Introduction to Java

Special Care Set for Java Begineers/ Ex 2

・Notice!
The motivation of this page is to lead regular exercises.
We prepare many useful information in this page as hints of exercises, but this page has no answer.
If you have question, please don't hesitate to ask and discuss your teacher and teaching assistants.
このページは、通常の演習問題への導きとして作られています。
多くの有用な情報をヒントとして準備しておりますが、ここに回答が隠されているとかはありません。
もし、演習に対して質問等ございましたら、遠慮なく教員・TAに質問・議論をしてください。

Example Problem: Fibonacci

フィボナッチ数列は次のような式で与えられるものです。
Fibonacci Sequence is given as follows:

$$F_0 = 0\,,\, F_1 = 1\,,\, F_{n+2} = F_n + F_{n+1} \quad (n \geq 0)$$ ....**(1)**

一般項: $$F_n = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right\} = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}$$

($$\phi \equiv \frac{1+\sqrt{5}}{2} \simeq 1.618033988749895$$) ....**(2)**

この数列は, レオナルド・フィボナッチ氏によって与えられ, 今日様々な形でよく表現される特別な数列の1つとなっている.
This sequence is found by Dr. Leonardo of Pisa (Leonardo Fibonacci), and this is one of the most famous sequence today.

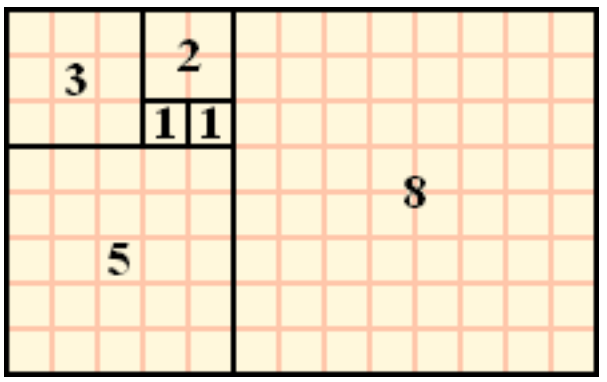
フィボナッチ数と呼ばれるこの数列の要素は,
Elements of Fibonacci number is as:

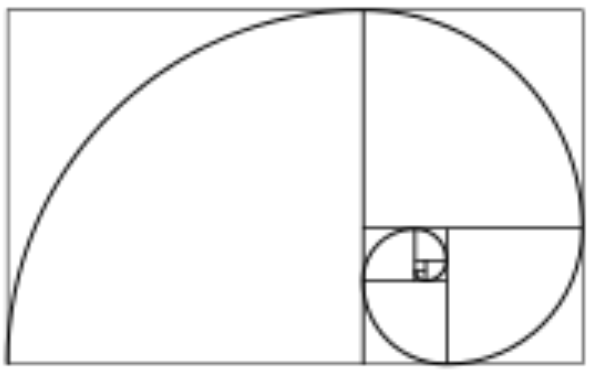
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
55, 89, 144, 233, 377, 610, 987, 1597, 2584 , 4181,
6765, 10946, 17711, 28657, 46368, 75025, 121393, 196418, 317811, 514229,
832040, 1346269, 2178309, 3524578, 5702887, 9227465, 14930352, 24157817, 39088169

という具合に増えていく.

なお, ***Φ***で与えられている数は黄金比と呼ばれる。
Φ is called golden ratio in the general in closed-form expression

これを、Javaのコードにしていくことを考えてみよう。
This time, let's think about coding in Java for this equation.



