1st Assignment

Recursive function and Introduction to JAVA programming

Self-Assessment: 100 ทำงานสำเร็จได้ด้วยตนเอง สามารถประยุกต์งานให้เข้ากับปัญหาอื่นที่มีลักษณะคล้ายๆกัน สามารถแนะนำ/แก้ปัญหาให้กับผู้อื่นได้

What I've learned: Basic java syntax, recursion process (which line will be executed first)

Source code:

```
public class AssignmentI {

    static long calcFactorial(int n){

        long ans;

        if(n == 0){

            System.out.println(n + "! is a base case. Returning 1 as an answer");
            ans = 1;
            return ans;
        }
        else{

            System.out.println(n + "! is a recursive case. Returning " + n + " * " + (n - 1) + "!");
            ans = calcFactorial(n - 1);
            System.out.print("Returning " + (n - 1) + "! = " + ans + " to calculate " + n + "!");
            System.out.println(" = [" + n + " * " + (n - 1) + "!] = " + n + " * " + ans + " = " + (n * ans));
            return ans * n;
        }
    }
```

Method: calcFactorial (int n)

Calculate n! using normal recursive method (n * (n – 1)!)

```
static int getInt(){

    Scanner sc = new Scanner(System.in);
    boolean success = false;
    int input = 0;

    while(!success){

        try{

            success = true;
            System.out.print("Input n: ");
            input = sc.nextInt();
        }
        catch(InputMismatchException e){

            System.out.println("Invalid input. Only positive integer 1 - 20 allowed. Please try again");
            sc.nextLine();
            success = !success;
        }
        if(input < 0 || 15 < input){

            System.out.println("Invalid input. Only positive integer 1 - 20 allowed. Please try again");
            sc.nextLine();
            success = !success;
        }
    }

    return input;
}
```

Method: getInt()

Get user input using Scanner class and verify whether the input is valid or not. Integer "input" will be returned if valid

Boolean variable "Success" is used to keep track and control the iteration

```java
public static void main(String[] args) {

    long normalAnswer, TRAnswer;
    int input;
    boolean running = true;

    Scanner sc = new Scanner(System.in);

    while(running){

        System.out.println("Ultimate recursion program");
        System.out.println("Written by 62070501051 Phattarapol Lertchaisirikul");
        System.out.println("Program calculate n! using both NORMAL recursive method and TAIL recursive method");
        System.out.println("(n must be 15 or lower)");

        input = getInt();

        System.out.println("\nInitiating normal recursive method...");
        normalAnswer = calcFactorial(input);

        System.out.println("\nInitiating tail call optimization...");
        TRAnswer = calcFactorialTR(input, 1);

        System.out.println("");
        System.out.println("The answer of " + input + " factorial calculated using ordinary recursion (O(n)) is " + normalAnswer);
        System.out.println("The answer of " + input + " factorial calculated using tail recursion (O(1)) is " + TRAnswer);

        System.out.println("press [y] to continue, others to exit.");

        if(sc.next().equals("y")){

            System.out.println("End program.");
            running = !running;
        }
```

Main method

Includes: heading, description, author, and input requirement

Boolean variable “running” is used to track the state and terminate the runtime

Test cases:

```
Ultimate recursion program
Written by 62070501051 Phattarapol Lertchaisirikul
Program calculate n! using both NORMAL recursive method and TAIL recursive method
(n must be 15 or lower)
Input n: 4

Initiating normal recursive method...
4! is a recursive case. Returning 4 * 3!
3! is a recursive case. Returning 3 * 2!
2! is a recursive case. Returning 2 * 1!
1! is a recursive case. Returning 1 * 0!
0! is a base case. Returning 1 as an answer
Returning 0! = 1 to calculate 1! = [1 * 0!] = 1 * 1 = 1
Returning 1! = 1 to calculate 2! = [2 * 1!] = 2 * 1 = 2
Returning 2! = 2 to calculate 3! = [3 * 2!] = 3 * 2 = 6
Returning 3! = 6 to calculate 4! = [4 * 3!] = 4 * 6 = 24

The answer of 4 factorial calculated using ordinary recursion (O(n)) is 24
press [y] to continue, others to exit.
```

1st case: n = 4

```
The answer of 4 factorial calculated using ordinary recursion (O(n)) is 24
press [y] to continue, others to exit.
y
Ultimate recursion program
Written by 62070501051 Phattarapol Lertchaisirikul
Program calculate n! using both NORMAL recursive method and TAIL recursive method
(n must be 15 or lower)
Input n: x
Invalid input. Only positive integer 1 - 20 allowed. Please try again
Input n: 3x
Invalid input. Only positive integer 1 - 20 allowed. Please try again
Input n: -1
Invalid input. Only positive integer 1 - 20 allowed. Please try again
Input n: 20
Invalid input. Only positive integer 1 - 20 allowed. Please try again
Input n: 0

Initiating normal recursive method...
0! is a base case. Returning 1 as an answer

The answer of 0 factorial calculated using ordinary recursion (O(n)) is 1
press [y] to continue, others to exit.
y
Ultimate recursion program
Written by 62070501051 Phattarapol Lertchaisirikul
Program calculate n! using both NORMAL recursive method and TAIL recursive method
(n must be 15 or lower)
Input n: 1

Initiating normal recursive method...
1! is a recursive case. Returning 1 * 0!
0! is a base case. Returning 1 as an answer
Returning 0! = 1 to calculate 1! = [1 * 0!] = 1 * 1 = 1

The answer of 1 factorial calculated using ordinary recursion (O(n)) is 1
press [y] to continue, others to exit.
n
End program.
BUILD SUCCESSFUL (total time: 2 minutes 34 seconds)
```

Invalid inputs: n = x, 3x, -1, 20

n = 0

n = 1

Recursive Process:

We'll take a look at our recursive function used to calculate the value of the nth factorial