Recursion

============

Recursion in Java is a technique in which a method calls itself in order to solve a problem. The problem is divided into smaller sub-problems of the same type, and the method repeats this process until a base condition is met.

### **Key Concepts of Recursion:**

* **Base Case**: The condition that stops the recursion.
* **Recursive Case**: The part of the recursion where the method calls itself with smaller or simpler inputs.

1: Factorial of a number

—--------------------------------------

**package** com.codegnan.recursionprograms;

**public** **class** FactorialNumberRecursion {

// Recursive function to calculate factorial of a number

**public** **static** **int** factorial(**int** n) {

**if** (n == 0) { // Base case: factorial of 0 is 1

**return** 1;

} **else** {

// Recursive case: n \* factorial of (n-1)

**return** n \* *factorial*(n - 1);

}

}

**public** **static** **void** main(String[] args) {

// Calling the factorial function and printing the result

System.***out***.println(*factorial*(5)); // Output: 120

}

}

2: Fibonacci Sequence

—-------------------------------

**package** com.codegnan.recursionprograms;

**public** **class** FibonacciSeriesRecursion {

**public** **static** **int** fibonacci(**int** n) {

// Base cases:

**if** (n == 0) {

**return** 0;

} **else** **if** (n == 1) {

**return** 1;

} **else** {

// Recursive case: calculate Fibonacci(n-1) + Fibonacci(n-2)

**return** *fibonacci*(n - 1) + *fibonacci*(n - 2);

}

}

**public** **static** **void** main(String[] args) {

**int** n = 10;

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

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

}

}

}

3: Sum of Digits of a Number

===============================================

**package** com.codegnan.recursionprograms;

**public** **class** SumOfDigitsUseRecursion {

// Recursive function to sum the digits of a number

**public** **static** **int** sumOfDigits(**int** n) {

**if** (n == 0) { // Base case: no digits left to sum

**return** 0;

} **else** {

// Recursive case: last digit + sum of remaining digits

**return** n % 10 + *sumOfDigits*(n / 10);

}

}

**public** **static** **void** main(String[] args) {

// Calling the sumOfDigits function and printing the result

System.***out***.println(*sumOfDigits*(12345)); // Output: 10

}

}

4. Reverse a String

**package** com.codegnan.recursionprograms;

**public** **class** ReverseStringUsingRecursion {

**public** **class** ReverseString {

**public** **static** String reverseString(String str) {

**if** (str.isEmpty()) {

**return** str;

}

**return** *reverseString*(str.substring(1)) + str.charAt(0);

}

**public** **static** **void** main(String[] args) {

String str = "hello";

String revStr = *reverseString*(str);

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

}

}

}

5. Print Numbers from 1 to n

—---------------------------------------

**package** com.codegnan.recursionprograms;

**public** **class** PrintRecursionUsingStrings {

// Recursive function to print numbers from 1 to n

**public** **static** **void** printNumbers(**int** n) {

**if** (n == 0) { // Base case: when n reaches 0, stop the recursion

**return**;

} **else** {

// Recursive case: print numbers up to n-1, then print n

*printNumbers*(n - 1);

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

}

}

**public** **static** **void** main(String[] args) {

// Calling the printNumbers function to print from 1 to 5

*printNumbers*(100); // Output: 1 2 3 4 5

}

}

6. Palindrome Check

**package** com.codegnan.recursionprograms;

**public** **class** PalindromeStringUsingRecursion {

**public** **static** **boolean** isPalindrome(String str) {

// Base case: If the string is empty or has one character, it's a palindrome

**if** (str.length() <= 1) {

**return** **true**;

}

// Check if the first and last characters are different

**if** (str.charAt(0) != str.charAt(str.length() - 1)) {

**return** **false**;

}

// Recursive case: Check the substring excluding the first and last characters

**return** *isPalindrome*(str.substring(1, str.length() - 1));

}

**public** **static** **void** main(String[] args) {

String str = "raceca";

**boolean** isPal = *isPalindrome*(str);

System.***out***.println(str + " is palindrome? " + isPal);

}

}

7. Count Digits in a Number

—---------------------------------------

**package** com.codegnan.recursionprograms;

**public** **class** CountDigitsUsingRecursion {

// Recursive function to count the number of digits in a number

**public** **static** **int** countDigits(**int** n) {

**if** (n == 0) { // Base case: if n is 0, no more digits to count

**return** 0;

} **else** {

// Recursive case: count the digits of n/10 and add 1 for the current digit

**return** 1 + *countDigits*(n / 10);

}

}

**public** **static** **void** main(String[] args) {

// Calling the countDigits function and printing the result

System.***out***.println(*countDigits*(12345)); // Output: 5

}

}

8.Sum of an Array

—-------------------------------

**package** com.codegnan.recursionprograms;

**public** **class** SumOfArrayUsingRecursion {

// Recursive function to calculate the sum of elements in an array

**public** **static** **int** sum(**int**[] arr, **int** n) {

**if** (n == 0) { // Base case: if no elements left, sum is 0

**return** 0;

} **else** {

// Recursive case: sum of n-1 elements + current element

**return** arr[n - 1] + *sum*(arr, n - 1);

}

}

**public** **static** **void** main(String[] args) {

// Array of integers

**int**[] arr = { 1, 2, 3, 4, 5,6,7 };

// Calling the sum function and printing the result

System.***out***.println(*sum*(arr, arr.length)); // Output: 15

}

}

9. Find Maximum Element in an Array

=======================================

**package** com.codegnan.recursionprograms;

**public** **class** MaximunElementUsingRecursion {

// Recursive function to find the maximum element in an array

**public** **static** **int** findMax(**int**[] arr, **int** n) {

**if** (n == 1) { // Base case: only one element left, return it

**return** arr[0];

}

**int** max = *findMax*(arr, n - 1); // Recursive case: find max of first n-1 elements

// Return the maximum of the current element and the max of the previous

// elements

**return** Math.*max*(arr[n - 1], max);

}

**public** **static** **void** main(String[] args) {

// Array of integers

**int**[] arr = { 1, 4, 2, 9, 7,70};

// Calling the findMax function and printing the result

System.***out***.println(*findMax*(arr, arr.length)); // Output: 9

}

}

10.Reverse Array

====================

**package** com.codegnan.recursionprograms;

**public** **class** PrintReverseArrayUsingRecursion {

// Recursive function to print an array in reverse order

**public** **static** **void** printReverse(**int**[] arr, **int** index) {

**if** (index == arr.length) { // Base case: if index reaches the end of the array, stop recursion

**return**;

}

*printReverse*(arr, index + 1); // Recursive call with the next index

// Print the element after the recursive call (reverse order)

System.***out***.print(arr[index] + " ");

}

**public** **static** **void** main(String[] args) {

// Array of integers

**int**[] arr = { 1, 2, 3, 4, 5 };

// Calling the printReverse function to print the array in reverse order

*printReverse*(arr, 0); // Output: 5 4 3 2 1

}

}