

DJILLALI LIABES UNIVERSITY OF SIDI BEL ABBES
FACULTY OF EXACT SCIENCES
DEPARTMENT OF COMPUTER SCIENCES



Module : Algorithmique et Complexité
1ST YEAR OF MASTER'S DEGREE IN
NETWORKS, INFORMATION SYSTEMS & SECURITY (RSSI)
2021/2022

Comparaison entre algorithmes itératifs pour le Calcul de la suite de Fibonacci

Author:

HADJAZI M.Hisham
AMUER Wassim Malik

Supervisor:

Dr. MME. BELKHODJA
ZENAIIDI Lamia

*A paper submitted in fulfillment of the requirements for the
TP-01*

October 21, 2021

Contents

List of Figures	iii
1 Solutions of Fiche TP-01	1
1.1 Question.1 / Lire attentivement le code ci-joint, suivre les indications puis compléter la classe Fibo.java avec 3 méthodes itératives différentes permettant de calculer le n ième terme de la suite. Pour le stockage des termes.	2
1.1.1 1. La première méthode doit utiliser un tableau de taille n+1. . .	2
1.1.2 2. La deuxième méthode doit utiliser 2 variables.	3
1.1.3 3. La troisième méthode doit utiliser un tableau de taille 2. . . .	3
1.2 Question.2 / Exécuter le programme pour différentes valeurs de n=0,1,..., 10, ..., 80,...100,...	4
1.2.1 N = 0	4
N = 0 and Eclipse IDE	4
N = 0 and Terminal	4
1.2.2 N = 1	5
N = 1 and Eclipse IDE	5
N = 1 and Terminal	5
1.2.3 N = 10	6
N = 10 and Eclipse IDE	6
N = 10 and Terminal	6
1.2.4 N = 80	7
N = 80 and Eclipse IDE	7
N = 80 and Terminal	7
1.2.5 N = 100	8
N = 100 and Eclipse IDE	8
N = 100 and Terminal	8
1.2.6 N = 150	9
N = 150 and Eclipse IDE	9
N = 150 and Terminal	9
1.3 Question.3 / Donner la signification de chacune des 4 instructions suivantes :	10
1.3.1 startTime = System.nanoTime();	10
1.3.2 System.out.println("F("+n+") = "+methode1(n));	10
1.3.3 endTime = System.nanoTime();	11
1.3.4 res= (float) (endTime-startTime) / 1000000 ;	11
1.4 Question.4 / Remplir le tableau suivant :	11
1.5 Question.5 / Faire des captures d'écran de 3 exécutions de votre choix.	11
1.5.1 N = 0	11
N = 0 and Eclipse IDE	11
N = 0 and Terminal	12
1.5.2 N = 10	12

	N = 10 and Eclipse IDE	12
	N = 10 and Terminal	13
1.5.3	N = 100	13
	N = 100 and Eclipse IDE	13
	N = 100 and Terminal	14
1.6	Question.6 / Calculer les complexités temporelle et spatiale de cha- cune des 3 méthodes utilisées. Que peut-on conclure ?	15
1.6.1	1. La première méthode doit utiliser un tableau de taille $n+1$	15
	Time Complexity of Method 1 with Barometric operation	15
	Time Complexity of Method 1 with Frequency Count Method	16
	Space Complexity of Method 1	16
1.6.2	2. La deuxième méthode doit utiliser 2 variables.	17
	Time Complexity of Method 2 with Barometric operation	17
	Time Complexity of Method 2 with Frequency Count Method	18
	Space Complexity of Method 2	18
1.6.3	3. La troisième méthode doit utiliser un tableau de taille 2.	19
	Time Complexity of Method 3 with Barometric operation	19
	Time Complexity of Method 3 with Frequency Count Method	20
	Space Complexity of Method 3	20
1.7	Conclusion	21

List of Figures

1.1	Value of n=0 using eclipse IDE	4
1.2	Value of n=0 using Terminal	4
1.3	Value of n=1 using eclipse IDE	5
1.4	Value of n=1 using Terminal	5
1.5	Value of n=10 using eclipse IDE	6
1.6	Value of n=10 using Terminal	6
1.7	Value of n=80 using eclipse IDE	7
1.8	Value of n=80 using Terminal	7
1.9	Value of n=100 using eclipse IDE	8
1.10	Value of n=100 using Terminal	8
1.11	Value of n=150 using eclipse IDE	9
1.12	Value of n=150 using Terminal	9
1.13	System.out.println in Java	10
1.14	Value of n=0 using eclipse IDE	11
1.15	Value of n=0 using Terminal	12
1.16	Value of n=10 using eclipse IDE	12
1.17	Value of n=10 using Terminal	13
1.18	Value of n=100 using eclipse IDE	13
1.19	Value of n=100 using Terminal	14

Chapter 1

Solutions of Fiche TP-01

Notes regarding this solution :

This solution and the executions of the code in it was done in the following machine :

- *OS* : Linux Mint 20.2 Cinnamon Kernel v.5.4.0-88
- *IDE* : Eclipse Eclipse IDE for Java Developers Version: 2019-12 (4.14.0)
- *Java version*: 11.0.11

Définition du problème :

Les nombres de la suite de Fibonacci représentent la reproduction des lapins : Possédant au départ un couple de lapins, combien de couples de lapins obtient-on en douze mois si chaque couple engendre tous les mois un nouveau couple à compter du second mois de son existence ?

Janvier : 1 couple

Février : 1 couple

Mars : $1 + 1 = 2$ couples

Avril : $2 + 1 = 3$ couples

Mai : $3 + 2 = 5$ couples

Juin : $5 + 3 = 8$ couples

Juillet : $8 + 5 = 13$ couples

Août : $13 + 8 = 21$ couples

Septembre : $21 + 13 = 34$ couples

Octobre : $34 + 21 = 55$ couples

Novembre : $55 + 34 = 89$ couples

Décembre : $89 + 55 = 144$ couple

Janvier : $144 + 89 = 233$ couples

.....

On note $F(n)$ le nombre de couples de lapins au mois n .

$F(n)$ = nombre de couples au mois $(n-1)$ + nombre de couples nés au mois $(n-2)$

Donc on peut exprimer $F(n)$ par la relation suivante :

1.1 Question.1 / Lire attentivement le code ci-joint, suivre les indications puis compléter la classe Fibo.java avec 3 méthodes itératives différentes permettant de calculer le n ième terme de la suite. Pour le stockage des termes.

1.1.1 1. La première méthode doit utiliser un tableau de taille $n+1$.

```
static BigInteger methode1(int n) {  
    if (n < 2) {  
        return BigInteger.valueOf(n);  
    } else  
    {  
        BigInteger[] fib = new BigInteger[n+1];  
  
        fib[0] = BigInteger.valueOf(0);  
        fib[1] = BigInteger.valueOf(1);  
  
        for (int i = 2; i < n+1 ; i++) {  
            fib[i] = fib[i - 1].add(fib[i - 2]);  
        }  
        return fib[n];  
    }  
}
```

1.1.2 2. La deuxième méthode doit utiliser 2 variables.

```
static BigInteger methode2(int n)
{
    if (n < 2) {
        return BigInteger.valueOf(n);
    } else
    {
        BigInteger a = BigInteger.valueOf(0);
        BigInteger b = BigInteger.valueOf(1);
        n = n - 2;
        while (n > 0) {

            b = a.add(b);
            a = b.subtract(a);
            n--;

        }

        return (a.add(b));
    }
}
```