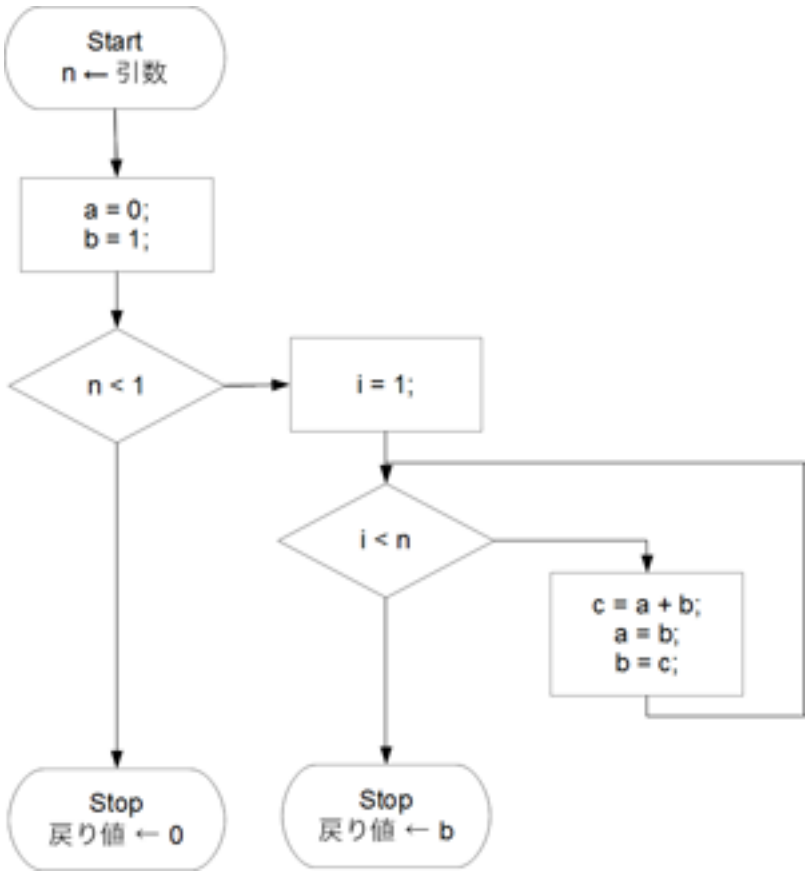


フィボナッチ数列の各項を一辺とする正方形を組み合わせると, 渦を巻くようになる(出典: Wikipedia)

Fibonacciのサンプルコードは[こちら]
The one of the sample code of Fibonacci is [Here].

Example 1: Flow Chart

ある数 n の時のフィボナッチ数 F(n) を計算するメソッド
int fibonacci (int n)
を考えましょう。
Let's try to think about fibonacci number calculation method.



これは, メインメソッドより呼び出されるものとするので, 使用例は次のような感じになります。
An example of usage for method fibonacci(n) is:

```
public class FibonacciDemo {
        public static void main(String[] args) {
                int a = fibonacci(10);
                System.out.println(a); // a = 55
        }
}
```

メソッドを書き始める前に、どのような手順で行うかを整理しましょう。
You should make plan how to construct this method.

- まず, 前述の漸化式(1)を見ると, 計算で与えられる左辺値が1つ (fn+1), 右辺値が2つ (fn, fn-1) があります。
  仮に, fn+1 を c, fn を b, fn-1 を a と置きましょう。
  First, we make f_n+1 as variable c, fn as variable b and fn-1 as variable a for equation (1).
- 次に, 初期値を考えます. n = 1をスタートとした場合,
  fn = f1 = 1, fn-1 = f0 = 0
  という具合に, それぞれの値を決定できますから,
  b = 1, a = 0, c=0
  としておきましょう。
  For initializing, valiable a, b and c makes a=0, b=1, c=0 for iterative calculation.
- そうすると, 自動的に fn+1として、 n=2の値は, 次のように計算できますね。
  fn+1 = c = b + a
  Next, c is able to calculate using equation (1), then, c = b + a.
- あとは, 次の計算であるn=3を求める際には、先ほどのn=2の値はfn (b)へ、n=1の値はfn-1 (a)へ移しておきますから、
  a = b, b = c
  という動作も含まれることになります。
  After calculating c, variable a and b should prepare to next calculation such as a = b and b = c.
- 最後に, n の回数まで上記の手続きを行った場合, nに対応する部分は, 変数bの値になりますから、b を戻り値とします。
  Finally, after iterative calculation until n times, the fibonacci number F(n) is explained by valiable

b, then the method return b.

この様な具合で、手続きをフローチャートとして記述していくと、右図のようになります。
Flowchart of this method is explained by right figure.

