JOBSHEET 11
Recursive Function

1. Learnig Outcomes

- Student must be able to master recursive function concept
- Student must be proficient to implement recursive function as a problem solving option

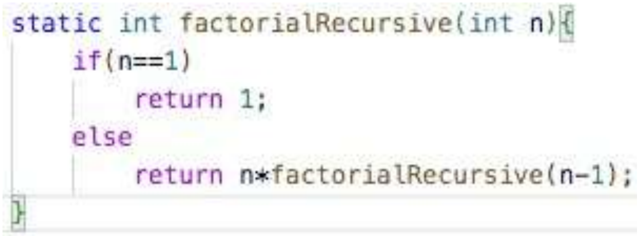
2. Labs Activity

2.1 Experiment 1

Times: 60 minutes

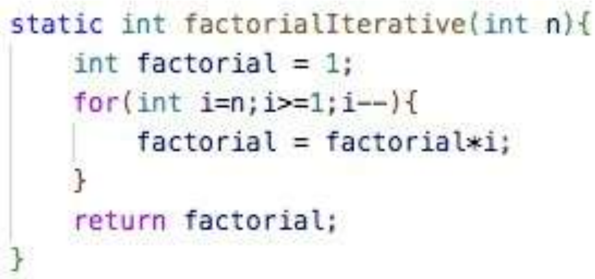
In this experiment, we will implement a program to calculate factorial number by using **recursive** function. An **iterative**-based factorial program will be developed also, for comparative reason.

1. Create a new Java class and save it as **RecursiveStudentIDNumber.java**
2. Create a new **static** function `factorialRecursive()` that has **int** datatype and 1 **int** parameter. This parameter will be used to pass a value to calculate factorial.



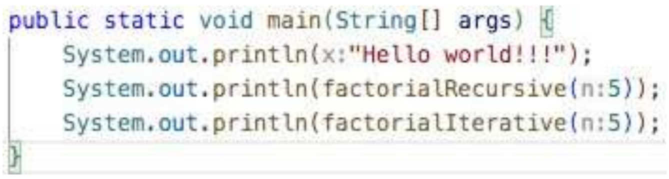
```java
static int factorialRecursive(int n){
    if(n==1)
        return 1;
    else
        return n*factorialRecursive(n-1);
}
```

3. Create a new **static** function to calculate factorial using iterative approach. Name the function as `factorialRecursive()`, with `int` datatype and also has 1 **int** parameter.



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

4. Create a **main** function to call both of recursive function by passing a value to be calculate the factorial number. Also display the result in this main function.



```
public static void main(String[] args) {
    System.out.println(x:"Hello world!!!");
    System.out.println(factorialRecursive(n:5));
    System.out.println(factorialIterative(n:5));
}
```

5. Run the code and observe the result!
6. When we call function `factorialRecursive(5)`,the following step illustrates how the recursive function called as well as how the result computed (in reverse of recursive calls):

5 * factorialRecursive(4)

4 * factorialRecursive(3)

3 * factorialRecursive (2)

2 * factorialRecursive (1)