1.1.3 3. La troisième méthode doit utiliser un tableau de taille 2.

```
static BigInteger methode3(int n) {

    if (n < 2) {
        return BigInteger.valueOf(n);
    } else
    {
        BigInteger[] fib = new BigInteger[2];
        fib[0] = BigInteger.valueOf(0);
        fib[1] = BigInteger.valueOf(1);
        n = n - 2;
        while (n > 0) {

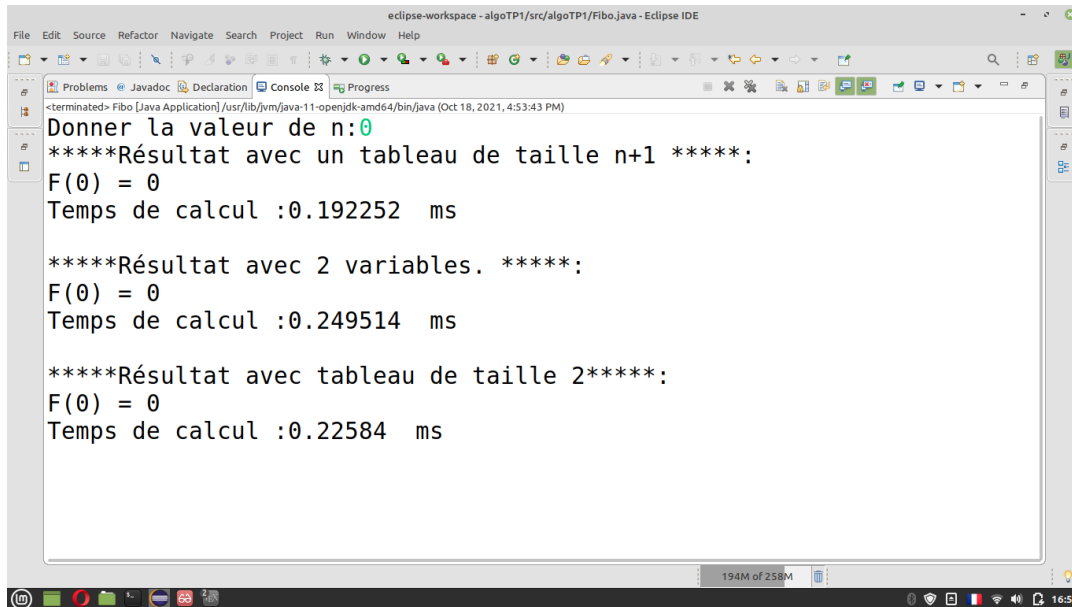
            fib[1] = fib[0].add(fib[1]);
            fib[0] = fib[1].subtract(fib[0]);
            n--;

        }
        return (fib[0].add(fib[1]));
    }
}
```

1.2 Question.2 / Exécuter le programme pour différentes valeurs de $n=0,1,\dots, 10, \dots, 80,\dots 100,\dots$

1.2.1 $N = 0$

$N = 0$ and Eclipse IDE



The screenshot shows the Eclipse IDE interface with the console window open. The console output is as follows:

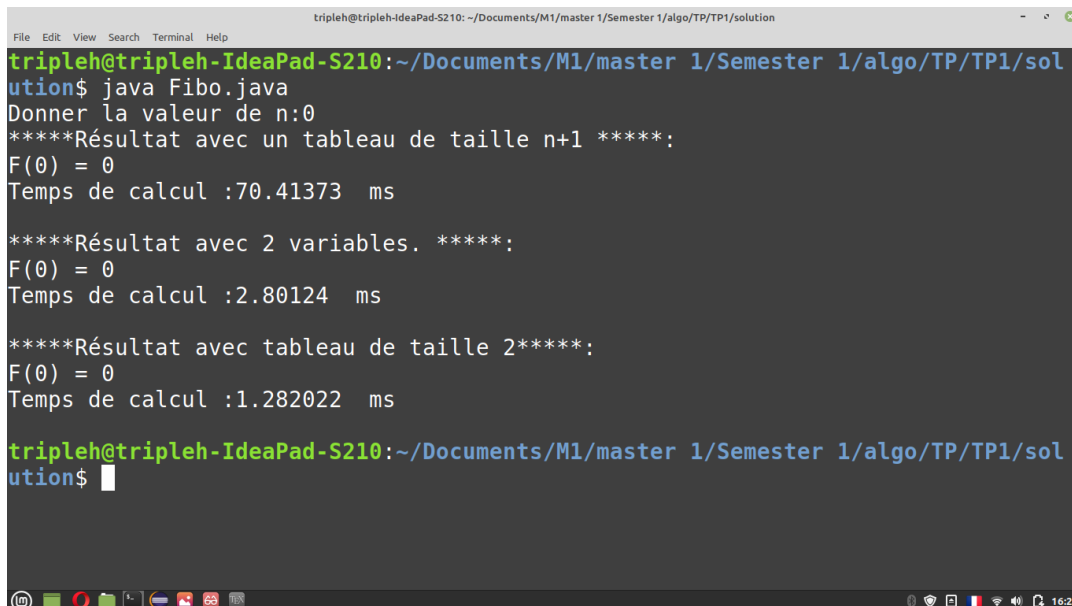
```
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 18, 2021, 4:53:43 PM)
Donner la valeur de n:0
*****Résultat avec un tableau de taille n+1 *****:
F(0) = 0
Temps de calcul :0.192252 ms

*****Résultat avec 2 variables. *****:
F(0) = 0
Temps de calcul :0.249514 ms

*****Résultat avec tableau de taille 2*****:
F(0) = 0
Temps de calcul :0.22584 ms
```

FIGURE 1.1: Value of $n=0$ using eclipse IDE

$N = 0$ and Terminal



The screenshot shows a terminal window with the following output:

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
$ java Fibo.java
Donner la valeur de n:0
*****Résultat avec un tableau de taille n+1 *****:
F(0) = 0
Temps de calcul :70.41373 ms

*****Résultat avec 2 variables. *****:
F(0) = 0
Temps de calcul :2.80124 ms

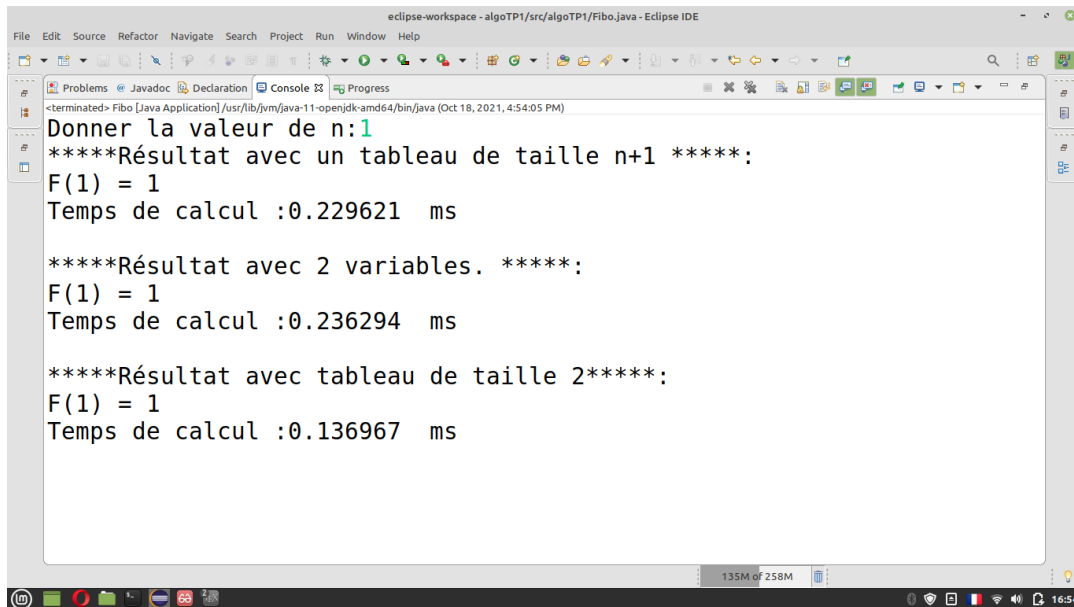
*****Résultat avec tableau de taille 2*****:
F(0) = 0
Temps de calcul :1.282022 ms

tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
$
```

FIGURE 1.2: Value of $n=0$ using Terminal

1.2.2 N = 1

N = 1 and Eclipse IDE



The screenshot shows the Eclipse IDE interface. The main editor displays the following code:

```
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 18, 2021, 4:54:05 PM)
Donner la valeur de n:1
*****Résultat avec un tableau de taille n+1 *****:
F(1) = 1
Temps de calcul :0.229621 ms

