# Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

# Problem Solving Team Members

Record the name of each of your problem solving team members here.

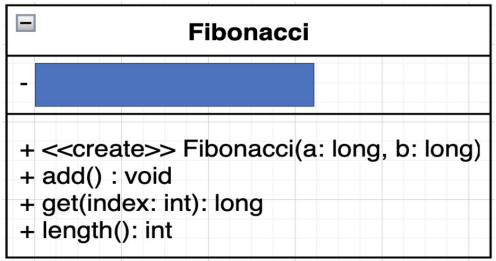
Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

| |
|---|
| Dessa Shapiro |
| Michael Rogowski |
| Raafay Ali |
| |
| |
| |

# Problem Solving 1

A Fibonacci sequence is a sequence of numbers in which each number is the sum of the two preceding numbers. Starting from two numbers, a and b, the first few values in the sequence is a, b, a+b, a+2b, 2a+3b, …

Consider a class named `Fibonacci` in the following diagram:




- The constructor creates an instance of Fibonacci sequence of length 2 with the specified two values.
- The `add` method is used to add the next number to the sequence.
- The `get` method returns the Fibonacci number at the specified index.
- The `length` method returns the length of the sequence.

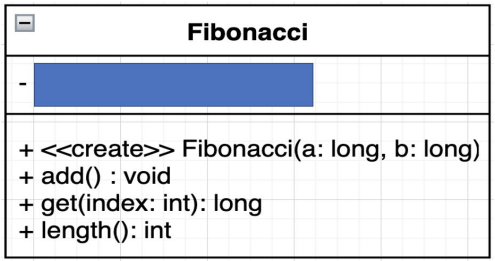
Assuming the `Fibonacci` class exists, write the output of the code to the right:

```
Fibonacci fib = new Fibonacci(2, 5);
fib.add();
fib.add();
fib.add();
for(int i=0; i<fib.length(); i++){
     System.out.println(fib.get(i));
}

Output:
2
5
7
12
19
```

```
public class Fibonacci {
	private long a;
	private long b;
	Private static final DEFAULT_SIZE = 10;
	Private Arraylist<Long> array;
	Private int size;

	public Fibonacci(long a, long b) {
		this.a = a;
		this.b = b;
		this.elements = new int[DEFULT_SIZE]
		This.size = 0;
	}

	public void add(){
	int first= get(this.size - 2);
	int second = get(this.size - 1);
	This.elements[size + 1] = first + second;
	size ++;
	}

	Public long get(int index){
		return array[index];
		Return arrayList.get(index)

	}
	Public int length(){
		return size;
		return arrayList.size();
	}
	@override
	Public Iterator<Long> iterator(){
		return new FibonacciIterator(arrayList);
}
```

# Problem Solving 2

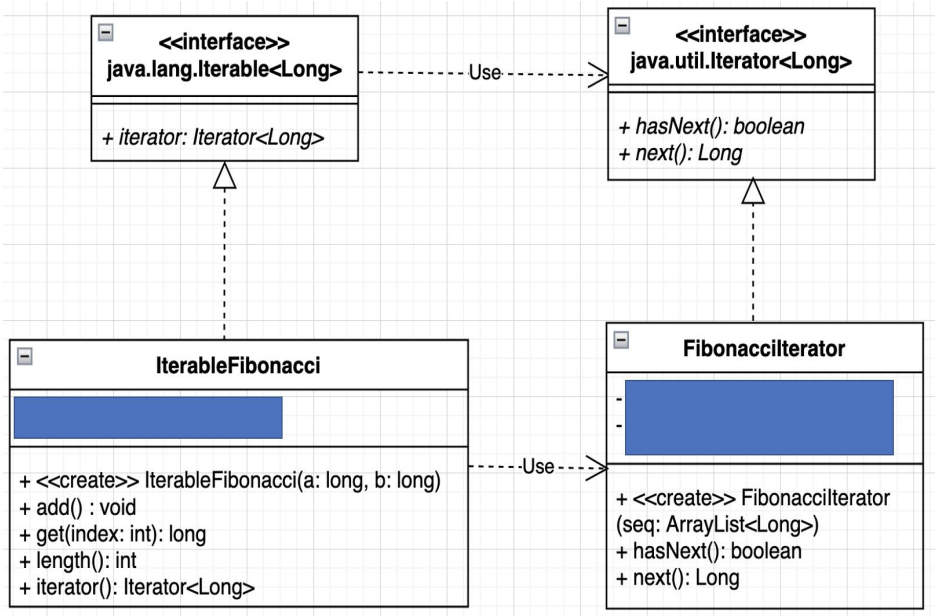
Write the `Fibonacci` class using the diagram as a guide.




- Discuss the state of the class with your team members and ensure that all of the required fields are declared with their corresponding types.
- Every method must run in O(C) time.

# Problem Solving 3

Transform your Fibonacci sequence into an **iterable** class for integration with the for-each loop mechanism. As you learned in lecture, this requires writing a custom iterator. Use the following UML class diagram as a guide, write a class named `FibonacciIterator` in the space at right. Then use it to modify the `Fibonacci` class in the previous slide.




```
Import java.util.Iterator
public class FibonacciIterator extend
Iterator<Long> {
      private long a;
      private long b;
      Private static final DEFAULT_SIZE = 10;
      Private int[] array;
      Private int size;

      public FibonacciIterator (long a, long b) {
            this.a = a;
            this.b = b;
            this.elements = new int[DEFULT_SIZE]
            This.size = 0;
      }
```

```
Import assertequals

@Test
Public void testName(){
//setup
      Fibonacci f = new Fibonacci(4, 5);
      f.add();

      String expected = “9, 14, 23”


//invoke
      String actual= “”
      for(int ele: array){
            actual +=+ “, “+Integer.toString(ele);
      }
//analyze
assertequals(expected, actual);
}
```

# Problem Solving 4

Write a unit test to verify that the Fibonacci class can be iterated over with a for-each loop.