Documentation:

Documentation: 1. Research of how Fibonacci works and gathered all information from reading and videos. • Fibonacci numbers are sequence of integers where the first two elements are 0 and 1. • Fibonacci sequence will have the property that a number is the addition of its two predecessors and the sequence will be computed by repetitive calculation. • The N Fibonacci number will out put the function. N <=1 N ==0 N-1 + N-2 • This will apply to both functions recursive and iterative. 2. Create algorithms for recursive and iterative: If N <=1) return N Int I, one =0 tow =1, three; If (N==0) Return one; … 3. Test algorithms and fix any errors. 4. Open GIT Bash • Create folders (desk top folders) – add java files. 5. Open GIT hub repository • Transfer files from GIT Bash to GIT hub repository.

Graphical user interface, text, application, email

Description automatically generated

//Iterative

**package** Fibonacci;

**public** **class** FibonacciFunction {

**public** **int** fibonacciIterative(**int** n) {

**if**(n <= 1) {

**return** n;

}

**int** i, one = 0, two = 1, three;

**if** (n == 0)

**return** one;

**for** (i = 2; i <= n; i++)

{

three = one + two;

one = two;

two = three;

}

**return** two;

}

**public** **int** fibonacciIterative(**int** n) {

**if**(n <= 1) {

**return** n;

}

**return** fibonacciIterative(n-1)+(n-2);

}

**public** **static** **void** main(String args[]) {

**for**(**int** i=1;i<=10;i++){

FibonacciFunction fib = **new** FibonacciFunction();

**long** startTime = System.*nanoTime*();

**int** fibnum = *fib*(i);

**long** endTime = System.*nanoTime*();

**long** programruntime = endTime - startTime;

System.***out***.println(i + " " + programruntime);

}

}

**private** **static** **int** fib(**int** i) {

**return** 0;

}

}

Graphical user interface, text, application, email

Description automatically generated

//Recursive

**package** Fibonacci;

**public** **class** FibonacciFunction {

**public** **int** fibonacciRecursive(**int** n) {

**if**(n <= 1) {

**return** n;

}

**int** i, one = 0, two = 1, three;

**if** (n == 0)

**return** one;

**for** (i = 2; i <= n; i++)

{

three = one + two;

one = two;

two = three;

}

**return** two;

}

**public** **int** fibonacciRecursive(**int** n) {

**if**(n <= 1) {

**return** n;

}

**return** fibonacciRecursive(n-1) + (n-2);

}

**public** **static** **void** main(String args[]) {

**for**(**int** i=1;i<=10;i++){

FibonacciFunction fib = **new** FibonacciFunction();

**long** startTime = System.*nanoTime*();

**int** fibnum = *fib*(i);

**long** endTime = System.*nanoTime*();

**long** programruntime = endTime - startTime;

System.***out***.println(i + " " + programruntime);

}

}

**private** **static** **int** fib(**int** i) {

**return** 0;

}

}

|  |  |  |
| --- | --- | --- |
| Fibonacci Numbers | Recursion | Iterative |
| 1 | 1500 | 700 |
| 2 | 500 | 200 |
| 3 | 200 | 200 |
| 4 | 100 | 100 |
| 5 | 100 | 100 |
| 6 | 200 | 100 |
| 7 | 500 | 100 |
| 8 | 200 | 200 |
| 9 | 300 | 100 |
| 10 | 100 | 200 |