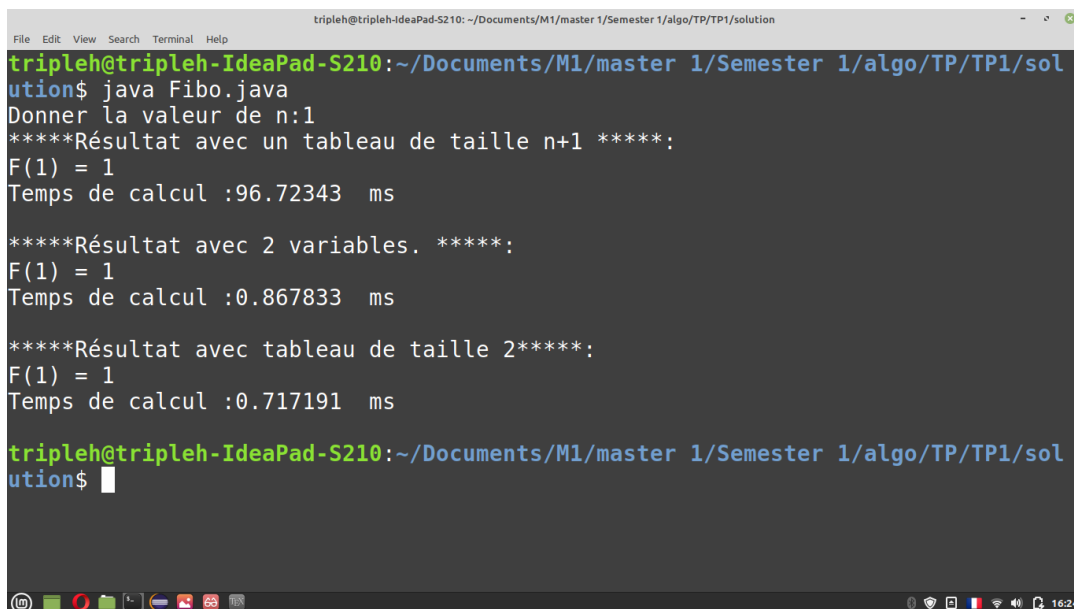
*****Résultat avec 2 variables. *****:
F(1) = 1
Temps de calcul :0.236294 ms

*****Résultat avec tableau de taille 2*****:
F(1) = 1
Temps de calcul :0.136967 ms
```

The console output matches the code. The status bar at the bottom indicates 135M of 258M memory usage.

FIGURE 1.3: Value of n=1 using eclipse IDE

N = 1 and Terminal



The screenshot shows a terminal window with the following output:

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$ java Fibo.java
Donner la valeur de n:1
*****Résultat avec un tableau de taille n+1 *****:
F(1) = 1
Temps de calcul :96.72343 ms

*****Résultat avec 2 variables. *****:
F(1) = 1
Temps de calcul :0.867833 ms

*****Résultat avec tableau de taille 2*****:
F(1) = 1
Temps de calcul :0.717191 ms

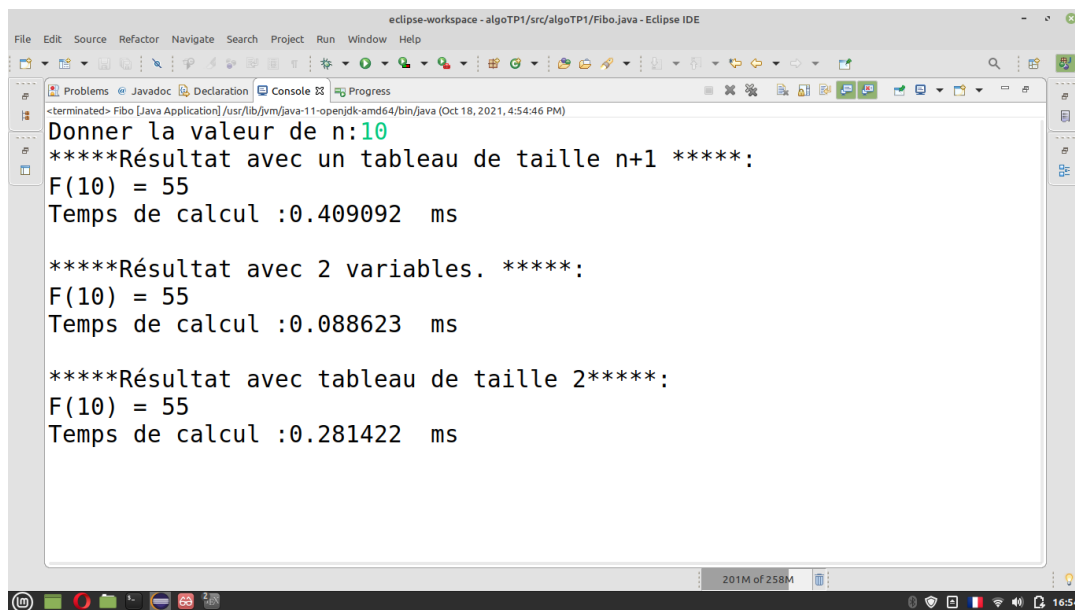
tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$
```

The terminal shows the execution of the Java program for N=1, with the same output as the Eclipse IDE screenshot.

FIGURE 1.4: Value of n=1 using Terminal

1.2.3 N = 10

N = 10 and Eclipse IDE



The screenshot shows the Eclipse IDE interface with the console window open. The console displays the output of a Java application named 'Fibo'. The output includes the prompt 'Donner la valeur de n:10', followed by three sets of results separated by separator lines '*****'. Each set shows the Fibonacci value F(10) = 55 and the execution time in milliseconds.

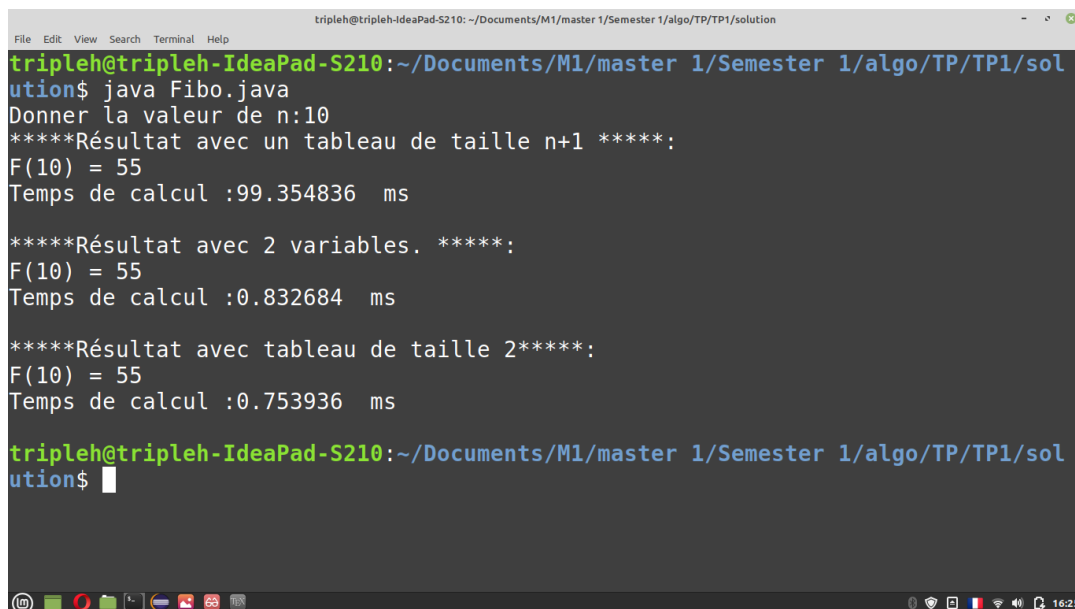
```
eclipse-workspace - algoTP1/src/algotP1/Fibo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 18, 2021, 4:54:46 PM)
Donner la valeur de n:10
*****Résultat avec un tableau de taille n+1 *****:
F(10) = 55
Temps de calcul :0.409092 ms

*****Résultat avec 2 variables. *****:
F(10) = 55
Temps de calcul :0.088623 ms

*****Résultat avec tableau de taille 2*****:
F(10) = 55
Temps de calcul :0.281422 ms
201M of 258M
16:54
```

FIGURE 1.5: Value of n=10 using eclipse IDE

N = 10 and Terminal



The screenshot shows a terminal window with the command 'java Fibo.java' executed. The output is identical to the Eclipse IDE screenshot, showing the prompt 'Donner la valeur de n:10' and three sets of results for F(10) = 55 with execution times.

