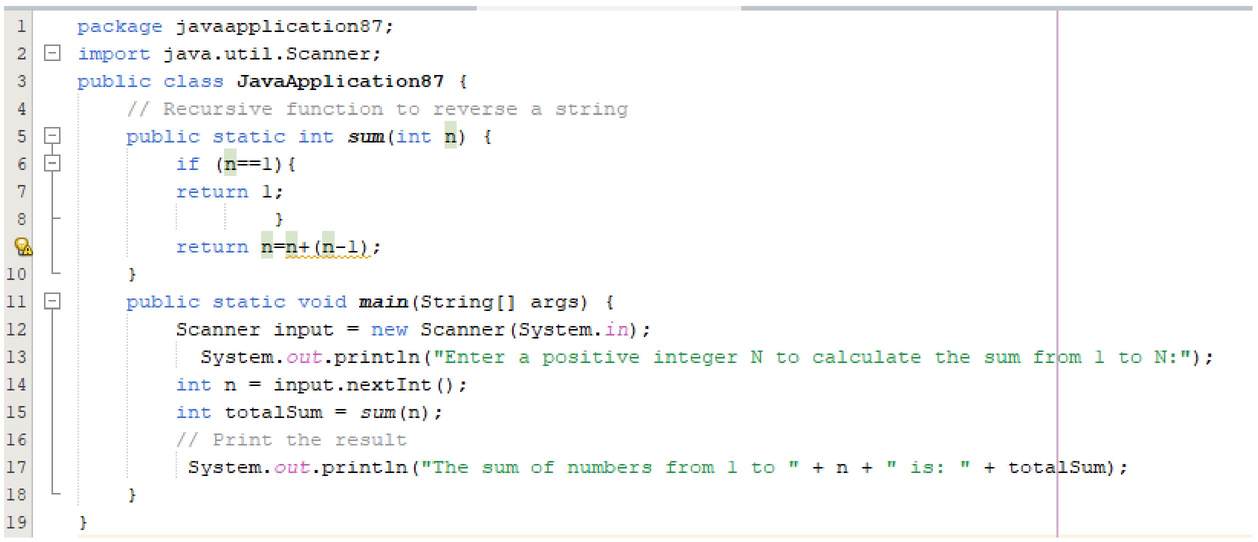
OUTPUT:



```
run:
Enter full name:
Aima khan
Reversed name:  nahk amiA
BUILD SUCCESSFUL (total time: 5 seconds)
```

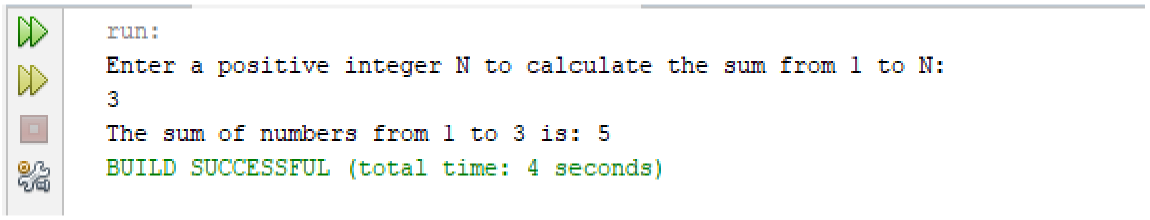
3. Write a program to calculate the sum of numbers from 1 to N using recursion. N should be user input.

CODE:



```java
package javaapplication87;
import java.util.Scanner;
public class JavaApplication87 {
    // Recursive function to reverse a string
    public static int sum(int n) {
        if (n==1){
        return 1;
            }
        return n=n+(n-1);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
          System.out.println("Enter a positive integer N to calculate the sum from 1 to N:");
        int n = input.nextInt();
        int totalSum = sum(n);
        // Print the result
         System.out.println("The sum of numbers from 1 to " + n + " is: " + totalSum);
    }
}
```