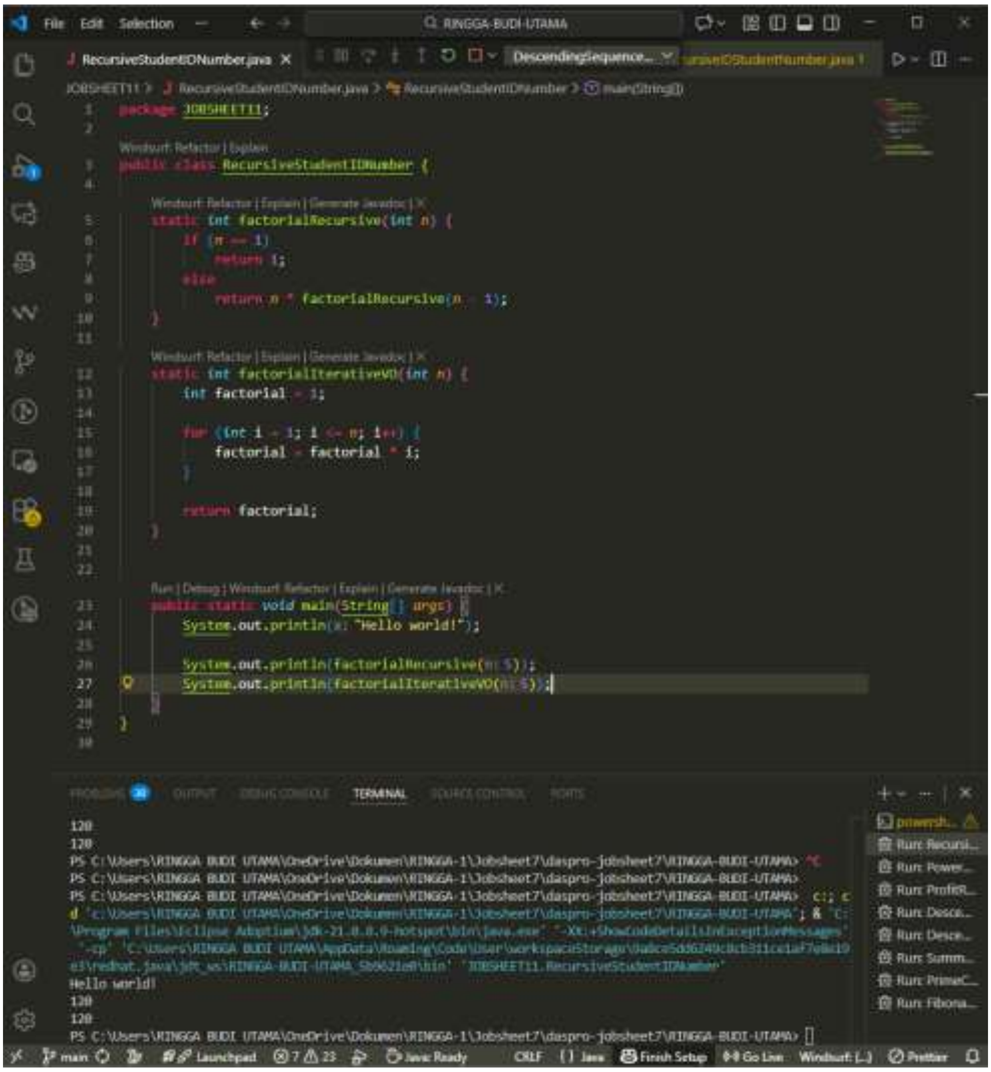
return 1

return 2 * 1 = 2

return 3 * 2 = 6

return 4 * 6 = 24

return 5 * 24 = 120



File Edit Selection
RINGGA-BUDI-UTAMA
RecursiveStudentIDNumber.java
DescendingSequence...
JOBSHEET11 > RecursiveStudentIDNumber.java > RecursiveStudentIDNumber > main(String[])
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SOURCE CONTROL PORTS
powersh...
Run: Recursi...
Run: Power...
Run: ProfitR...
Run: Desce...
Run: Desce...
Run: Summ...
Run: PrimeC...
Run: Fibona...
main Launchpad 7 23 Java: Ready CRLF {} Java Finish Setup Go Live Windsurf: (...) Prettier

```java
package JOBSHEET11;

Windsurf: Refactor | Explain
public class RecursiveStudentIDNumber {

    Windsurf: Refactor | Explain | Generate Javadoc | X
    static int factorialRecursive(int n) {
        if (n == 1)
            return 1;
        else
            return n * factorialRecursive(n - 1);
    }

    Windsurf: Refactor | Explain | Generate Javadoc | X
    static int factorialIterativeVO(int n) {
        int factorial = 1;

        for (int i = 1; i <= n; i++) {
            factorial = factorial * i;
        }

        return factorial;
    }


    Run | Debug | Windsurf: Refactor | Explain | Generate Javadoc | X
    public static void main(String[] args) {
        System.out.println(x: "Hello world!");

        System.out.println(factorialRecursive(n: 5));
        System.out.println(factorialIterativeVO(n: 5));
    }
}
```

```
120
120
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> ^C
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>  c:; c
d 'c:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA'; & 'C:
\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
 '-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\0a6ca5dd6249c8cb311ca1af7e0d19
e3\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_5b9621a0\bin' 'JOBSHEET11.RecursiveStudentIDNumber'
Hello world!
120
120
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> 
```

Question

1. After observing the experiment of recursive function above, what is the definition of recursive function?
2. How the recursive function works?
3. From the experiment above, do factorialRecursive() and factorialIterative() have the similar result? Then, what are the differences between recursive and iterative if both are having the same result?