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$ java Fibo.java
Donner la valeur de n:10
*****Résultat avec un tableau de taille n+1 *****:
F(10) = 55
Temps de calcul :99.354836 ms

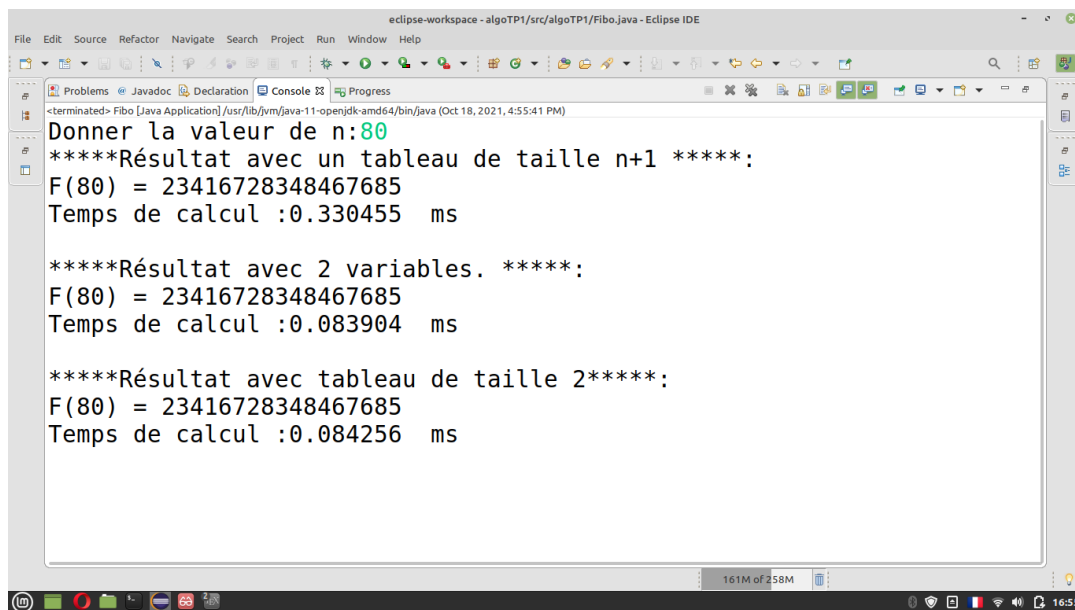
*****Résultat avec 2 variables. *****:
F(10) = 55
Temps de calcul :0.832684 ms

*****Résultat avec tableau de taille 2*****:
F(10) = 55
Temps de calcul :0.753936 ms
tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$
```

FIGURE 1.6: Value of n=10 using Terminal

1.2.4 N = 80

N = 80 and Eclipse IDE



The screenshot shows the Eclipse IDE interface with the console window open. The console output is as follows:

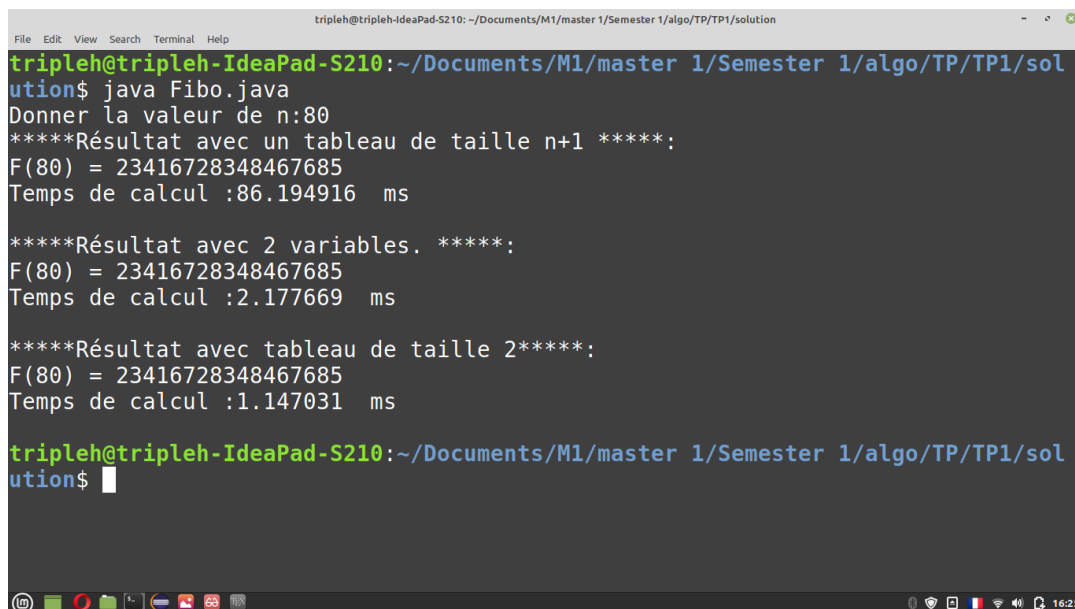
```
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 18, 2021, 4:55:41 PM)
Donner la valeur de n:80
*****Résultat avec un tableau de taille n+1 *****:
F(80) = 23416728348467685
Temps de calcul :0.330455 ms

*****Résultat avec 2 variables. *****:
F(80) = 23416728348467685
Temps de calcul :0.083904 ms

*****Résultat avec tableau de taille 2*****:
F(80) = 23416728348467685
Temps de calcul :0.084256 ms
```

FIGURE 1.7: Value of n=80 using eclipse IDE

N = 80 and Terminal



The screenshot shows a terminal window with the following output:

```
tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$ java Fibo.java
Donner la valeur de n:80
*****Résultat avec un tableau de taille n+1 *****:
F(80) = 23416728348467685
Temps de calcul :86.194916 ms

*****Résultat avec 2 variables. *****:
F(80) = 23416728348467685
Temps de calcul :2.177669 ms

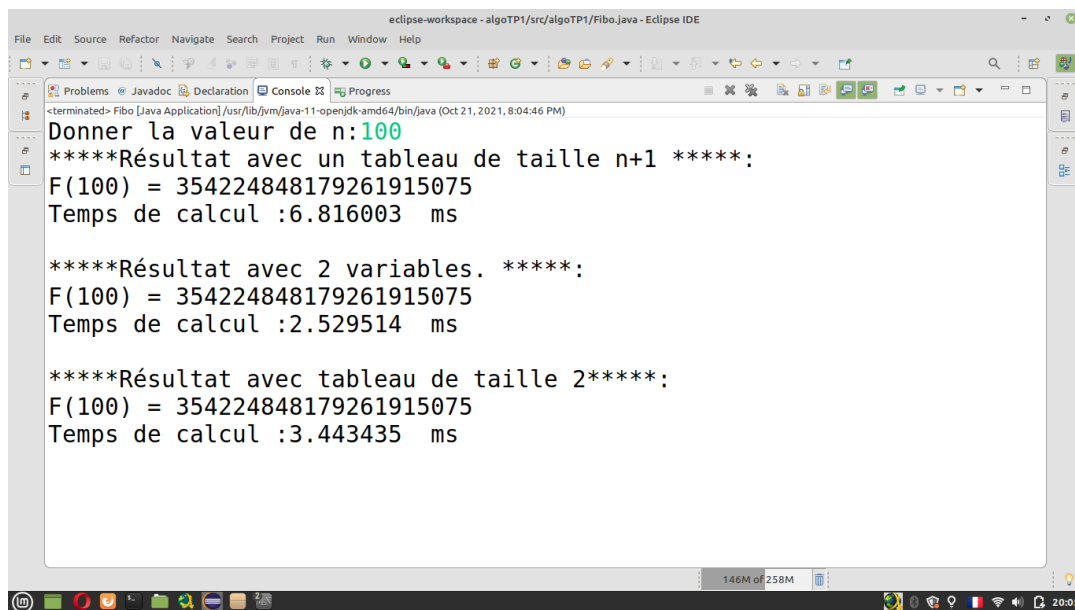
*****Résultat avec tableau de taille 2*****:
F(80) = 23416728348467685
Temps de calcul :1.147031 ms

tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/sol
ution$
```

FIGURE 1.8: Value of n=80 using Terminal

1.2.5 N = 100

N = 100 and Eclipse IDE



The screenshot shows the Eclipse IDE interface with the console window open. The console output is as follows:

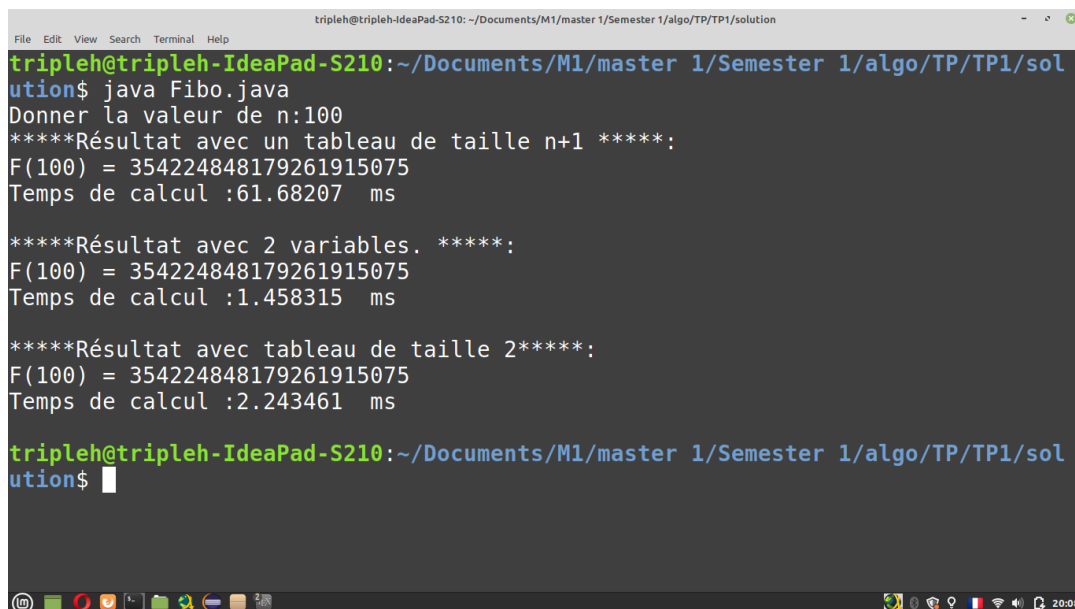
```
<terminated> Fib [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 21, 2021, 8:04:46 PM)
Donner la valeur de n:100
*****Résultat avec un tableau de taille n+1 *****:
F(100) = 354224848179261915075
Temps de calcul :6.816003 ms