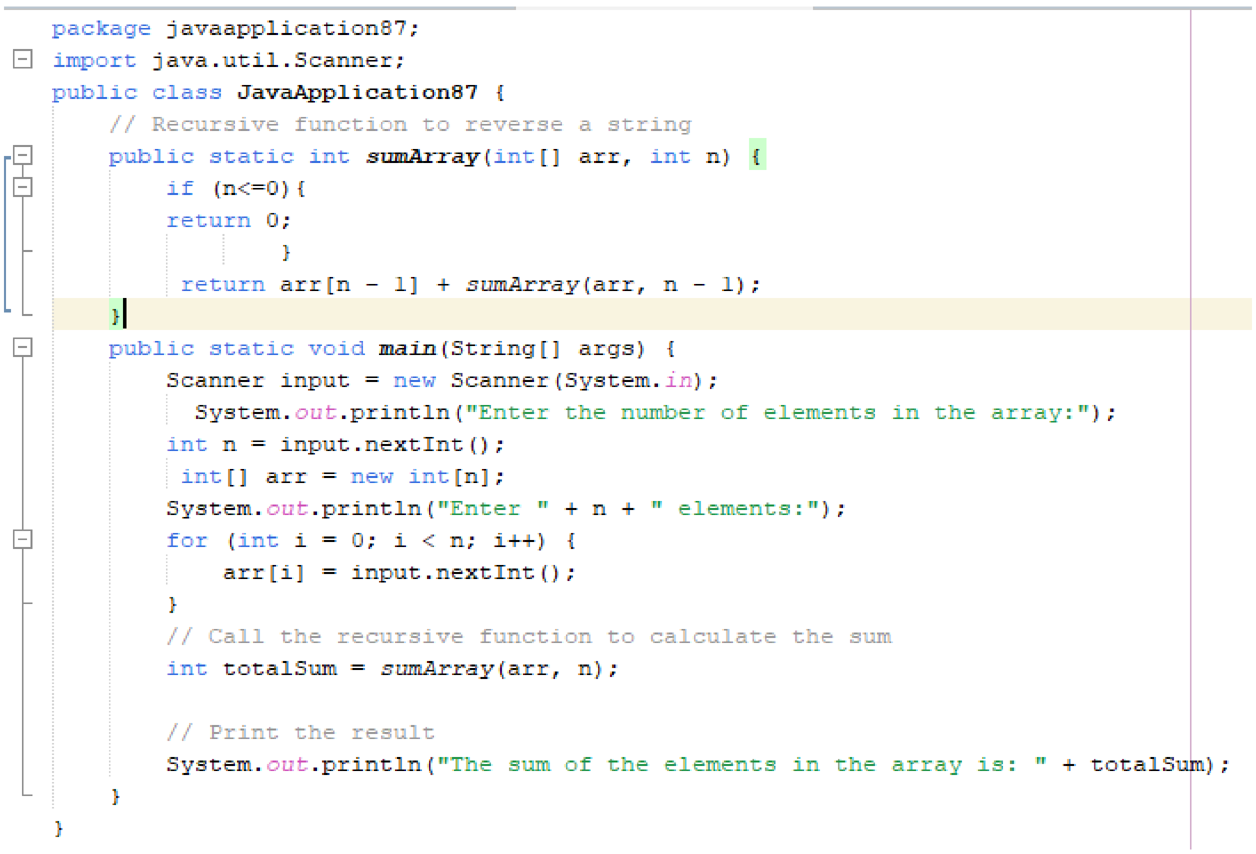
OUTPUT:



```
run:
Enter a positive integer N to calculate the sum from 1 to N:
3
The sum of numbers from 1 to 3 is: 5
BUILD SUCCESSFUL (total time: 4 seconds)
```

4. Write a recursive program to calculate the sum of elements in an array

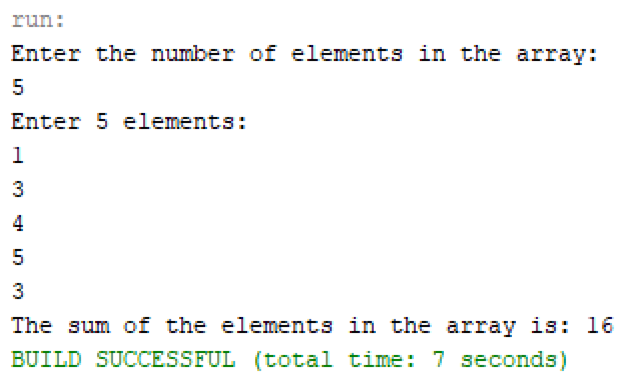
CODE:



```java
package javaapplication87;
import java.util.Scanner;
public class JavaApplication87 {
    // Recursive function to reverse a string
    public static int sumArray(int[] arr, int n) {
        if (n<=0){
        return 0;
                }
         return arr[n - 1] + sumArray(arr, n - 1);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
          System.out.println("Enter the number of elements in the array:");
        int n = input.nextInt();
         int[] arr = new int[n];
        System.out.println("Enter " + n + " elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = input.nextInt();
        }
        // Call the recursive function to calculate the sum
        int totalSum = sumArray(arr, n);

        // Print the result
        System.out.println("The sum of the elements in the array is: " + totalSum);
    }
}
```

