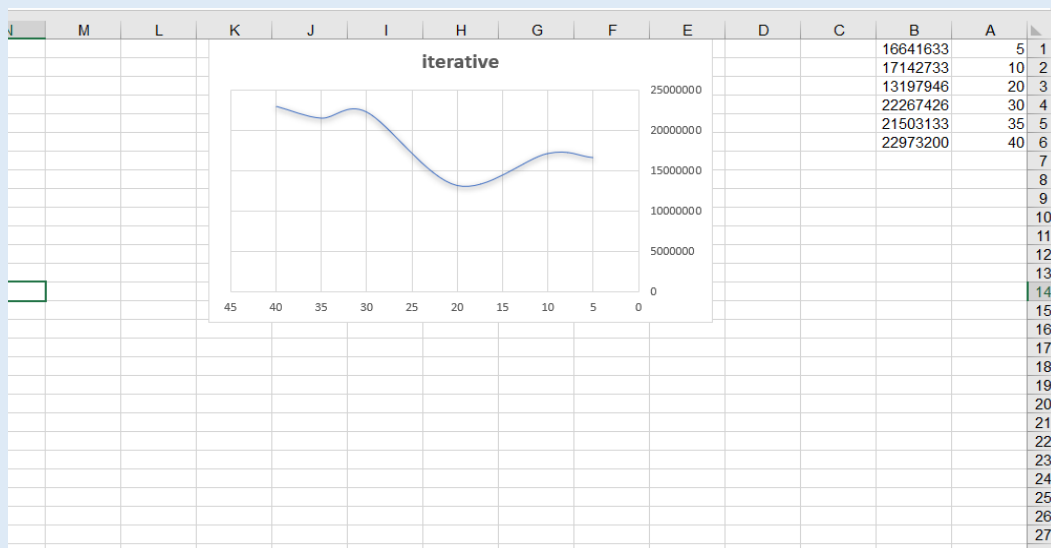
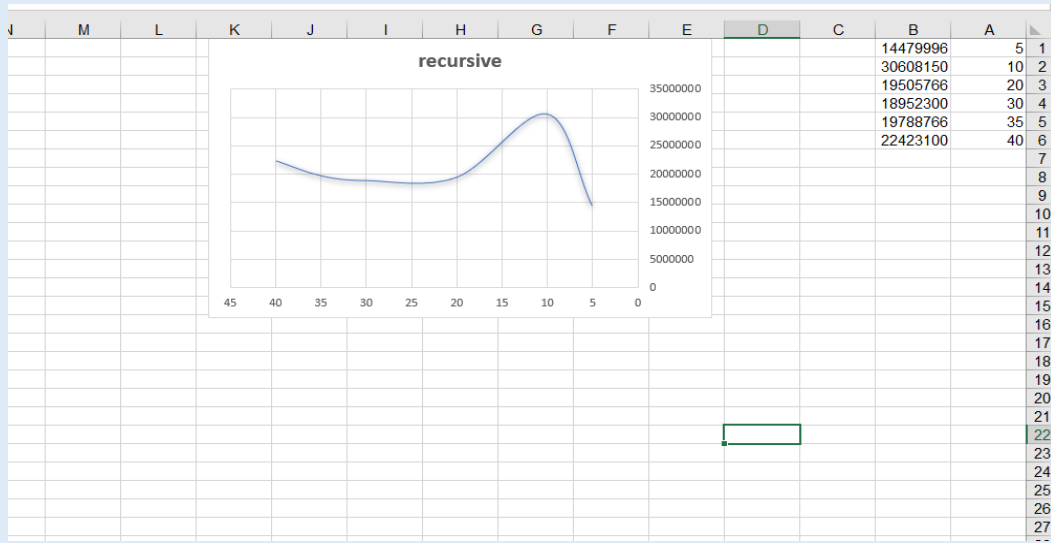