*****Résultat avec 2 variables. *****:
F(100) = 354224848179261915075
Temps de calcul :2.529514 ms

*****Résultat avec tableau de taille 2*****:
F(100) = 354224848179261915075
Temps de calcul :3.443435 ms
```

FIGURE 1.9: Value of n=100 using eclipse IDE

N = 100 and Terminal



The screenshot shows a terminal window with the following output:

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$ java Fibo.java
Donner la valeur de n:100
*****Résultat avec un tableau de taille n+1 *****:
F(100) = 354224848179261915075
Temps de calcul :61.68207 ms

*****Résultat avec 2 variables. *****:
F(100) = 354224848179261915075
Temps de calcul :1.458315 ms

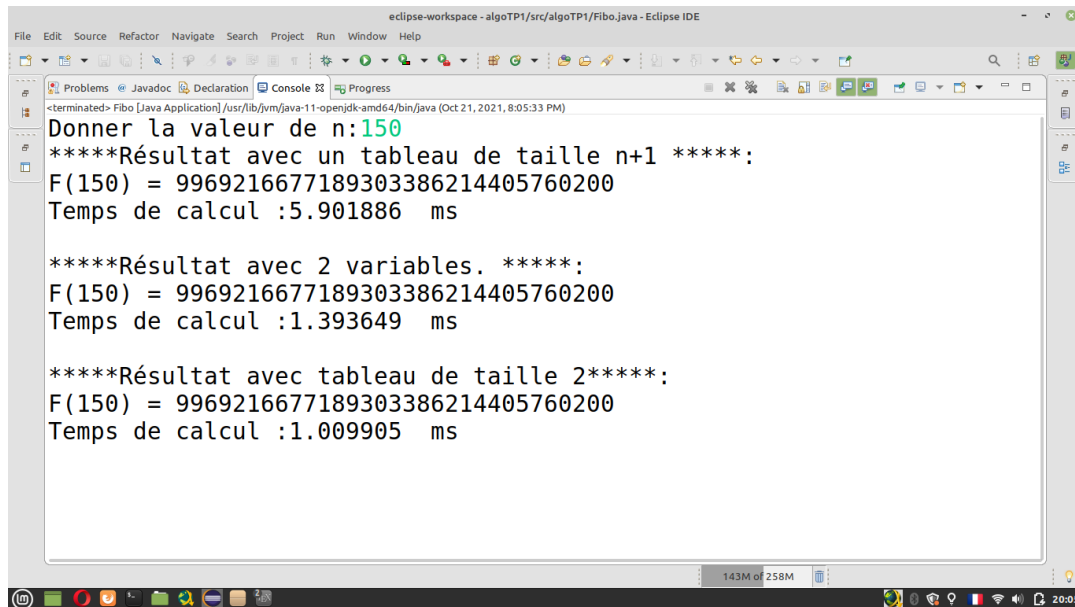
*****Résultat avec tableau de taille 2*****:
F(100) = 354224848179261915075
Temps de calcul :2.243461 ms

tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$
```

FIGURE 1.10: Value of n=100 using Terminal

1.2.6 N = 150

N = 150 and Eclipse IDE



The screenshot shows the Eclipse IDE interface with the console window open. The console output is as follows:

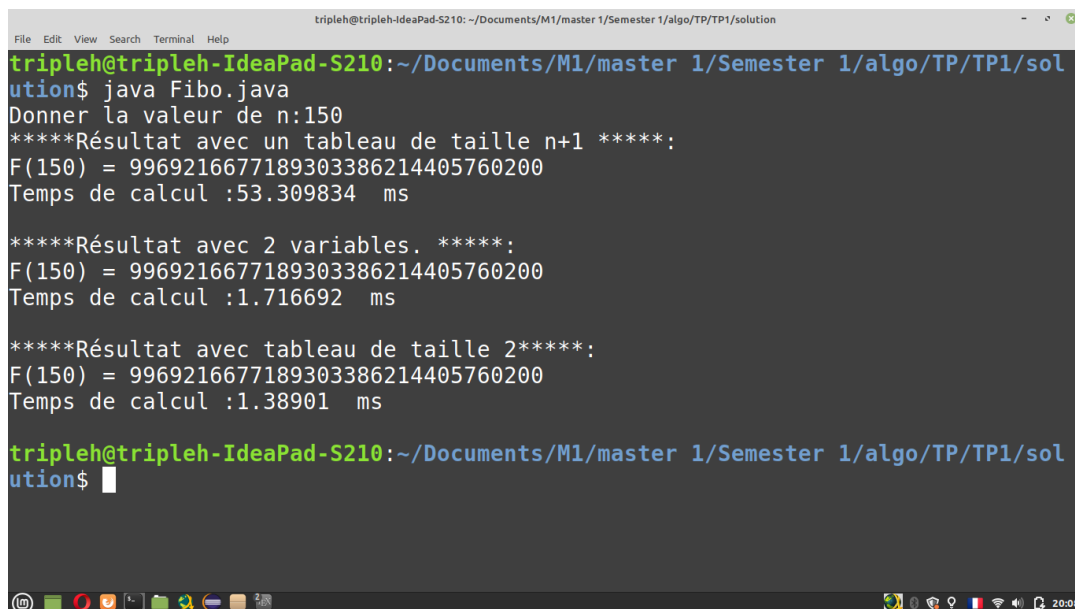
```
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 21, 2021, 8:05:33 PM)
Donner la valeur de n:150
*****Résultat avec un tableau de taille n+1 *****:
F(150) = 9969216677189303386214405760200
Temps de calcul :5.901886 ms

*****Résultat avec 2 variables. *****:
F(150) = 9969216677189303386214405760200
Temps de calcul :1.393649 ms

*****Résultat avec tableau de taille 2*****:
F(150) = 9969216677189303386214405760200
Temps de calcul :1.009905 ms
```

FIGURE 1.11: Value of n=150 using eclipse IDE

N = 150 and Terminal



The screenshot shows a terminal window with the following output:

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
$ java Fibo.java
Donner la valeur de n:150
*****Résultat avec un tableau de taille n+1 *****:
F(150) = 9969216677189303386214405760200
Temps de calcul :53.309834 ms

