JOBSHEET 11
Recursive Function

1. Objective

- Students understand the concept of recursive functions
- Students are able to implement recursive functions in program code

2. Theory

Recursive functions are functions that call themselves. This can happen because in a recursion function, there is a statement that calls the function itself. When a recursive function is called / executed and then the execution process has arrived at the function call statement itself, the function will be called / executed again. And AGAIN, when the execution process has arrived at the function call statement itself, then the function will be called / executed again, and SO ON until a FINAL CONDITION is obtained where the function call process is no longer performed. If the final condition is not found / does not exist, then the function will be called continuously (infinite loop) and this is not allowed.

In recursive functions, there are two components of the code block, as follows:

- **Base Case**: program code that shows the stop limit of the recursive process, so that if this limit value is met, the recursive process is terminated.

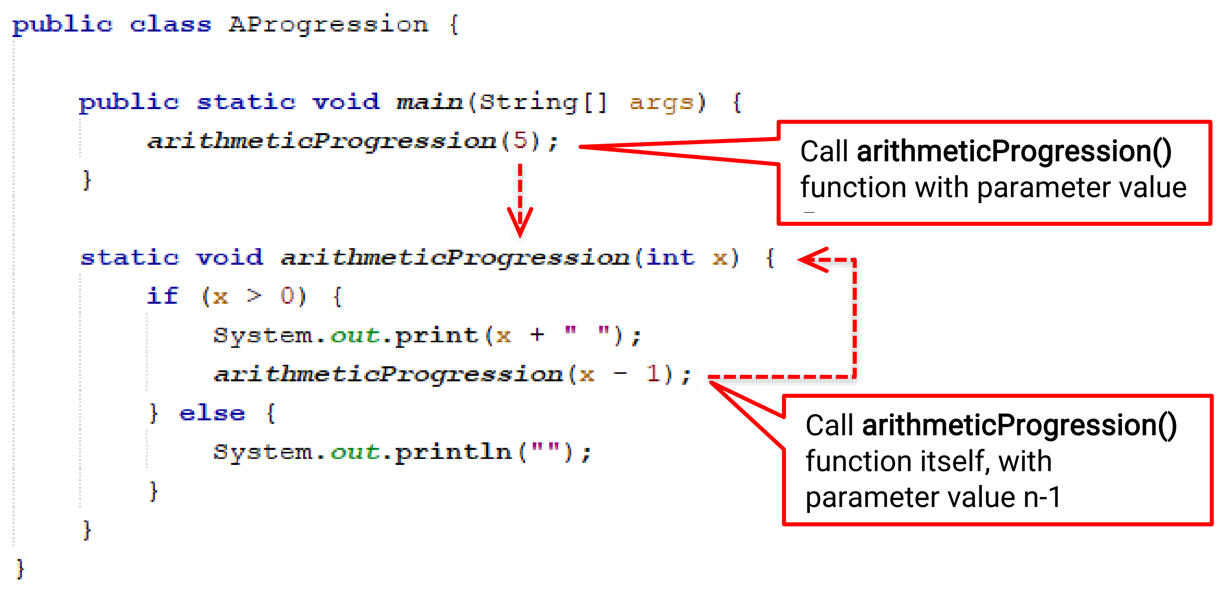
- **Recursion Call** or **Reduction Step**: program code to make calls to itself.

  In general, the format of a recursive function has the following form:

```
if (limit value)
        //solve the problem
else
        //redefines the problem using recursion
```

The IF branch is the base case, while ELSE is the recursion call. For recursion to stop, the recursion call must approach the base case in every recursive function call

Based on this explanation, at first glance it looks like the function is executed repeatedly, and recursive functions are usually used in looping cases. Examine the arithmeticProgression() function below.



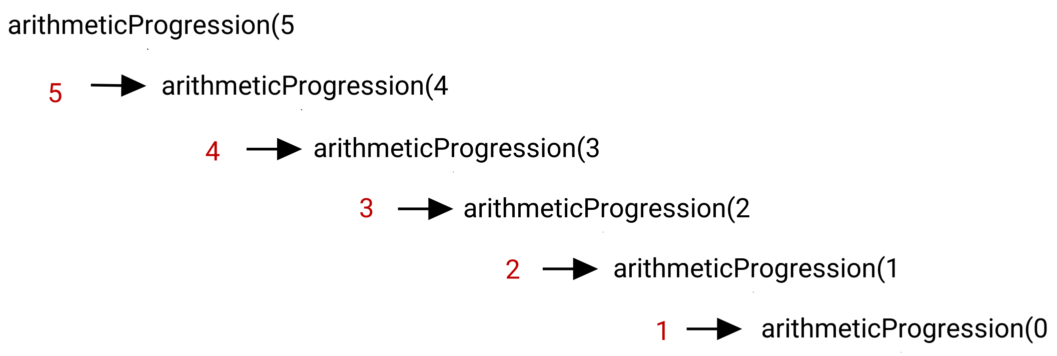
```java
public class AProgression {

    public static void main(String[] args) {
        arithmeticProgression(5);
    }

    static void arithmeticProgression(int x) {
        if (x > 0) {
            System.out.print(x + " ");
            arithmeticProgression(x - 1);
        } else {
            System.out.println("");
        }
    }
}
```

Call **arithmeticProgression()** function with parameter value
Call **arithmeticProgression()** function itself, with parameter value n-1

In the example of the recursive function, the function **arithmeticProgression** is first executed (inside the main function) by sending the parameter value 5 using the command arithmeticProgression(5). From the function call process, the parameter value of 5 will be displayed followed by a call to the arithmeticProgression(4) function. Furthermore, the function call process will display the parameter value, namely 4, followed by a call to the arithmeticProgression(3) function. Furthermore, the function call process will display the parameter value, namely 3, followed by a call to the arithmeticProgression(2) function. Furthermore, the function call process will display the parameter value, namely 2, followed by a call to the arithmeticProgression(1) function. From the function call process, the parameter value will be displayed, namely 1, followed by a call to the arithmeticProgression(0) function. From the function call process, a line break will be displayed and there will be no more **arithmeticProgression** function calls (the recursive function call has been completed).