Answer :

1. ***A recursive function is a function that calls itself during its execution until it reaches a stopping condition (base case).***
2. ***How a recursive function works:***
    - ***The function keeps calling itself repeatedly with a smaller value.***

   - ***The process stops when it reaches the base case.***
   - ***After the base case is reached, the function starts returning back step-by-step.***

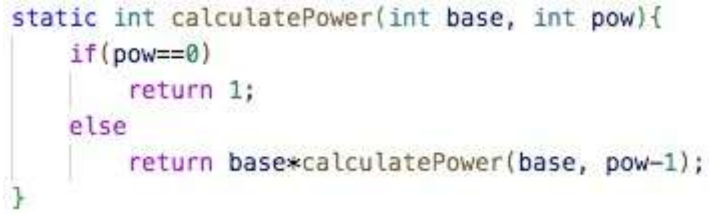
3. ***Yes, factorialRecursive() and factorialIterative() produce the same result. The differences are:***
   - ***Recursive uses self-calling functions.***
   - ***Iterative uses loops (for/while) without self-calling.***
   - ***Recursive is more elegant/shorter, while iterative is faster and uses less memory.***

2.2 Experiment 2

Times : 60 minutes

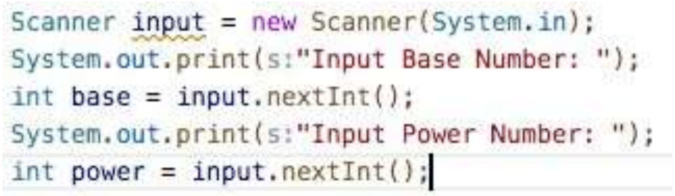
In the following experiment, a program to calculate power of a number using recursive function.

1. Create a new Java class, and name it as **PowerRecursiveIDStudentNumber.java**
2. Create a new **static** function **calculatePower()** that has **int** datatype with 2 int parameters (base and power number).



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

3. Create a new **main()** function that will get user input for base number and power number, will call the recursive function to calculate the power result and will display the final result.

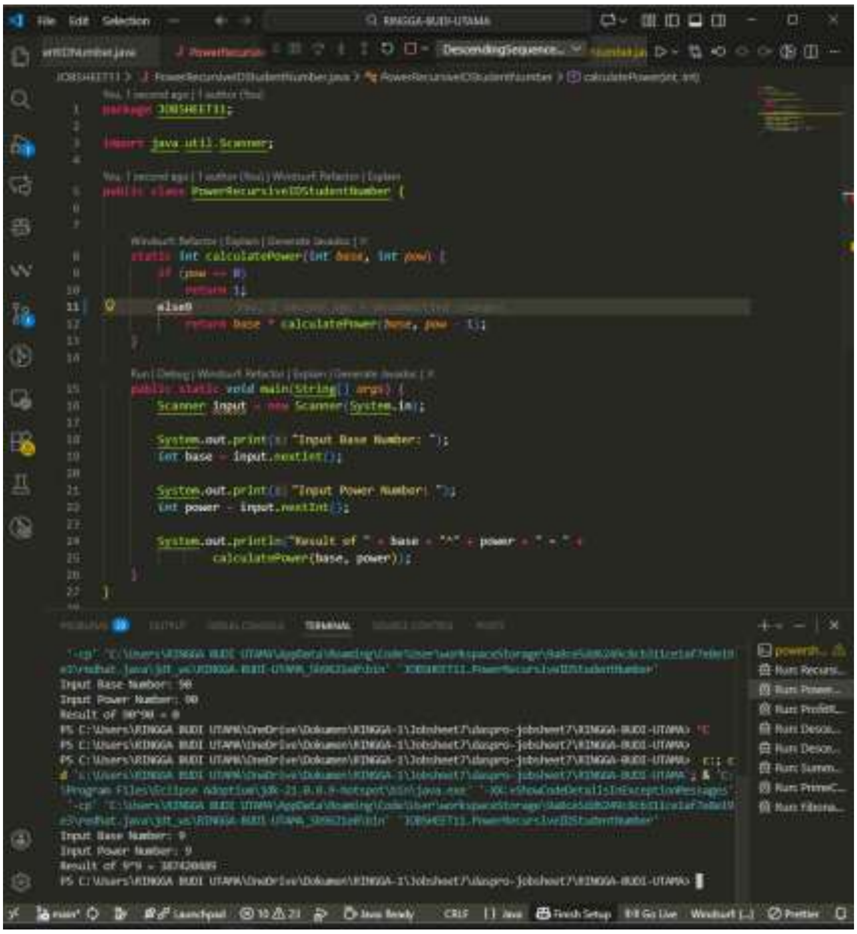


```
Scanner input = new Scanner(System.in);
System.out.print(s:"Input Base Number: ");
int base = input.nextInt();
System.out.print(s:"Input Power Number: ");
int power = input.nextInt();
```