*****Résultat avec 2 variables. *****:
F(150) = 9969216677189303386214405760200
Temps de calcul :1.716692 ms

*****Résultat avec tableau de taille 2*****:
F(150) = 9969216677189303386214405760200
Temps de calcul :1.38901 ms

tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
$
```

FIGURE 1.12: Value of n=150 using Terminal

1.3 Question.3 / Donner la signification de chacune des 4 instructions suivantes :

1.3.1 `startTime = System.nanoTime();`

Here we are registering the time of our machine to the variable **startTime**, In order to note the **start** of the execution of our methodes **methode1()**,**methode2()**,**methode3()**.

System.nanoTime() is a system call that gives the current time in its most accurate way, when executed the OS will respond to Java VM with the time in nanoseconds.

The `java.lang.System.nanoTime()` method returns the current value of the most precise available system timer, in nanoseconds. The value returned represents nanoseconds since some fixed but arbitrary time (in the future, so values may be negative) and provides nanosecond precision, but not necessarily nanosecond accuracy.[2]

1.3.2 `System.out.println("F("+n+") = "+methode1(n));`

Here we are using the output method from system and the print line method from `system.io.printstream` to **print** on the **screen** the following **String F** and the **variable n** with the **execution of method methode1(n)**

Java `System.out.println()` is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood separately as[1]:

1. `System`: It is a final class defined in the `java.lang` package.[1]
2. `out`: This is an instance of `PrintStream` type, which is a public and static member field of the `System` class.[1]
3. `println()`: As all instances of `PrintStream` class have a public method `println()`, hence we can invoke the same on `out` as well. This is an upgraded version of `print()`. It prints any argument passed to it and adds a new line to the output. We can assume that `System.out` represents the Standard Output Stream.[1]

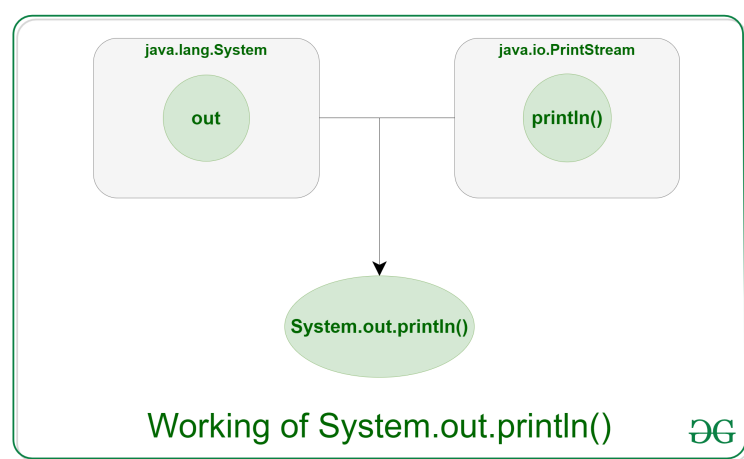


FIGURE 1.13: System.out.println in Java

1.3.3 endTime = System.nanoTime();

Now we are registering the current time from our **OS** to the variable **endTime**, this marks the end of our execution to calculate later the execution time of our method.

1.3.4 res= (float) (endTime-startTime) / 1000000 ;

Here we are **type casting** the result from **long** to **float** in the variable **res** the subtraction of the previously stored system times of **endTime** and **startTime** then we are dividing by 1000000 to get the time to **milliseconds**.

1.4 Question.4 / Remplir le tableau suivant :

	Methode 1	Methode 2	Methode 3
F(0)	0	0	0
F(5)	5	5	5
F(10)	55	55	55
F(50)	12586269025	12586269025	12586269025
F(60)	1548008755920	1548008755920	1548008755920
F(100)	354224848179261915075	354224848179261915075	354224848179261915075
Temps pour F(0)	0.236661 ms	0.206799 ms	0.244266 ms
Temps pour F(10)	0.214711 ms	0.092267 ms	0.100546 ms
Temps pour F(60)	0.289378 ms	0.190603 ms	0.185939 ms

1.5 Question.5 / Faire des captures d'écran de 3 exécutions de votre choix.

1.5.1 N = 0

N = 0 and Eclipse IDE

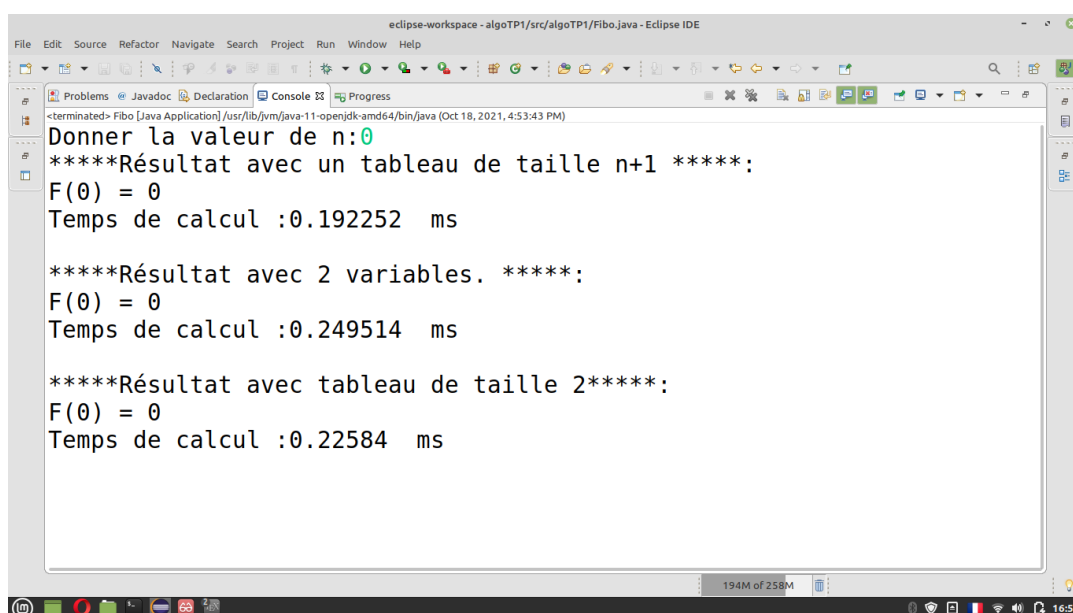
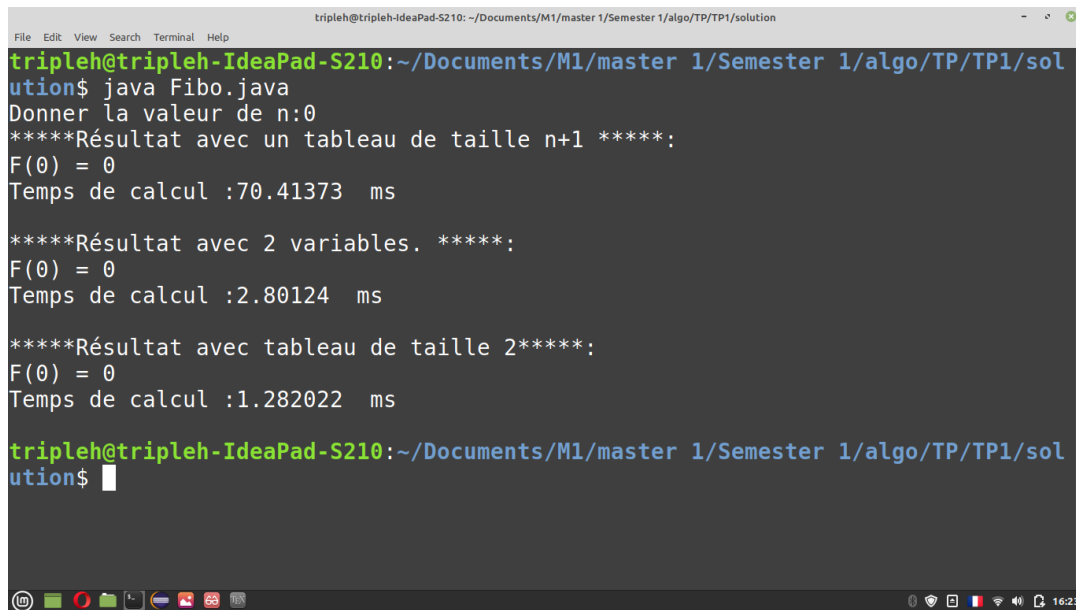


FIGURE 1.14: Value of n=0 using eclipse IDE

N = 0 and Terminal



```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$ java Fibo.java
Donner la valeur de n:0
*****Résultat avec un tableau de taille n+1 *****:
F(0) = 0
Temps de calcul :70.41373  ms

