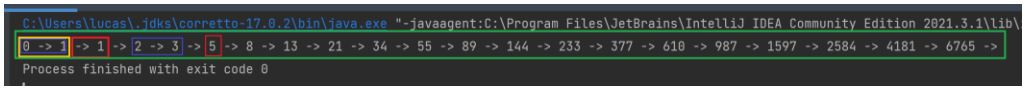
Sequência do Fibonacci

A **sequência de Fibonacci** é um desafio manjado da programação (obviamente essa sequência existe antes da programação, mas isso é outra parada). Basicamente **essa sequência** é formada por **termos que são a soma** dos **dois termos anteriores** ***(Os dois primeiros termos sempre são 0 e 1):***



Resolvendo com while

A gente sabe que o **primeiro** e **segundo** termo no começo sempre vão ser **0** e **1 respectivamente**, consequentemente temos o **valor do terceiro termo** (soma entre os dois primeiros):

```java
long primeiroTermo = 0;
long segundoTermo = 1;
long produto = primeiroTermo + segundoTermo ;
```

A chave para fazer o Fibonacci funcionar é **mover os termos a cada iteração.** Depois que a primeira iteração é concluída ***(dois termos formando um terceiro)*** é preciso fazer com que o **primeiro termo se torne o segundo**, **o segundo termo se torne o produto** e o **produto se torne o primeiro termo + o segundo:**

```java
while(contador <= n){

    System.out.print(produto + " -> ");

    primeiroTermo = segundoTermo;
    segundoTermo = produto;
    produto = primeiroTermo + segundoTermo;

    contador ++;
}
```

```
T1   T2     PRODUTO
0 -> 1 -> 1

     T1   T2     PRODUTO
0 -> 1 -> 1 -> 2

          T1   T2     PRODUTO
0 -> 1 -> 1 -> 2 -> 3
```

Resolvendo com recursividade

A formula do Fibonacci sempre tem o **produto** como sendo a **soma dos dois termos anteriores a ele**. Então a gente sabe que: **(termo – 1)** + **(termo – 2)** = **produto.** *(se o termo for menor que 2 então ele obrigatoriamente vai retornar 1 que é o começo da sequência):*

```java
public static int fiboComRecursividade(int termo){

    if(termo < 2){
        return 1;
    }

    return fiboComRecursividade( termo: termo - 1 ) + fiboComRecursividade( termo: termo - 2);
}
```

Daí é só iterar com um for chamando a função recursiva, cada iteração gera um termo.