4. Call function **calculatePower()** and do not forget to pass 2 values into parameter, otherwise the error will arises.

```
System.out.println("Result of "+base+" power "+power+" = "+
                    calculatePower(base, power));
```

5. Run the program and observer the result!



```java
package JOBSHEET11;

import java.util.Scanner;

public class PowerRecursiveIDStudentNumber {


    static int calculatePower(int base, int pow) {
        if (pow == 0)
            return 1;
        else
            return base * calculatePower(base, pow - 1);
    }

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

        System.out.print(s: "Input Base Number: ");
        int base = input.nextInt();

        System.out.print(s: "Input Power Number: ");
        int power = input.nextInt();

        System.out.println("Result of " + base + "^" + power + " = " +
                calculatePower(base, power));
    }
}
```

```
'-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\[illegible]\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_[illegible]\bin' 'JOBSHEET11.PowerRecursiveIDStudentNumber'
Input Base Number: 90
Input Power Number: 90
Result of 90^90 = 0
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> ^C
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> c:; c
d 'c:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA'; & 'C:
\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
 '-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\[illegible]\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_[illegible]\bin' 'JOBSHEET11.PowerRecursiveIDStudentNumber'
Input Base Number: 9
Input Power Number: 9
Result of 9^9 = 387420489
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
```

Question!

1. In Experiment 2, there is a recursive function call **calculatePower()** in the main function, then the function **calculatePower**() is called repeatedly. Explain how long the function calling process will run!

The recursive function `calculatePower()` ***will keep calling itself as long as the power value is greater than 0. Each recursive call decreases the power by 1, and the process stops when*** `power == 0` ***(the base case). So the function runs until the power reaches 0, then it returns the result back step-by-step.***

2. Add program code to print the power calculation series. Example: **calculatePower(2,5)** will print 2x2x2x2x2x1 = 32!

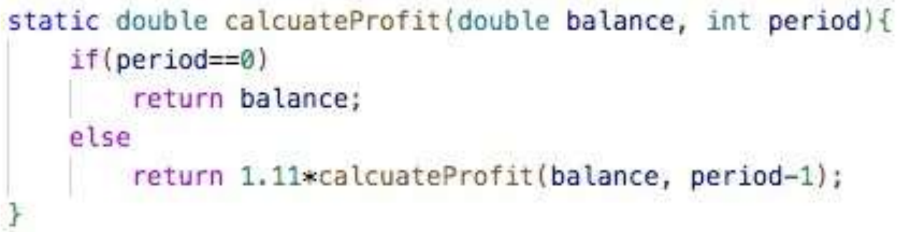
2.3 Experiment 3

Time : 60 minutes

We will continue this experiment by creating a new program to calculate the amount of investor money that will be used to invest after earning profits for several years. This program must implement recursive function to calculate the profits that would be got by investor.

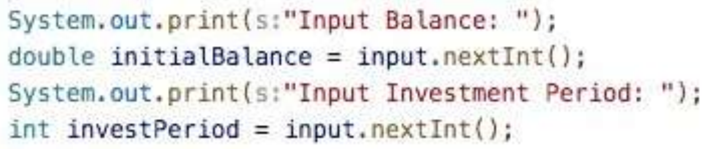
1. Create a new class with name **ProvitRecursiveStudentIDNumber.java**
2. Create a new **static** function **calculateProfit**() that has **double** datatype and 2 **int** parameters. The **balance** would be passed through first parameter and **investment period** would be passed through parameter 2. The investor will get 11% interest annually.

   Since to calculate profit is **interest * balance**, so the total amount of investor money will be **balance+interest*balance**. In this case, the amount of **interest** is **0.11 * balance**, and the final balance is considered as 1*balance, so that 1 * balance + 0.11 * or equivalent formula to calculate the final balance will be **1.11 * balance** in a year.



