



## Definition

- Die Fibonacci-Sequenz ist eine Folge von Zahlen, wobei jede Zahl die Summe der beiden vorhergehenden ist, normalerweise beginnend mit 0 und 1. Die Sequenz lautet wie folgt:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17,

[Link to Wikipedia:](#)

**Es ist iterativ definiert als:**

$$\text{Fibonacci}(n) = \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2)$$

## FiboPrimeNew.java

Dieses Java-Programm berechnet die Fibonacci-Sequenz bis zu einer bestimmten Größe und prüft, ob jede Zahl in der Sequenz eine Primzahl ist. Der in diesem Code verwendete Prime-Checking-Algorithmus ist nicht der effizienteste, funktioniert jedoch für kleinere Fibonacci-Nummern.

Hier ist eine Aufschlüsselung des Codes:

- Wir importieren die erforderlichen Bibliotheken
  - `math.BigDecimal` (Operationen)
  - `math.RoundingMode` (Operationen).

```
import java.math.BigDecimal;
import java.math.RoundingMode;
```

- Wir definieren die Hauptklasse FiboPrimeNew und die `main` Methode.

- Wir geben ein Argument an die Größe der Fibonacci-Sequenz,
- Die Sequenz berechnet Fibonacci-Sequenz und prüfen ob diese eine Primzahl enthält.
- Wir lassen uns anzeigen wie lange das Programm braucht um es auszuführen **Ausführungszeit**.  
Wenn mehr als ein Argument oder gar kein Argumente angegeben wurde, dann zeigt die Funktion **showHelp()**.

## private static boolean isPrime(BigDecimal fibonacci) Methode

- Wir überprüfen, ob das Argument(Eingabe) eine Primzahl ist.
- Mit der Schleife teilen wir durch 2 bis der Hälfte der Zahl.
- Wenn die Zahl durch einen seiner Teiler - teilbar ist, dann ist sie keine Primzahl.
- Wenn die Schleife ohne einen Divisor zu finden durchläuft, dann ist die eingegebene Zahl eine Primzahl.

```
private static boolean isPrime(BigDecimal fibonacci) {

    BigDecimal divisor = BigDecimal.valueOf(2);

    BigDecimal max = new BigDecimal(
        String.valueOf(fibonacci.divide(
            BigDecimal.valueOf(2), RoundingMode.DOWN
        )
    )
    );

    while(divisor.compareTo(max) < 0) {

        BigDecimal quotient = fibonacci.divide(divisor, RoundingMode.UP);

        BigDecimal quotientRounded = fibonacci.divide(
            divisor,
            RoundingMode.DOWN
        );
        boolean isDivisible = quotient.compareTo(quotientRounded) == 0;
        if (isDivisible) {
            return false;
        }
        divisor = divisor.add(
            BigDecimal.valueOf(1)
        );
        //NOTE divisor++ funktioniert hier nicht deswegen valueOf(1)
    }
    return true;
}
```

## countFibonacci(...) Methode

- zur Berechnung der Fibonacci-Sequenz bis zur angegebenen Größe.
- wir initialisieren die ersten beiden Fibonacci-Zahlen (0 und 1).
- wir berechnen für jede Zahl in der Sequenz die nächste Fibonacci-Zahl, indem beide vorherigen Zahlen hinzugefügt werden.

- Wir überprüfen, ob die neue Fibonacci-Nummer eine Primzahl ist und zeigen Sie als Ergebnis an.

```
private static void countFibonacci(int size, BigDecimal[] fibonacci) {
    fibonacci[0] = new BigDecimal("1"); //NOTE #1
    System.out.printf("#%d%s : %s\n", 1, checkNumeral(1), fibonacci[0]);
    if (size > 1) {
        fibonacci[1] = new BigDecimal("1"); //NOTE #2
        System.out.printf("#%d%s : %s\n", 2, checkNumeral(2), fibonacci[1]);
    }
    for (int i = 2; i < size; i++) {
        boolean isPrime;

        fibonacci[i] = fibonacci[i - 2].add(fibonacci[i - 1]); //NOTE #n
        System.out.printf("#%d%s : %s\n", i + 1, checkNumeral(i +
1), fibonacci[i]);

        isPrime = isPrime(fibonacci[i]);
        if (isPrime) {
            System.out.printf(
                "#%d%s Fibonacci Number is a Prime Number\n",
                i + 1,
                checkNumeral(i + 1)
            );
        }
    }
}
```

## checkNumeral(int n) Methode

- Wir implementieren die checkNumeral Methode zur Bestimmung der korrekten Suffixe für die Fibonacci-Nummer.

```
private static String checkNumeral(int n) {
    switch (n) {
        case 1 -> {
            return "-st";
        }
        case 2 -> {
            return "-nd";
        }
        case 3 -> {
            return "-rd";
        }
        default -> {
            return "-th";
        }
    }
}
```

## showHelp() Methode

- Wir lassen uns den Hilfetext anzeigen, bei falscher Eingabe des Parameters.

```
public static void showHelp() {  
    System.out.println("n muss ein positiv integer sein!");  
    System.out.println("Starte das Programm neu mit Korrekten Parametern!");  
}
```