このフローチャートをソースコードとすると、次のような記述になります。
Also, source code is explained as below:

```
public class FibonacciDemo {
        public static void main(String[] args) {
    ...
        }
  public static int fibonacci(int n){
    int a = 0;
    int b = 1;
    int c = 0;
    if(n < 1) return 0;
    for(int i = 1; i < n; i++){
      c = a + b;
      a = b;
      b = c;
    }
    return b;
  }
}
```

Javaはこの様な運びで、アルゴリズムやソフトウエアを記述していくことになります。
This is one of the process how to coding Java language.

なお、fibonacciメソッドについている『static』とは、『静的な』という意味です。これは、オブジェクトを実体化させない（newさせない）状態でも、そのクラスの機能を使いたいという場合に扱うものです。
今回のスライドに、変数についてのstaticが出てきましたが、メソッドについても同様に扱うことが出来ます。
"static" method is able to access without instance. In this lecture, we already study about static variable, and static method is also same to use in Java.

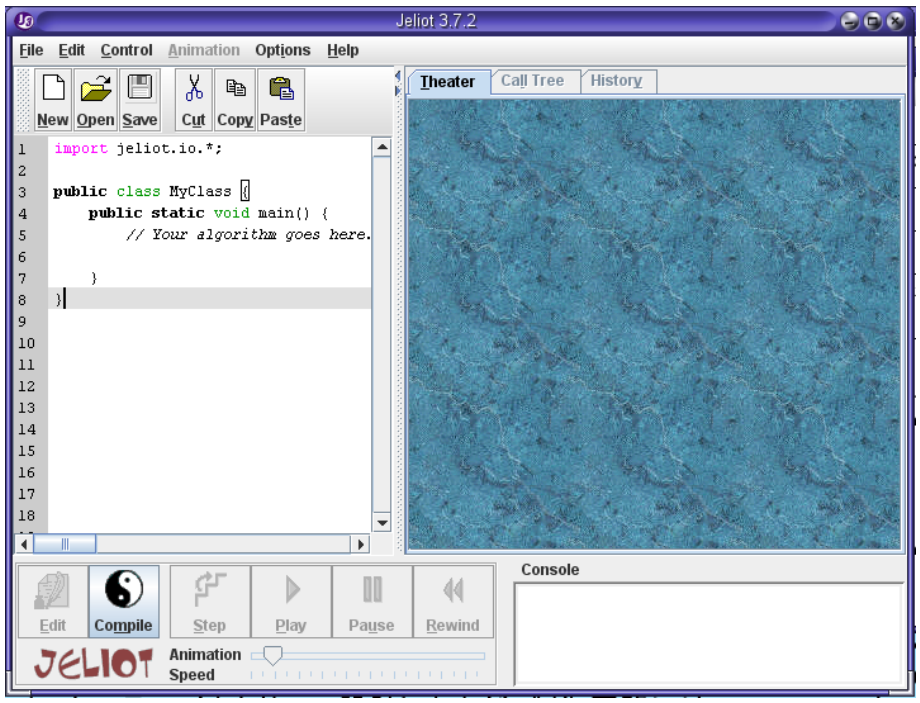
Example 2: Jeliotによる挙動の確認

Example 1では、アルゴリズムやソフトウエアの設計について、どのように考えたらいいかを説明しましたが、実際にその通りに動いているかを確認するためには、Jeliotを使用することで、内部の挙動を確認することが出来ます。
In example 1, we study about how to plan and code algorithm and software in Java, and we should check how to move written codes. Jeliot is an tool for checking action of the Java codes.

Jeliotを動かすには、次のコマンドを入力してください。
To run Jeliot type:

```
% java -jar /home/course/javaone/jeliot3/jeliot.jar
```

Jeliotを実際に動かして、その画面を確認してみましょう。
Please execute Jeliot, and check that interface.



左側のスペースにコードを挿入し, 下にあるコンパイルボタンを押すと、動作閲覧のための準備が整いますので、Playを押して右側に出てくるアニメーションを確認してください。
Please input codes on left space and push "compile" button, then it is able to prepare to explain animation.
Please push "play" button for seeing animation.