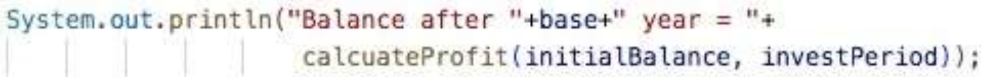
```
static double calcuateProfit(double balance, int period){
    if(period==0)
        return balance;
    else
        return 1.11*calcuateProfit(balance, period-1);
}
```

3. Create a new **main()** function, then get the user input for initial **balance** and **investment period**.



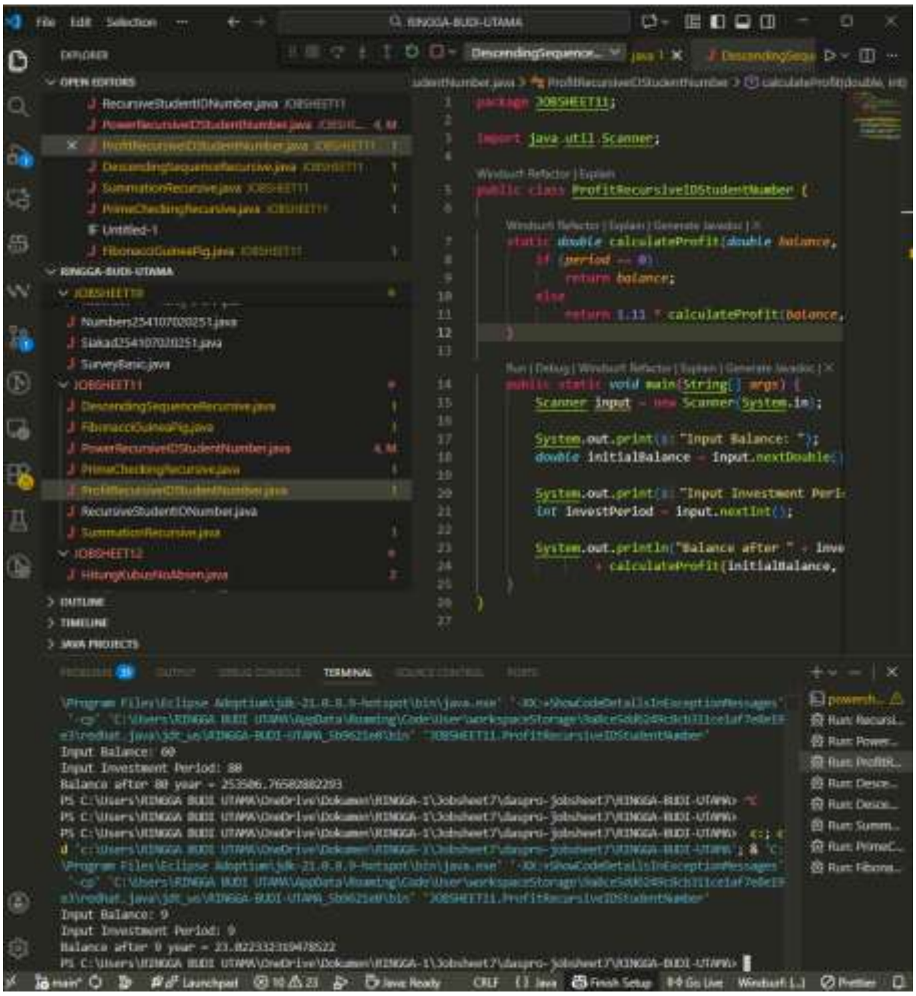
```
System.out.print(s:"Input Balance: ");
double initialBalance = input.nextInt();
System.out.print(s:"Input Investment Period: ");
int investPeriod = input.nextInt();
```

4. Then call function **calculateProfit()** from main function.



```
System.out.println("Balance after "+base+" year = "+
                    calcuateProfit(initialBalance, investPeriod));
```

5. Run the program and observe the result!



```java
package JOBSHEET11;

import java.util.Scanner;

public class ProfitRecursiveIDStudentNumber {

    static double calculateProfit(double balance,
        if (period == 0)
            return balance;
        else
            return 1.11 * calculateProfit(balance,
    }

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

        System.out.print("Input Balance: ");
        double initialBalance = input.nextDouble();

        System.out.print("Input Investment Peri
        int investPeriod = input.nextInt();

        System.out.println("Balance after " + inve
                + calculateProfit(initialBalance,
    }
}
```

```
Input Balance: 60
Input Investment Period: 88
Balance after 88 year = 253506.76582882293
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
Input Balance: 9
Input Investment Period: 9
Balance after 9 year = 23.022332319478522
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
```

Question!

1. From the above experiment, which statements that is classified as "**base case**" and "**recursion call**"!

Base case : if (period == 0)

return balance;

recursion call : return 1.11 * calculateProfit(balance, period - 1);

2. Explain using simulation or trace the expansion phase and substitution phase of **calculateProfit(100000,3)** function!