*****Résultat avec 2 variables. *****:
F(0) = 0
Temps de calcul :2.80124  ms

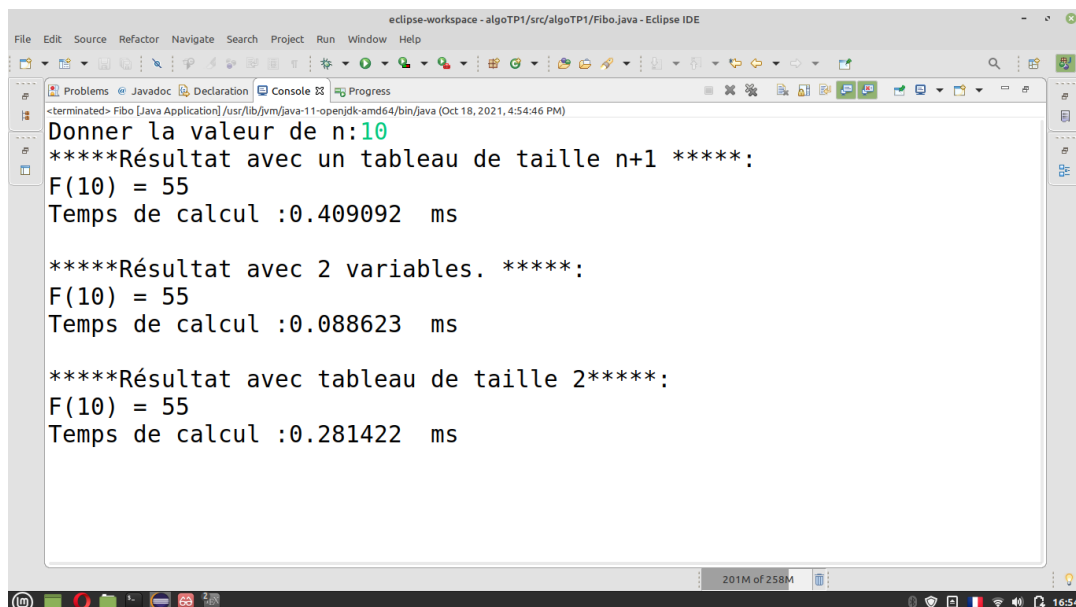
*****Résultat avec tableau de taille 2*****:
F(0) = 0
Temps de calcul :1.282022  ms

tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$
```

FIGURE 1.15: Value of n=0 using Terminal

1.5.2 N = 10

N = 10 and Eclipse IDE



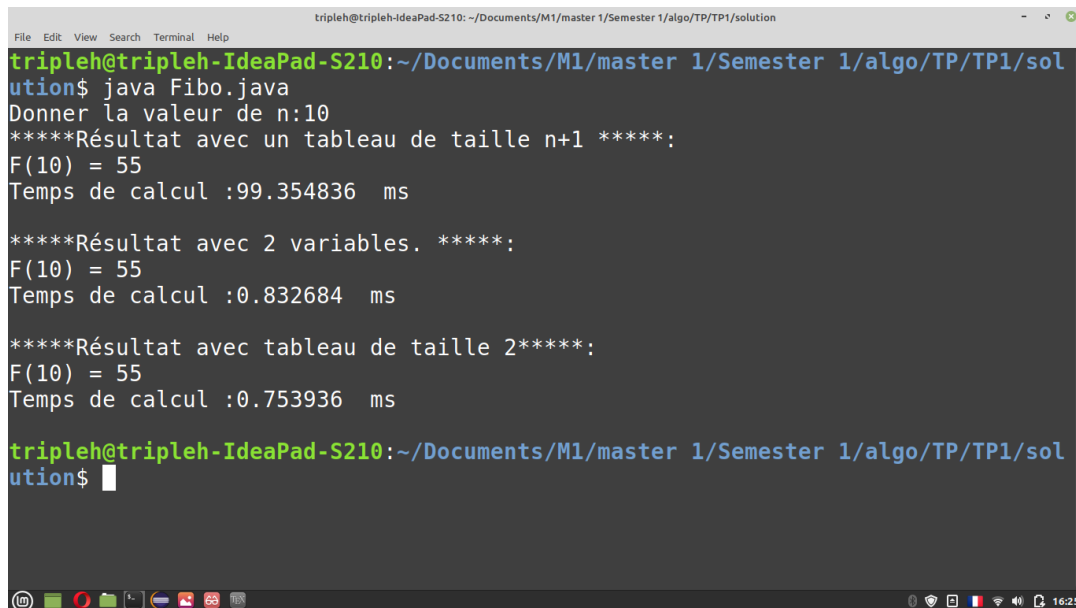
```
eclipse-workspace - algoTP1/src/TP1/Fibo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 18, 2021, 4:54:46 PM)
Donner la valeur de n:10
*****Résultat avec un tableau de taille n+1 *****:
F(10) = 55
Temps de calcul :0.409092  ms

*****Résultat avec 2 variables. *****:
F(10) = 55
Temps de calcul :0.088623  ms

*****Résultat avec tableau de taille 2*****:
F(10) = 55
Temps de calcul :0.281422  ms
```

FIGURE 1.16: Value of n=10 using eclipse IDE

N = 10 and Terminal



```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$ java Fibo.java
Donner la valeur de n:10
*****Résultat avec un tableau de taille n+1 *****:
F(10) = 55
Temps de calcul :99.354836 ms

*****Résultat avec 2 variables. *****:
F(10) = 55
Temps de calcul :0.832684 ms

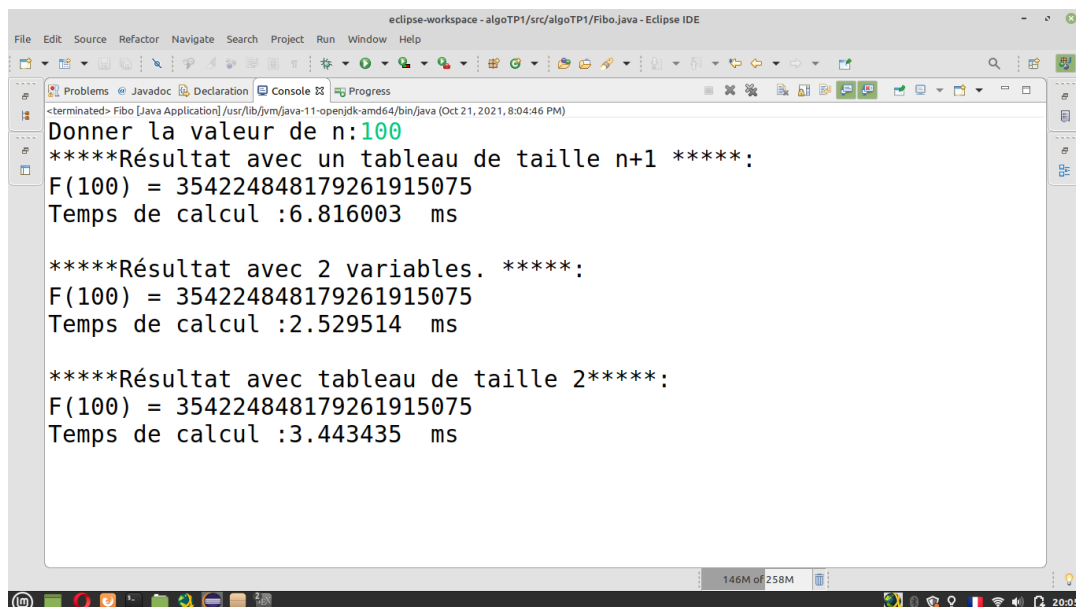
*****Résultat avec tableau de taille 2*****:
F(10) = 55
Temps de calcul :0.753936 ms

tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$
```

FIGURE 1.17: Value of n=10 using Terminal

1.5.3 N = 100

N = 100 and Eclipse IDE

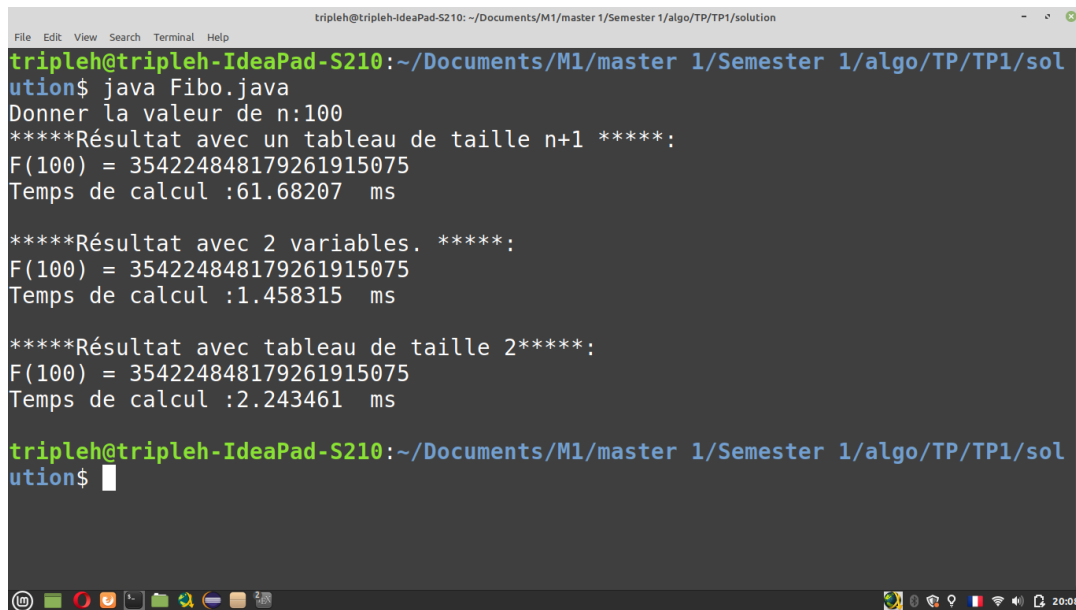


