GCIS 124 Practicum 2
Instructor: Gahyun Park

1. **JavaFX (50%)**

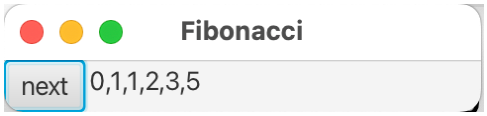
Examine provided code **FibonacciModel.java**, **FibonacciGUI.java**. and **Observer.java**.

1) Run the `FibonacciGUI` class. The GUI contains one button and one label that displays the first two Fibonacci numbers:

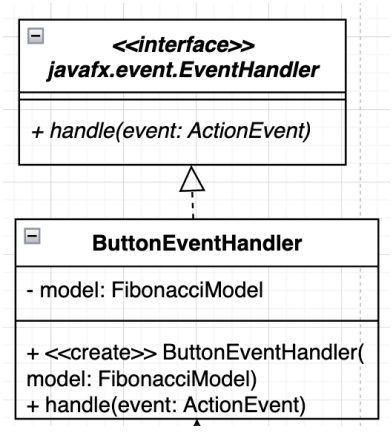

The objective is to improve the given classes in a way that the label displays the next number in the Fibonacci sequence every time the button is clicked.

<After the first click>

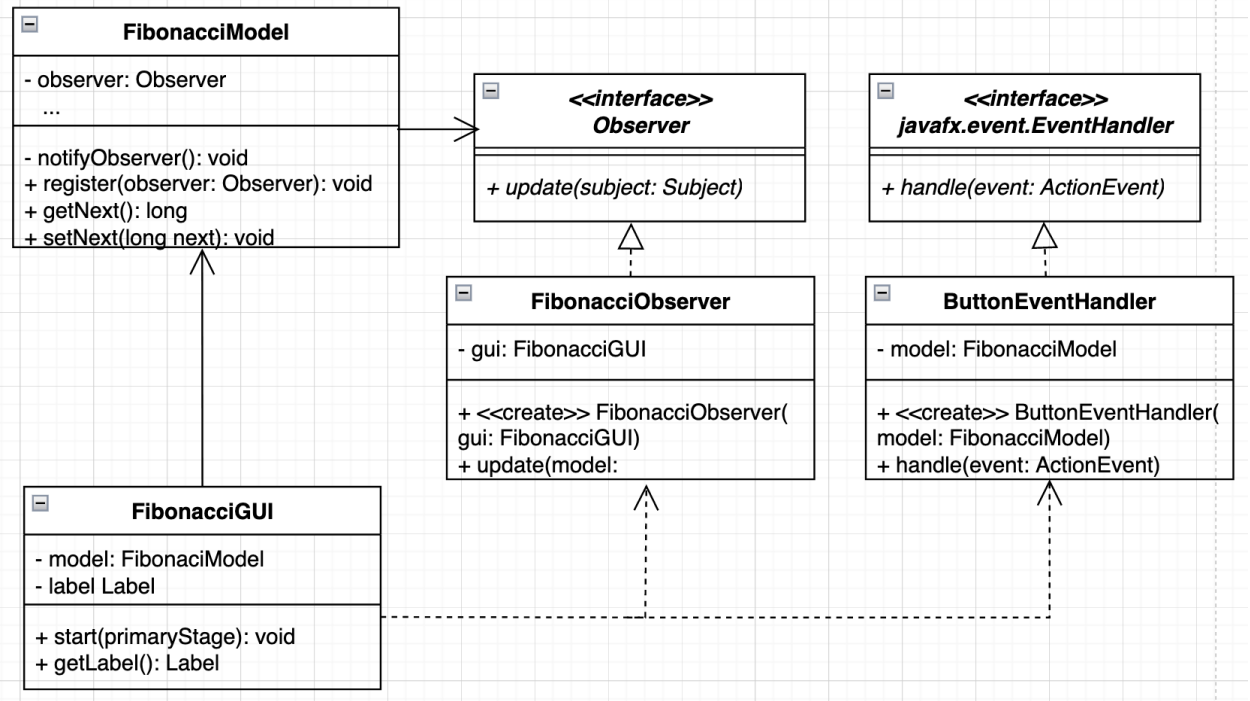

<After three more clicks>




2) When the button is pressed, the backend `FibonacciModel` should be updated first. Create a class that implements the `EventHandler` interface using the following UML as a guide. Don't forget to register on the button with an EventHandler object.




3) After updating the model, it's necessary to update the GUI to reflect the updated state of the model.
   a. First, create a class named `FibonacciObserver` that implements the `Observer` interface.
   b. You will also need to modify the `FibonacciModel` and `FibonacciGUI` classes. Don't forget to register on the model with an observer.
   c. Refer to the following UML if needed.




Choose between question 2 or question 3, and indicate which one should be graded when you commit to your git repository. You don't have to solve both questions.

2. **Iterable Strings (50%)**
   Open the file named **IterableString.java**. Your goal is to make the class iterable so that it can work with the for-each loop. This can be achieved by implementing a concrete Iterator (name it `StringIterator`) that will be used by the `iterator` method in the `IterableString` class.
   Hint: String's `charAt(int index)` method can be useful.

3. **Sorting people (50%)**
   Examine the file named **Person.java**. Create a file named **PeopleSort.java** file.
   1) In the `PeopleSort` class, write a static method named `peopleSort` that declares a parameter of type `List<Person>`.
      a. The method sorts the parameter list by person's name alphabetically.
      b. You must use either `PriorityQueue` or `TreeSet` to sort the list.
      c. You should not use any built-in sort functions or try to implement a sort algorithm.

   2) It may be necessary to modify the given Person class in order to arrange Person objects within the preferred data structure.

3) In the `PeopleSort` class, create a `main` method.
    a. Create an (unsorted) list that contains at least 5 `Person` objects and use the list to manually test the `peopleSort` method. Print the returned value.