Expansion phase :

calculateProfit(100000,3)

= 1.11 * calculateProfit(100000,2)

= 1.11 * calculateProfit(100000,1)

= 1.11 * calculateProfit(100000,0)

Substituion phase :

calculateProfit(100000,0) = 100000

= 1.11 * 100000 = 111000

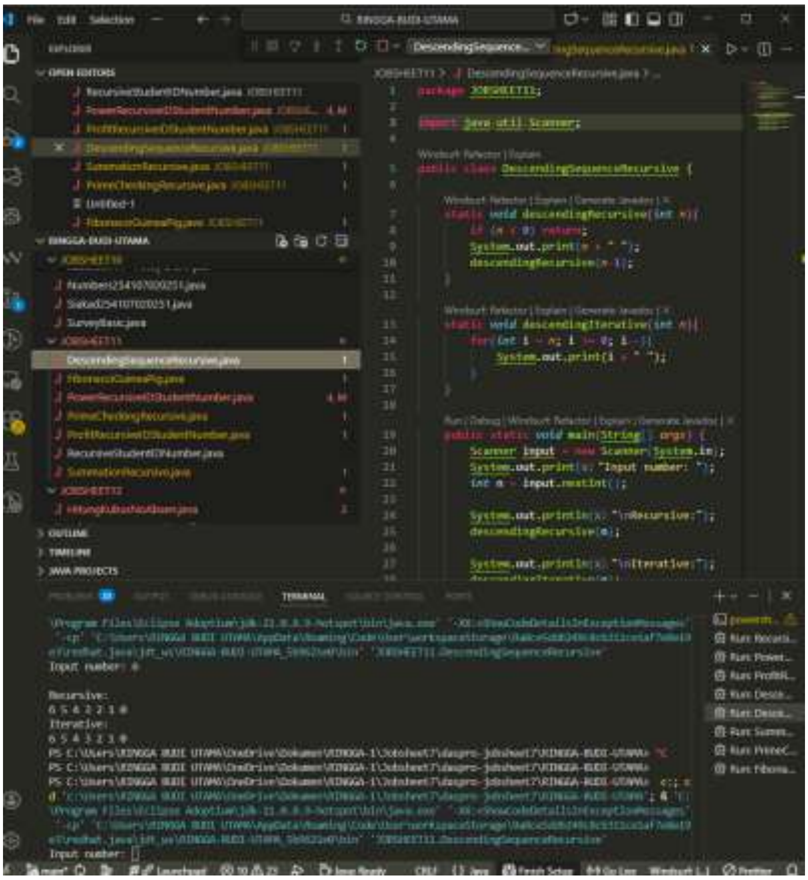
= 1.11 * 111000 = 123210

= 1.11 * 123210 = 136761

3. Assignment

Times: 120 minutes

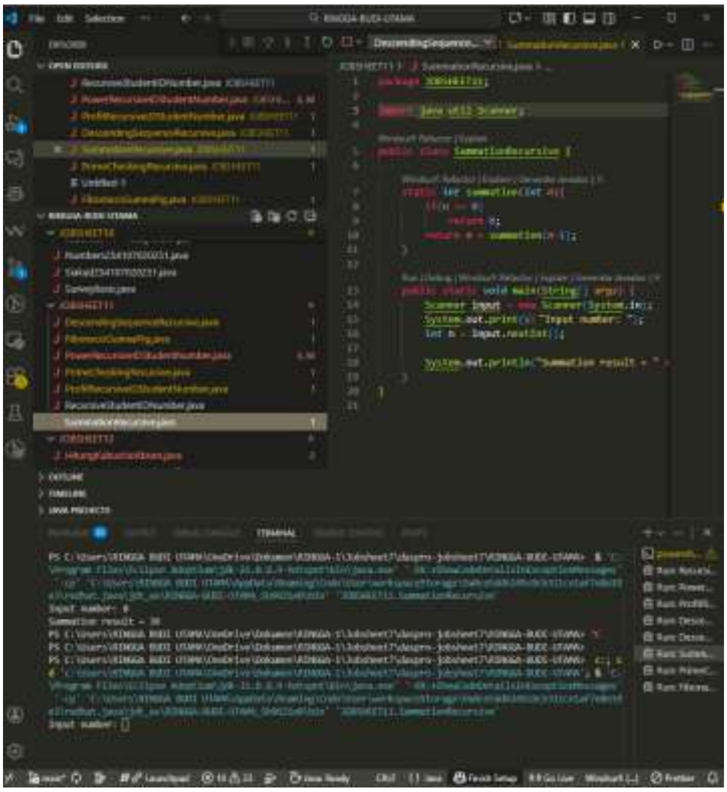
1. Write a program to display numbers n to 0 using recursive functions and iterative functions. (**DescendingSequenceRecursive**).



```java
package JOBSHEET11;

import java.util.Scanner;

public class DescendingSequenceRecursive {

    static void descendingRecursive(int n){
        if (n < 0) return;
        System.out.print(n + " ");
        descendingRecursive(n-1);
    }

    static void descendingIterative(int n){
        for(int i = n; i >= 0; i--){
            System.out.print(i + " ");
        }
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input number: ");
        int n = input.nextInt();

        System.out.println("\nRecursive:");
        descendingRecursive(n);

        System.out.println("\nIterative:");
        descendingIterative(n);
```

```
Input number: 6

Recursive:
6 5 4 3 2 1 0
Iterative:
6 5 4 3 2 1 0
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> ^C
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
Input number:
```