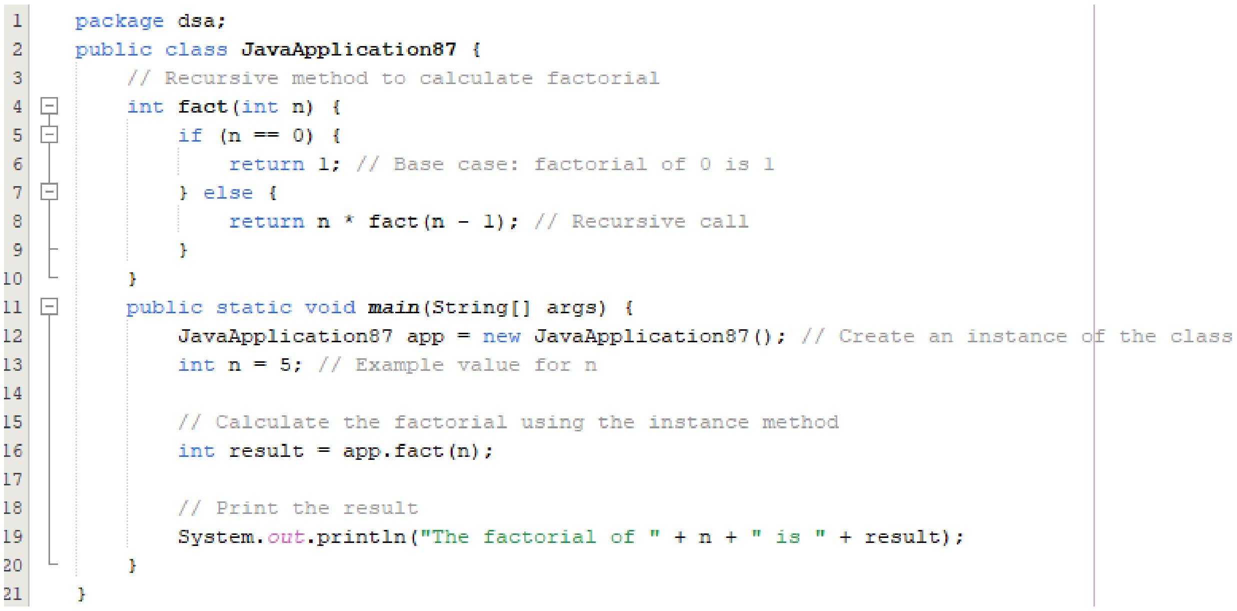
OUTPUT:



```
run:
Enter the number of elements in the array:
5
Enter 5 elements:
1
3
4
5
3
The sum of the elements in the array is: 16
BUILD SUCCESSFUL (total time: 7 seconds)
```

5. Write a recursive program to calculate the factorial of a given integer n

CODE:

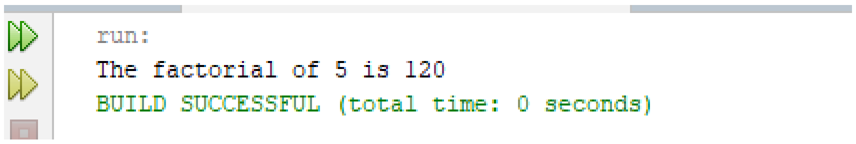


```java
package dsa;
public class JavaApplication87 {
    // Recursive method to calculate factorial
    int fact(int n) {
        if (n == 0) {
            return 1; // Base case: factorial of 0 is 1
        } else {
            return n * fact(n - 1); // Recursive call
        }
    }
    public static void main(String[] args) {
        JavaApplication87 app = new JavaApplication87(); // Create an instance of the class
        int n = 5; // Example value for n

        // Calculate the factorial using the instance method
        int result = app.fact(n);

        // Print the result
        System.out.println("The factorial of " + n + " is " + result);
    }
}
```