بسم الله الرحمن الرحيم

الاسم: حنين رسمي عواد

الرقم الجامعي: 202320815

طالبة زائرة من غزة في الجامعة العربية الامريكية



Conclusion:

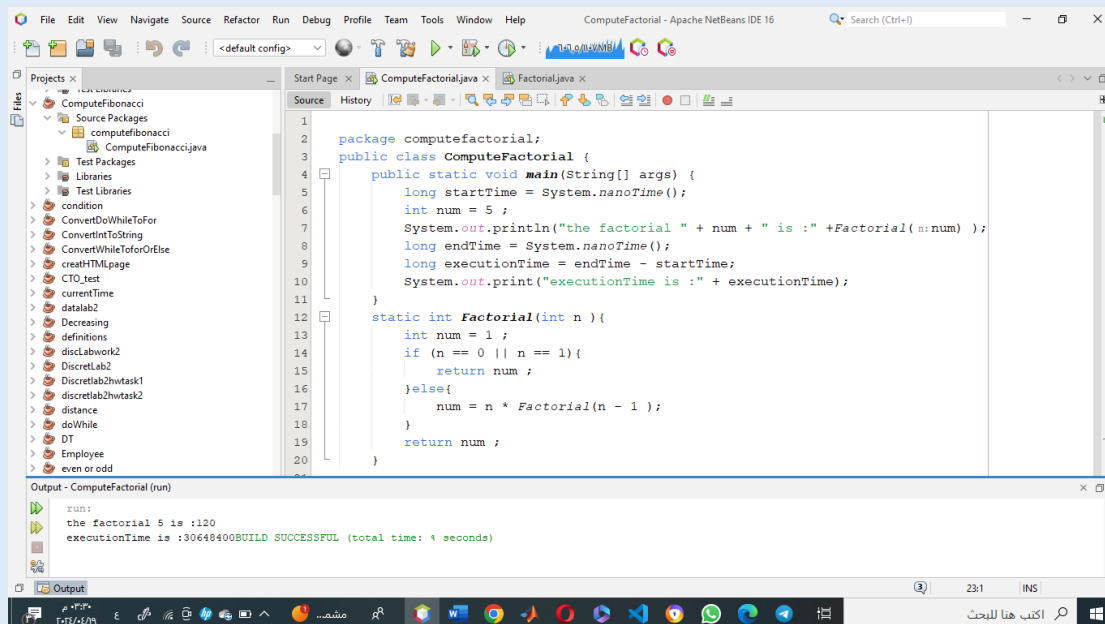
The two codes works correctly

Both have similar speeds at small numbers

iterative causes problems due to overflow

It causes a load on memory and slows down more

recursive



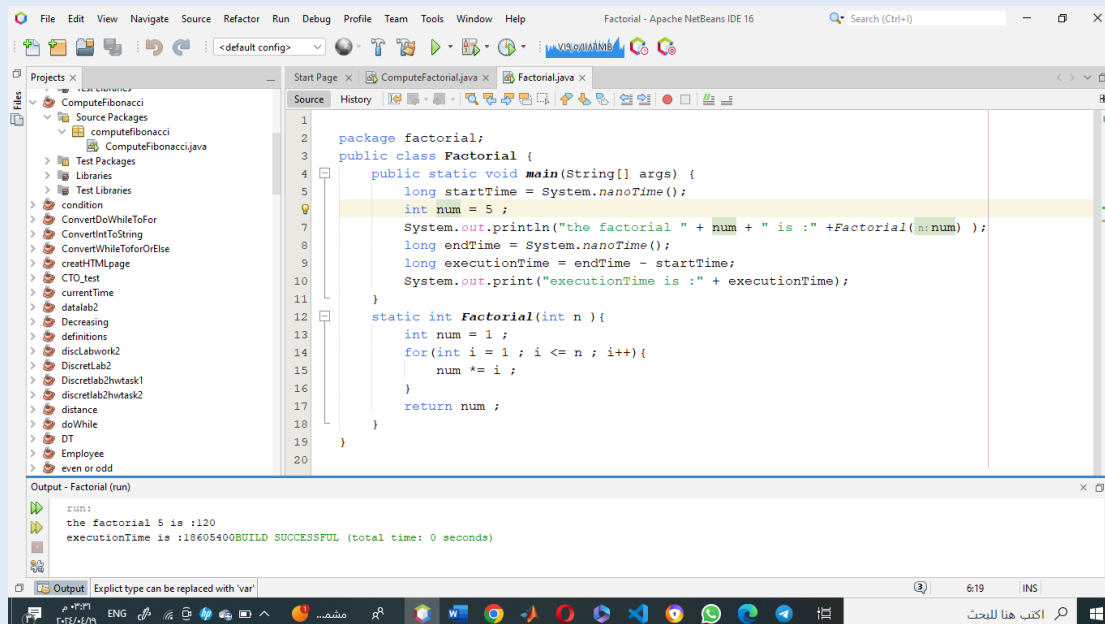
The screenshot shows the Apache NetBeans IDE interface. The main editor window displays the source code for a Java class named `ComputeFactorial`. The code implements a recursive factorial function. The `main` method initializes a number `num = 5`, records the start time, calls the `Factorial` method, records the end time, and prints the execution time. The `Factorial` method is a static recursive function that returns the factorial of the input number `n`. The output window at the bottom shows the results of running the program: "the factorial 5 is :120" and "executionTime is :1306484000BUILD SUCCESSFUL (total time: 4 seconds)".

```
1 package computeFactorial;
2
3 public class ComputeFactorial {
4     public static void main(String[] args) {
5         long startTime = System.nanoTime();
6         int num = 5 ;
7         System.out.println("the factorial " + num + " is :"+Factorial(n:num) );
8         long endTime = System.nanoTime();
9         long executionTime = endTime - startTime;
10        System.out.print("executionTime is :"+ executionTime);
11    }
12    static int Factorial(int n){
13        int num = 1 ;
14        if (n == 0 || n == 1){
15            return num ;
16        }else{
17            num = n * Factorial(n - 1 );
18        }
19        return num ;