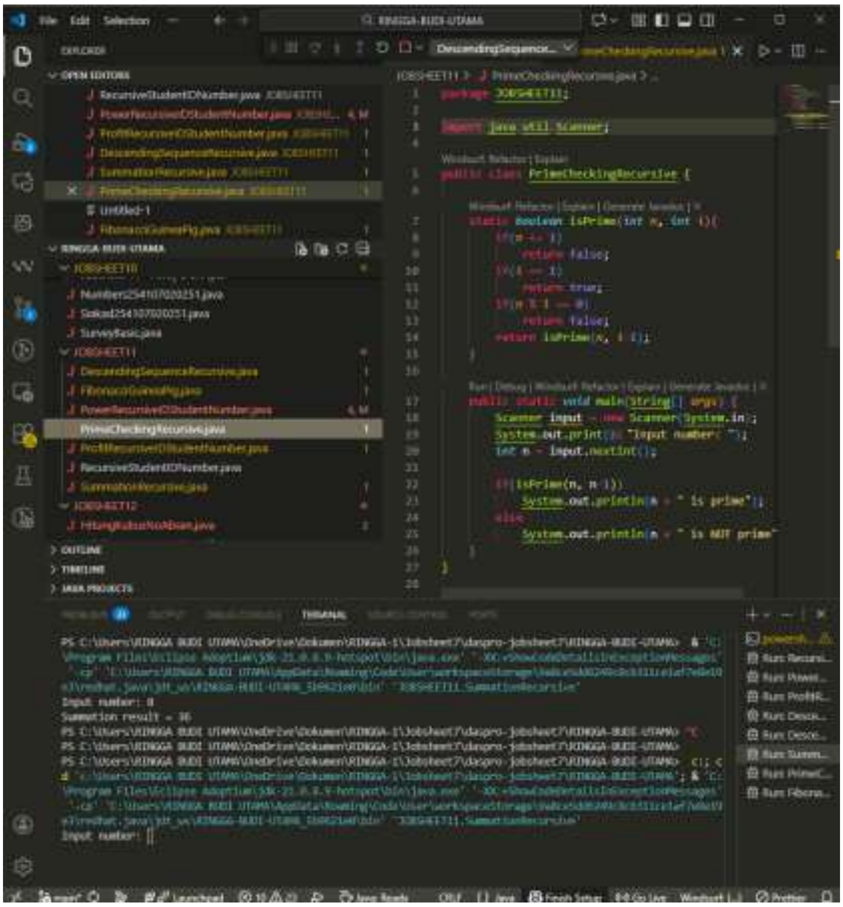
2. Create a program to sum the numbers using recursive function. For example n = 8, then it will result 1+2+3+4+5+6+7+8 = **36** (**SummationRecursive**).



```
package JOBSHEET11;

import java.util.Scanner;

public class SummationRecursive {

    static int summation(int n){
        if(n == 0)
            return 0;
        return n + summation(n-1);
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input number: ");
        int n = input.nextInt();

        System.out.println("Summation result = " +
    }
}
```

```
Input number: 8
Summation result = 36
```

3. Create a program that contains a recursive function to check whether a number **n** is a prime number or not. A number will not be classified as a prime number if it is divisible by a number less than **n**. (**PrimeCheckingRecursive**).