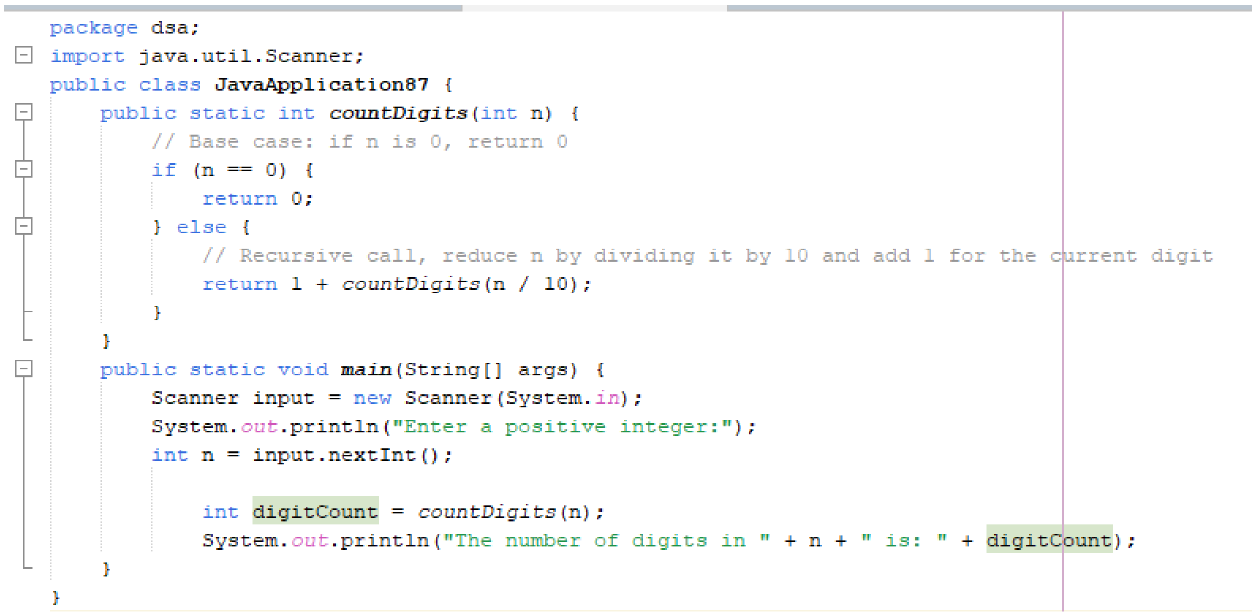
OUTPUT:



```
run:
The factorial of 5 is 120
BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Write a program to count the digits of a given number using recursion.

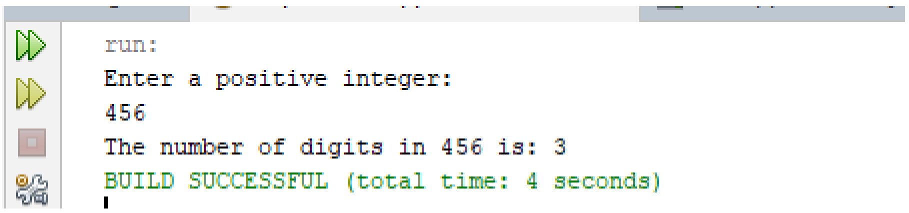
CODE:



```java
package dsa;
import java.util.Scanner;
public class JavaApplication87 {
    public static int countDigits(int n) {
        // Base case: if n is 0, return 0
        if (n == 0) {
            return 0;
        } else {
            // Recursive call, reduce n by dividing it by 10 and add 1 for the current digit
            return 1 + countDigits(n / 10);
        }
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a positive integer:");
        int n = input.nextInt();

            int digitCount = countDigits(n);
            System.out.println("The number of digits in " + n + " is: " + digitCount);
    }
}
```

OUTPUT:

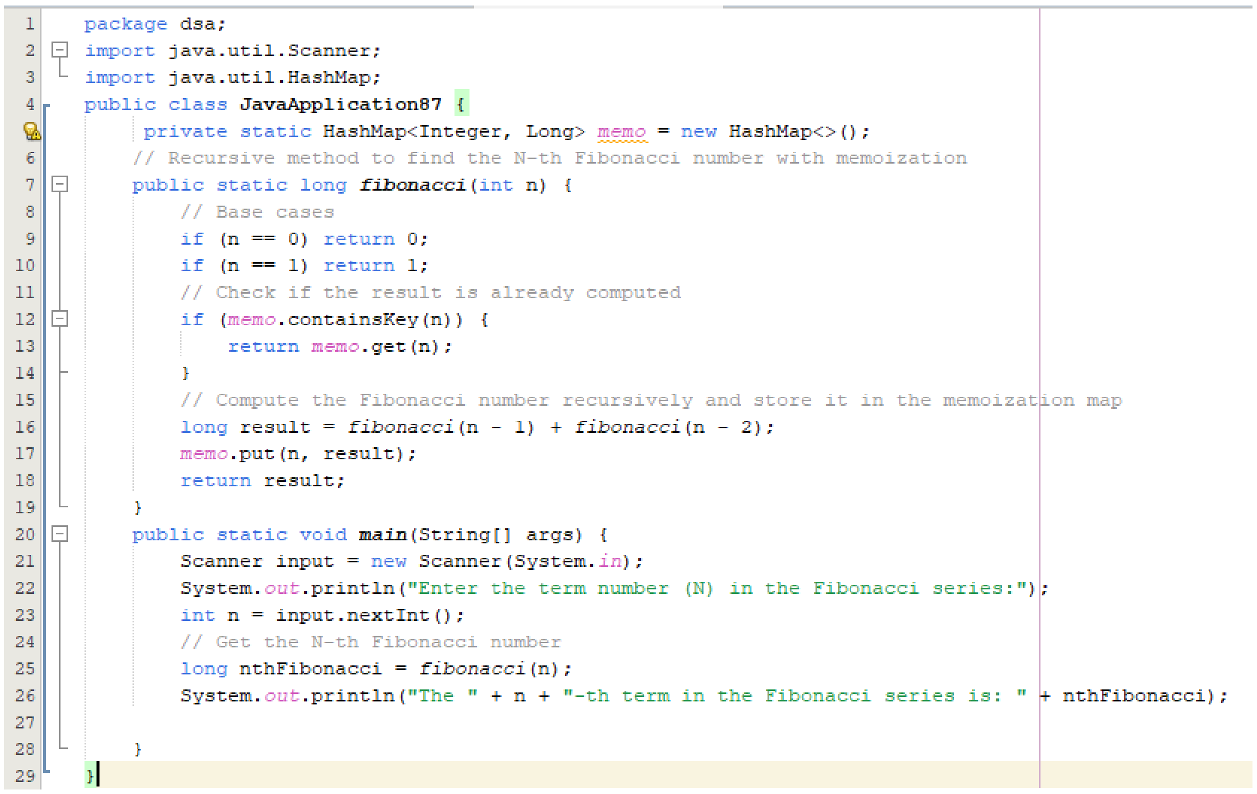