```
eclipse-workspace - algoTP1/src/TP1/Fibo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> Fibo [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Oct 21, 2021, 8:04:46 PM)
Donner la valeur de n:100
*****Résultat avec un tableau de taille n+1 *****:
F(100) = 354224848179261915075
Temps de calcul :6.816003 ms

*****Résultat avec 2 variables. *****:
F(100) = 354224848179261915075
Temps de calcul :2.529514 ms

*****Résultat avec tableau de taille 2*****:
F(100) = 354224848179261915075
Temps de calcul :3.443435 ms
```

FIGURE 1.18: Value of n=100 using eclipse IDE

N = 100 and Terminal

```
tripleh@tripleh-IdeaPad-S210: ~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution
tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$ java Fibo.java
Donner la valeur de n:100
*****Résultat avec un tableau de taille n+1 *****:
F(100) = 354224848179261915075
Temps de calcul :61.68207 ms

*****Résultat avec 2 variables. *****:
F(100) = 354224848179261915075
Temps de calcul :1.458315 ms

*****Résultat avec tableau de taille 2*****:
F(100) = 354224848179261915075
Temps de calcul :2.243461 ms

tripleh@tripleh-IdeaPad-S210:~/Documents/M1/master 1/Semester 1/algo/TP/TP1/solution$
```

FIGURE 1.19: Value of n=100 using Terminal

1.6 Question.6 / Calculer les complexités temporelle et spatiale de chacune des 3 méthodes utilisées. Que peut-on conclure ?

1.6.1 1. La première méthode doit utiliser un tableau de taille $n+1$.

```
static BigInteger method1(int n) {  
  
    if (n < 2) {  
        return BigInteger.valueOf(n);  
    } else  
    {  
  
        BigInteger[] fib = new BigInteger[n+1];  
  
        fib[0] = BigInteger.valueOf(0);  
        fib[1] = BigInteger.valueOf(1);  
  
        for (int i = 2; i < n+1 ; i++) {  
            fib[i] = fib[i - 1].add(fib[i - 2]);  
        }  
  
        return fib[n];  
    }  
}
```

Time Complexity of Method 1 with Barometric operation

```
fib[i] = fib[i - 1].add(fib[i - 2]);
```

The **Time Complexity** of the following operation is $f(n)=n-2$ which makes the time complexity of method 1 is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Time Complexity of Method 1 with Frequency Count Method

```

static BigInteger method1(int n) {
if (n < 2) { _____ (1)
    return BigInteger.valueOf(n); _____ (1)
} else
{
    BigInteger[] fib = new BigInteger[n+1]; _____ (1)

    fib[0] = BigInteger.valueOf(0); _____ (1)
    fib[1] = BigInteger.valueOf(1); _____ (1)

    for (int i = 2; i < n+1 ; i++) { _____ (n-2)
        fib[i] = fib[i - 1].add(fib[i - 2]); — (n-1)
    }
    return fib[n]; _____ (1)
}
}

```

$f(n) = 2n + 3$

Therefore the **Time Complexity** is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Space Complexity of Method 1

The **Space Complexity** of **Method 1** is the following.

n		1
fib [n+1]		n+1
i		1
$S(n) =$		$n+3$

Therefore the **Space Complexity** is $\Theta(n)$.

1.6.2 2. La deuxième méthode doit utiliser 2 variables.

```
static BigInteger methode2(int n) {  
    if (n < 2) {  
        return BigInteger.valueOf(n);  
    } else  
    {  
        BigInteger a = BigInteger.valueOf(0);  
        BigInteger b = BigInteger.valueOf(1);  
        n = n - 2;  
        while (n > 0) {  
            b = a.add(b);  
            a = b.subtract(a);  
            n--;  
        }  
        return (a.add(b));  
    }  
}
```

Time Complexity of Method 2 with Barometric operation

We will be choosing the following operation as the Barometric operation.

```
b = a.add(b);
```

The **Time Complexity** of the following operation is **$f(n)=n-1$** which makes the time complexity of method 1 is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Time Complexity of Method 2 with Frequency Count Method

```

static BigInteger methode2(int n) {
    if (n < 2) { _____(1)
        return BigInteger.valueOf(n); _____(1)
    } else
    {
        BigInteger a = BigInteger.valueOf(0); _____(1)
        BigInteger b = BigInteger.valueOf(1); _____(1)
        n = n - 2; _____(1)
        while (n > 0) { _____(n-2)

            b = a.add(b); _____(n-1)
            a = b.subtract(a); _____(n-1)
            n--; _____(n-1)

        }

        return (a.add(b)); _____(1)
    }
}

```

$f(n) = 4n + 1$

Therefore the **Time Complexity** is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Space Complexity of Method 2

The **Space Complexity** of **Method 2** is the following.

n		1
a		1
b		1
$S(n) =$		3

Therefore the **Space Complexity** is $\Theta(1)$.

1.6.3 3. La troisième méthode doit utiliser un tableau de taille 2.

```

static BigInteger methode3(int n) {
    if (n < 2) {
        return BigInteger.valueOf(n);
    } else
    {

        BigInteger[] fib = new BigInteger[2];
        fib[0] = BigInteger.valueOf(0);
        fib[1] = BigInteger.valueOf(1);
        n = n - 2;
        while (n > 0) {

            fib[1] = fib[0].add(fib[1]);
            fib[0] = fib[1].subtract(fib[0]);
            n--;
        }
        return (fib[0].add(fib[1]));
    }
}

```

Time Complexity of Method 3 with Barometric operation

We will be choosing the following operation as the Barometric operation.

```
while (n > 0)
```

The **Time Complexity** of the following operation is **$f(n)=n-1$** which makes the time complexity of method 1 is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Time Complexity of Method 3 with Frequency Count Method

```

static BigInteger methode3(int n) {
    if (n < 2) { _____(1)
        return BigInteger.valueOf(n); _____(1)
    } else
    {
        BigInteger[] fib = new BigInteger[2]; -----(1)
        fib[0] = BigInteger.valueOf(0); -----(1)
        fib[1] = BigInteger.valueOf(1); -----(1)
        n = n - 2; _____(1)
        while (n > 0) { _____(n-1)
            fib[1] = fib[0].add(fib[1]); -----(n)
            fib[0] = fib[1].subtract(fib[0]); --(n)
            n--; _____(n)
        }
        return (fib[0].add(fib[1])); -----(1)
    }
}

```

$f(n) = 4n + 6$

Therefore the **Time Complexity** is $\Theta(n)$.

Best case is $\Omega(1)$

Worst case is $\mathcal{O}(n)$

Space Complexity of Method 3

The **Space Complexity** of **Method 1** is the following.

n	1
fib[2]	2
	$S(n) = 3$

Therefore the **Space Complexity** is $\Theta(1)$.

1.7 Conclusion

In the end we notice the following regarding the **execution time** of the three algorithms that they are very **similar** even when we increase the Fibonacci number calculated. It indicates that it is harder to determine the better algorithm because of many factors regarding **OS, CPU Architecture, Programming language** and other factors.

Another thing we have noticed is the execution of our **Fibo.java** inside the Eclipse IDE is different than the execution with the OS Terminal, we think it is more related to memory buffering feature in the IDE that is lacked and not available inside the Terminal.

We also noticed that although the 3 methods all have an average time complexity of $f(n)=\Theta(n)$. the space complexity is different as method 1 has $S(n)=\Theta(n)$ while method 2 and method 3 are more efficient at $S(n)=\Theta(1)$

Declaring variables in Java as **BigInteger** in our methods instead of **int** was needed as we reached higher numbers where we had to use **BigInteger** to get 2^{32} . this much better than **long** which uses **64bit** that our machine can actually use for calculating larger Fibonacci numbers.

Bibliography

- [1] geeksforgeeks. "System.out.println in Java". In: (2019). URL: <https://www.geeksforgeeks.org/system-out-println-in-java/>.
- [2] tutorialspoint. "Java.lang.System.nanoTime() Method". In: (). URL: https://www.tutorialspoint.com/java/lang/system_nanotime.htm.