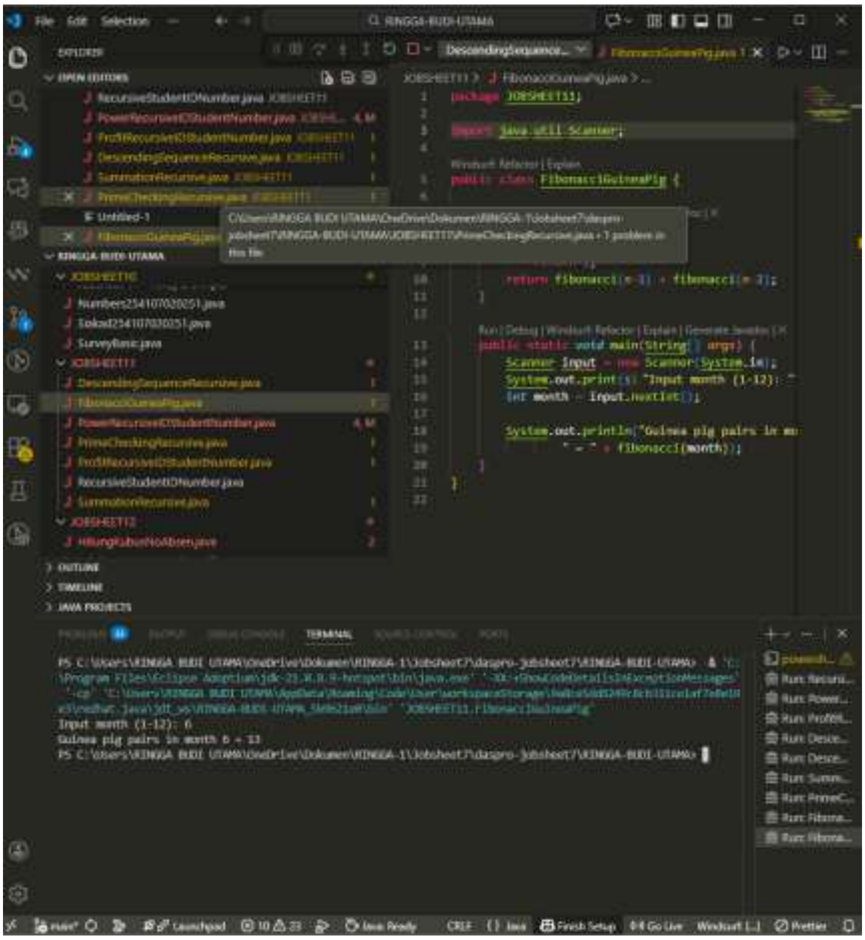


```java
package JOBSHEET11;

import java.util.Scanner;

public class PrimeCheckingRecursive {

    static boolean isPrime(int n, int i){
        if(n <= 1)
            return false;
        if(i == 1)
            return true;
        if(n % i == 0)
            return false;
        return isPrime(n, i-1);
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input number: ");
        int n = input.nextInt();

        if(isPrime(n, n-1))
            System.out.println(n + " is prime");
        else
            System.out.println(n + " is NOT prime");
    }
}
```

```
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.0.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\[illegible]\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_[illegible]\bin' 'JOBSHEET11.SummationRecursive'
Input number: 8
Summation result = 36
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> ^C
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>  c:; cd 'c:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.0.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\[illegible]\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_[illegible]\bin' 'JOBSHEET11.SummationRecursive'
Input number: 
```

4. A pair of newborn guinea pigs (male and female) are placed in a nursery. After two months the guinea pig pair gave birth to a pair of twin guinea pigs (male and female). Every pair of guinea pigs that is born will also give birth to a pair of guinea pigs every 2 months. How many pairs of guinea pigs were there at the end of the 12th month? Write a program using a recursive function! (**Fibonacci**). And the following table illustrates the calculation.

Month	Pair Number		Pair Total
	Productive	Non-Productive	
1	0	1	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8
7	5	8	13
8	8	13	21
9	13	21	34
10	21	34	55
11	34	55	89
12	55	89	144



```
package JOBSHEET11;

import java.util.Scanner;

public class FibonacciGuineaPig {

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

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print(s: "Input month (1-12): "
        int month = input.nextInt();

        System.out.println("Guinea pig pairs in mo
                " = " + fibonacci(month));
    }
}
```

```
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> & 'C:
\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
 '-cp' 'C:\Users\RINGGA BUDI UTAMA\AppData\Roaming\Code\User\workspaceStorage\[illegible]
[illegible]\redhat.java\jdt_ws\RINGGA-BUDI-UTAMA_[illegible]\bin' 'JOBSHEET11.FibonacciGuineaPig'
Input month (1-12): 6
Guinea pig pairs in month 6 = 13
PS C:\Users\RINGGA BUDI UTAMA\OneDrive\Dokumen\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
```