```
run:
Enter a positive integer:
456
The number of digits in 456 is: 3
BUILD SUCCESSFUL (total time: 4 seconds)
```

HOME TASK

1. Write a java program to find the N-th term in the Fibonacci series using Memoization.

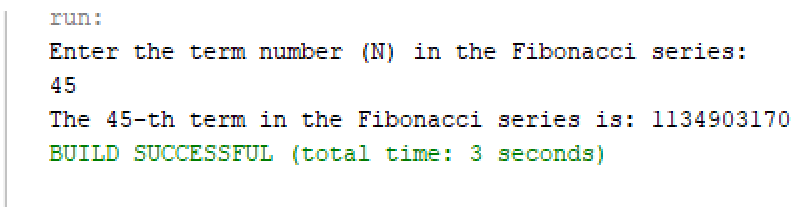
CODE:



```java
package dsa;
import java.util.Scanner;
import java.util.HashMap;
public class JavaApplication87 {
     private static HashMap<Integer, Long> memo = new HashMap<>();
    // Recursive method to find the N-th Fibonacci number with memoization
    public static long fibonacci(int n) {
        // Base cases
        if (n == 0) return 0;
        if (n == 1) return 1;
        // Check if the result is already computed
        if (memo.containsKey(n)) {
            return memo.get(n);
        }
        // Compute the Fibonacci number recursively and store it in the memoization map
        long result = fibonacci(n - 1) + fibonacci(n - 2);
        memo.put(n, result);
        return result;
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter the term number (N) in the Fibonacci series:");
        int n = input.nextInt();
        // Get the N-th Fibonacci number
        long nthFibonacci = fibonacci(n);
        System.out.println("The " + n + "-th term in the Fibonacci series is: " + nthFibonacci);

    }
}
```

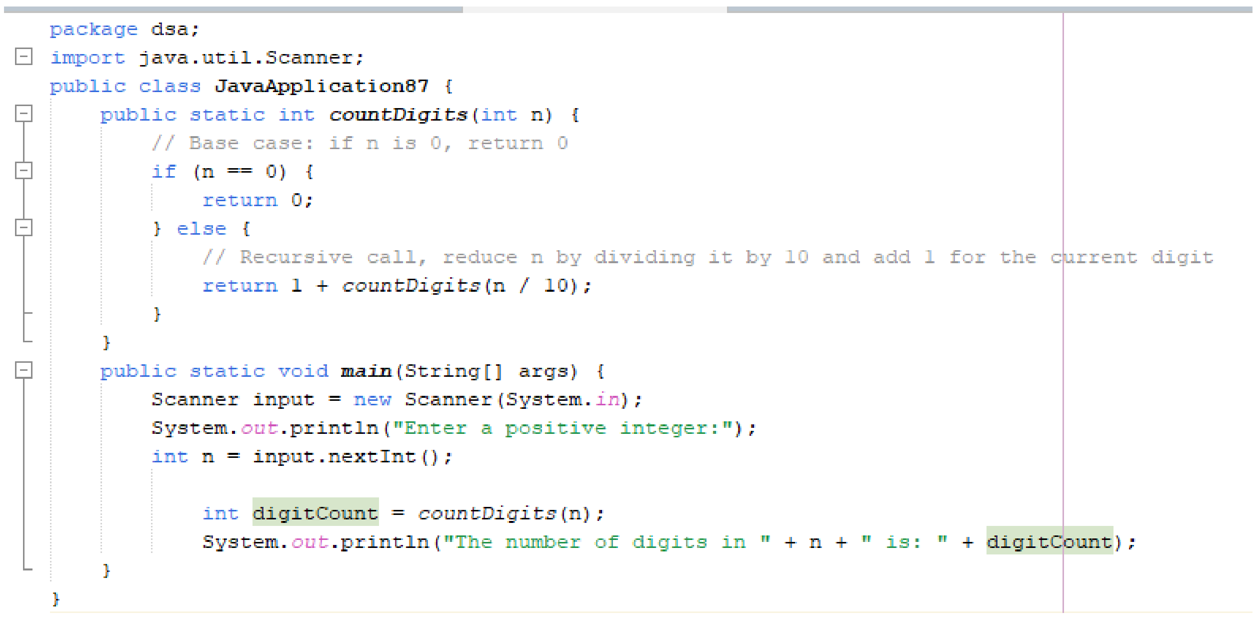
OUTPUT:



```
run:
Enter the term number (N) in the Fibonacci series:
45
The 45-th term in the Fibonacci series is: 1134903170
BUILD SUCCESSFUL (total time: 3 seconds)
```

2. Write a program to count the digits of a given number using recursion

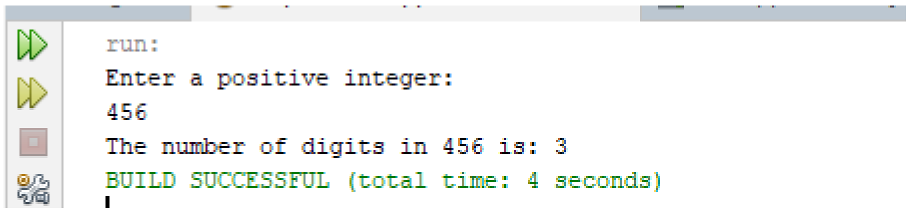
   CODE:



```java
package dsa;
import java.util.Scanner;
public class JavaApplication87 {
    public static int countDigits(int n) {
        // Base case: if n is 0, return 0
        if (n == 0) {
            return 0;
        } else {
            // Recursive call, reduce n by dividing it by 10 and add 1 for the current digit
            return 1 + countDigits(n / 10);
        }
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a positive integer:");
        int n = input.nextInt();

            int digitCount = countDigits(n);
            System.out.println("The number of digits in " + n + " is: " + digitCount);
    }
}
```

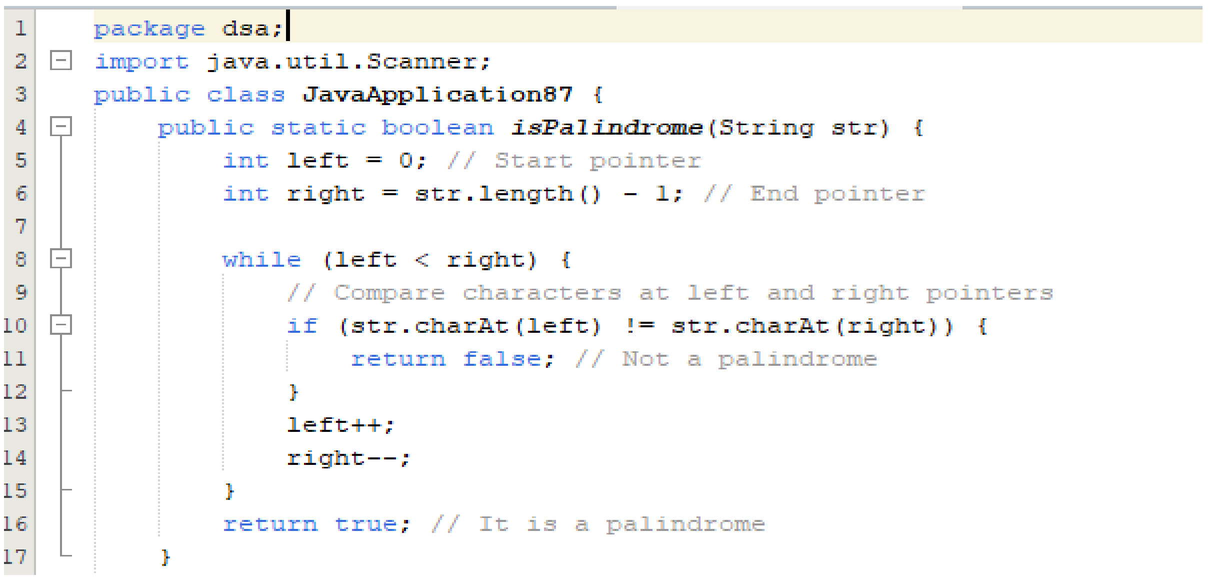
OUTPUT:



```
run:
Enter a positive integer:
456
The number of digits in 456 is: 3
BUILD SUCCESSFUL (total time: 4 seconds)
```

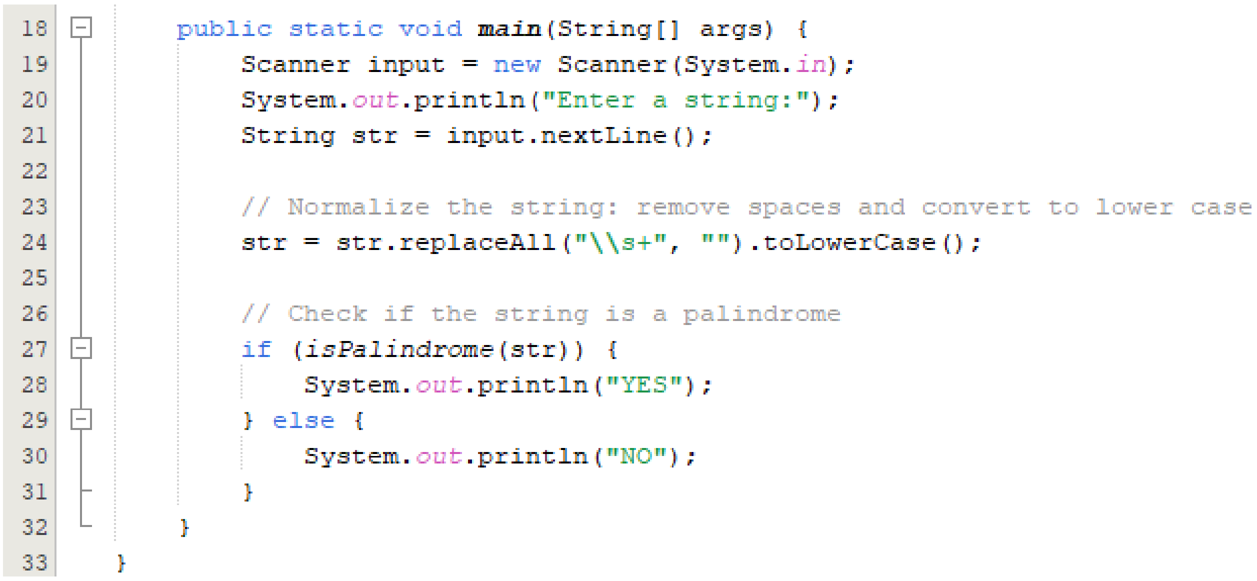
3. Write a java program to check whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards.Print "YES" if the string is a palindrome, otherwise print "NO".

   CODE



```java
package dsa;
import java.util.Scanner;
public class JavaApplication87 {
    public static boolean isPalindrome(String str) {
        int left = 0; // Start pointer
        int right = str.length() - 1; // End pointer

        while (left < right) {
            // Compare characters at left and right pointers
            if (str.charAt(left) != str.charAt(right)) {
                return false; // Not a palindrome
            }
            left++;
            right--;
        }
        return true; // It is a palindrome
    }
```

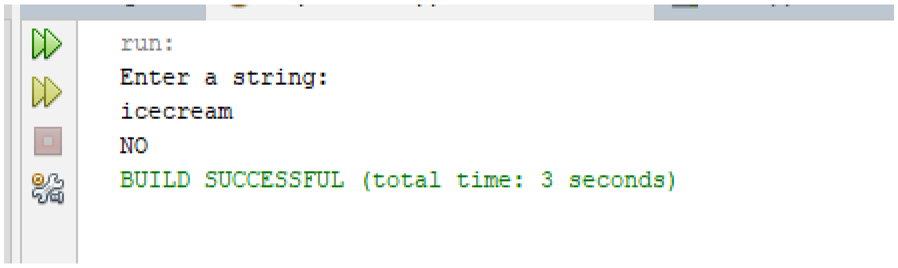


```java
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a string:");
        String str = input.nextLine();

        // Normalize the string: remove spaces and convert to lower case
        str = str.replaceAll("\\s+", "").toLowerCase();