20    }
21 }
```

Output - ComputeFactorial (run)

```
run:
the factorial 5 is :120
executionTime is :1306484000BUILD SUCCESSFUL (total time: 4 seconds)
```

Iterative:



The screenshot shows the Apache NetBeans IDE with a project named 'Factorial'. The main editor displays the source code for 'Factorial.java'. The code defines a package 'factorial' and a public class 'Factorial'. Inside the class, there is a static void method 'main' that takes a String array 'args'. It initializes 'long startTime = System.nanoTime()', sets 'int num = 5', and prints 'the factorial 5 is :120'. It then calculates 'long endTime = System.nanoTime()', 'long executionTime = endTime - startTime', and prints 'executionTime is :'. The output window shows the successful execution of the program, displaying 'run: the factorial 5 is :120' and 'executionTime is :18605400BUILD SUCCESSFUL (total time: 0 seconds)'. The bottom status bar indicates the time is 6:19 and the location is INS.

```
1 package factorial;
2
3 public class Factorial {
4     public static void main(String[] args) {
5         long startTime = System.nanoTime();
6         int num = 5 ;
7         System.out.println("the factorial " + num + " is :"+Factorial(n: num) );
8         long endTime = System.nanoTime();
9         long executionTime = endTime - startTime;
10        System.out.print("executionTime is : " + executionTime);
11    }
12
13    static int Factorial(int n ){
14        int num = 1 ;
15        for(int i = 1 ; i <= n ; i++){
16            num *= i ;
17        }
18        return num ;
19    }
20 }
```

run:
the factorial 5 is :120
executionTime is :18605400BUILD SUCCESSFUL (total time: 0 seconds)