The display that appears from the result of the arithmeticProgression function call is 5 4 3 2 1.

3. Laboratory

3.1 Experiment 1

In this experiment, a program will be created to calculate the factorial value of a number using a recursive function. In addition, a function for calculating factorial values will be made using an iterative algorithm as a comparison.

1. Create a new project
2. Create a new class, name it **Experiment1**
3. Create a static function with the name **factorialRecursive()**, with return data type that is **int** and has 1 parameter with **int** data type in the form of a number calculated for its factorial value.

```
static int factorialRecursive(int n) {
    if (n == 0) {
        return (1);
    } else {
        return (n * factorialRecursive(n - 1));
    }
}
```

4. Create another static function with the name **factorialIterative()**,with return data type that is **int** and has 1 parameter with **int** data type in the form of a number calculated for its factorial value.

```
static int factorialIterative(int n) {
    int factor = 1;
    for (int i = n; i >= 1; i--) {
        factor = factor * i;
    }
    return factor;
}
```

5. Create a **main** function and make a call to the two previously created functions, and display the results obtained.

```java
public static void main(String[] args) {
    System.out.println(factorialRecursive(5));
    System.out.println(factorialIterative(5));
}
```

6. Compile and run the program.
7. If traced, when calling the **factorialRecursive(5)** function, the process that occurs can be illustrated as follows:

```
5 * factorialRecursive(4)
        4 * factorialRecursive(3)
                3 * factorialRecursive(2)
                        2 * factorialRecursive(1)
                                return 1
                        return 2 * 1 = 2
                return 3 * 2 = 6
        return 4 * 6 = 24
return 5 * 24 = 120
```

3.2Experiment 2

In this experiment, a program will be created to calculate the power of a number using a recursive function.

1. Create a new class, name it **Experiment2**
2. Create a static function with the name **calculatePower()**, with return data type that is **int** and has 2 parameter with **int** data type in the form of numbers to be calculated and the exponents.

```
static int calculatePower(int x, int y) {
    if (y == 0) {
        return (1);
    } else {
        return (x * calculatePower(x, y - 1));
    }
}
```

3. Create a **main** function and declare a Scanner with the name **sc**
4. Create two variables of type int with name **number** and **exponent**
5. Add the following code to accept input from the keyboard

```
System.out.print("Enter a number: ");
number = sc.nextInt();
System.out.print("Enter the exponent: ");
exponent = sc.nextInt();
```

6. Call the **calculatePower** function that was created previously by sending two parameter values.

```
System.out.println(calculatePower(number, exponent));
```

7. Compile and run the program.

3.3Experiment 3

In this experiment, a program will be created to calculate the amount of customer money deposited in the bank after earning interest for several years using the recursive function.

1. Create a new class, name it **Experiment3**

2. Create a static function with the name **calculateInterest()**, with return data type that is **double** and has 2 parameter with **int** data type int in the form of customer balance and time spent saving.

   In this case, it is assumed that the interest set by the bank is 11% annually. Because the interest calculation is **interest * balance**, so to calculate the amount of money after adding interest is **balance + interest * balance**. In this case, the interest rate is 0.11 * balance, and the balance is considered 1 * balance, so 1 * balance + 0.11 * balance can be summarized into **1.11 * balance** for calculating the balance after adding interest (in a year).

```
static double calculateInterest(double balance, int year) {
    if (year == 0) {
        return (balance);
    } else {
        return (1.11 * calculateInterest(balance, year - 1));
    }
}
```

3. Create a **main** function and declare a Scanner with the name **sc**
4. Create a variable of type double named **openingBalance** and a variable of type int named **year**
5. Add the following code to accept input from the keyboard

```
System.out.print("Enter the opening balance: ");
openingBalance = sc.nextDouble();
System.out.print("Enter the duration of saving (years): ");
year = sc.nextInt();
```

6. Call the **calculateInterest** function that was created previously by sending two parameter values.

```
System.out.print("Amount of money after " + year + " years: ");
System.out.println((int) calculateInterest(openingBalance, year));
```

7. Compile and run the program.

Questions!

1. What is a recursive function?
2. What are the examples of recursive functions?
3. In **Experiment 1**, do the **factorialRecursive()** function and **factorialIterative()** function give the same result? Explain the difference in the flow of the program in the use of recursive functions and iterative functions!
4. In **Experiment 2**, there is a recursive function call **calculatePower(number, exponent)** in the main function, then the **calculatePower()** function calls are repeated. Explain how long the function calling process will be executed!
5. In **Experiment 3**, state which program code block is the "base case" and "recursion call"!

4. Assignment

1. Create a program to display the numbers n through 0 using recursive and iterative functions. (**RecursiveDescendingSeries**).
2. Create a program that includes a recursive function for calculating factorial numbers. For example f = 8, it will produce 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = **36** (**RecursiveAddition**).
3. Create a program that includes a recursive function to check whether a number n is a prime number or not. n is said to be not a prime number if it is evenly divided by a number less than n. (**RecursivePrimaCheck**).