        // Check if the string is a palindrome
        if (isPalindrome(str)) {
            System.out.println("YES");
        } else {
            System.out.println("NO");
        }
    }
}
```

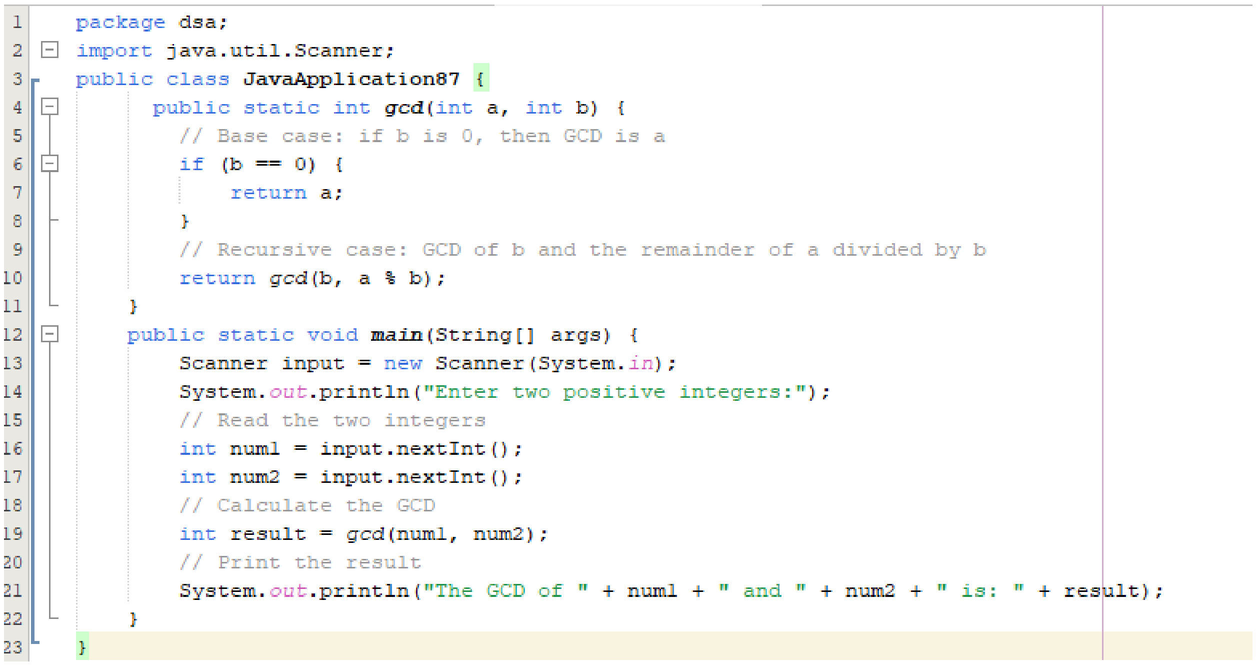
OUTPUT:



```
run:
Enter a string:
icecream
NO
BUILD SUCCESSFUL (total time: 3 seconds)
```

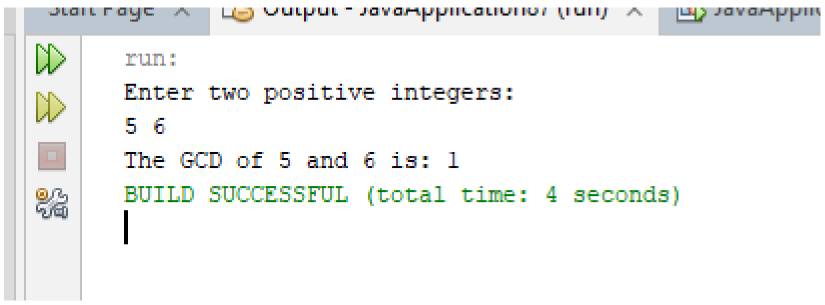
4. Write a recursive program to find the greatest common divisor (GCD) of two numbers using Euclid's algorithm

CODE:



```java
package dsa;
import java.util.Scanner;
public class JavaApplication87 {
      public static int gcd(int a, int b) {
        // Base case: if b is 0, then GCD is a
        if (b == 0) {
            return a;
        }
        // Recursive case: GCD of b and the remainder of a divided by b
        return gcd(b, a % b);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter two positive integers:");
        // Read the two integers
        int num1 = input.nextInt();
        int num2 = input.nextInt();
        // Calculate the GCD
        int result = gcd(num1, num2);
        // Print the result
        System.out.println("The GCD of " + num1 + " and " + num2 + " is: " + result);
    }
}
```

OUTPUT:



```
run:
Enter two positive integers:
5 6
The GCD of 5 and 6 is: 1
BUILD SUCCESSFUL (total time: 4 seconds)
```