CS 1713 Fall 2011 Project 1:

Fibonacci Numbers

This is the first one of three major projects that you are required to complete in this class. You MUST design and code this project on your own. You can ask for debugging help, particularly from the tutors, TAs, or instructor, but this project is to be your own work. If you have a question about this project other than simple debugging, the only one who can help you is your instructor. The project relies on knowledge and skills that you will develop in Recitation Laboratories 1-3 and 6, and in the lectures. In particular, this project deals with simple arrays, loops, and conditionals. If you are having trouble with these recitations after your in-class session, please be sure that you come to the lab at another time and sit down with a tutor.

This project is due on October 9.

Overview
The Fibonacci numbers are a sequence of numbers, where the first two numbers are \( F_0 = 0 \) and \( F_1 = 1 \), and each remaining number is the sum of the previous two, i.e., \( F_n = F_{n-1} + F_{n-2} \). For example, the sequence of Fibonacci numbers of length 10 is the following sequence of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

Fibonacci numbers are used in computer algorithms such as the Fibonacci search technique and the Fibonacci heap data structure. (You can search on the Internet to find out more about them.)

In this project, you will write a program to generate Fibonacci numbers of various lengths. The purpose of this project is for you to gain experience working with simple arrays, loops, and conditionals. You must implementation your program according to the following description.

Part 1 (8 points)
In this part you will create two classes, the Fibonacci class and the FibonacciTester class. The Fibonacci class stores the Fibonacci numbers of any length in an array of integers. The Fibonacci class has a constructor to generate Fibonacci numbers of a given length \( n \), which is smaller than or equal to 25, and fill the array with these numbers. This class has a method called getIntAt(k) that returns the \( k \)th Fibonacci number if it is available, and -1 otherwise. The toString() method returns a string that represents the Fibonacci numbers, one number per line. The format is shown below in the case of length 5:

- Fibonacci number F0 is: 0
- Fibonacci number F1 is: 1
- Fibonacci number F2 is: 1
- Fibonacci number F3 is: 2
Fibonacci number \( F_4 \) is: 3

The **FibonacciTester** class has a main method to test the other classes. The program should start by printing a line in the form:

_FibonacciTester Written by YourName_

This line should include your full name. Add code to test the above three methods of the **Fibonacci** class you implement. Create several instances of this class including one of size 0 and one of size 25. Develop several test cases to test the other two methods thoroughly. Additionally, create two test cases for method `getIntAt()` with values that are too large (i.e., greater than 25) and too small (i.e., smaller than 0).

**Part 2 (8 points)**

In this part, you will add some additional features to class **Fibonacci** and add code to class **FibonacciTester** to test each added method thoroughly.

Create a method called `getStringAt(k)` that is like `getIntAt(k)` but returns a String presenting the corresponding Fibonacci number. If there is no such number, return null.

Test this method with several values for \( k \) including values are too large (i.e., greater than 25) and too small (i.e., smaller than 0).

Create a method called `showAllAsInt()` that print an array of integer, containing the Fibonacci numbers of this object, all on one line. The elements of the array shall be enclosed in brackets and separated by a comma and a single blank. There should be no blank between a bracket and the nearest number. The format is shown below in the case of length 5.

\[
[0, 1, 1, 2, 3]
\]

You will also create two methods, called `getOddAsInt()` and `getEvenAsInt()` to return only all of the odd numbers and only all of the even Fibonacci numbers from this object. Note an object shall be immutable. The signatures of these methods are listed below.

```java
public void showAllAsInt();
public int[] getOddAsInt();
public int[] getEvenAsInt();
```

You should add code to class **FibonacciTester** to test the above methods added to class **Fibonacci**. You have to at least test two Fibonacci numbers of length 10 and length 25 respectively. It is up to you to develop other test cases you feel necessary to convince yourself that the class has been implemented correctly. Your test program should run until completion without any user input.

**Part 3 (4 points)**

In this part of the project, you will create another class called **FibonacciLarge** that will be able to handle very large Fibonacci numbers. This part of the project is significantly more difficult than the other parts. In this class, each Fibonacci number is represented by
exactly 4 consecutive array elements. The first array element represents the low (i.e., the right-most) 9 digits of the number. The second and the following array elements each store 9 additional digits of the Fibonacci number. Each integer in the array will have a value between 0 and 10^9. For example, \(F_{100} = 354,224,848,179,261,915,075\) can be represented by four array elements:

- \(\text{arrF}[n] = 261915075\)
- \(\text{arrF}[n+1] = 224848179\)
- \(\text{arrF}[n+2] = 354\)
- \(\text{arrF}[n+3] = 0\)

When each subsequent Fibonacci number is generated by adding the previous two Fibonacci numbers, you should be careful to detect and propagate the carries when adding two respective array elements.

Define a static constant, \(\text{MAX\_LENGTH}\), in the class to have the value 150. Create a constructor to generate a series of Fibonacci numbers of any length \(N\), where \(N \leq \text{MAX\_LENGTH}\). Implement the following public method.

\[
\text{public String getStringAt(int k);}
\]

Add code to the \texttt{FibonacciNumberTester} class to test this method thoroughly. Use this method to implement the \texttt{toString()} method. But only test this method with small values of length.

**Extra Credit**

**Part 4**

Modify the \texttt{getAsString()} method in both the \texttt{Fibonacci} and \texttt{FibonacciLarge} class to return a string containing commas as appropriate. For example, \(\texttt{getAsString(100)}\) should return \(354, 224, 848, 179, 261, 915, 075\).

**Part 5**

Create and test a new class called \texttt{FibonacciUnlimited} that removes the limit on the maximum size of a Fibonacci number. You can use any internal representation you like, but do not use any of the Java classes (such as \texttt{BigInteger}) that are designed to handle large integers. Implement a constructor and methods \texttt{getIntAt(k)} and \texttt{getStringAt(k)}. Use your class to calculate \(F_{300}\) and return the value as a String.

**Hand-In Requirements (Deliverables):**

This project must be done using Eclipse. The project name should be your login name followed by the digit 1, all lower case. For example, if your name is \texttt{John Doe} your project name might be \texttt{jDoe1}. Put all of your classes in a package called \texttt{pr01}. Implement and test your \texttt{Fibonacci} and \texttt{FibonacciLarge} in stages. As you implement a feature, add code to the \texttt{FibonacciTester} to test the feature just added. Keep all of your test code.

You will submit a zip file of your project directory electronically to the lecture section on the Blackboard.