Arrays

An array in Java is used to store a fixed number of elements of the same type. New arrays can be created either by using the Java new operator or by using curly brackets ({ and }) to provide an explicit list of initial values for the array. Below are two examples:

```
//create a new array with five elements, each initialised to zero
int[] intArray = new int[5];

//create a new array and initialise the elements with the
provided values
long[] longArray = {1L, 2L};
```

You can only use the curly bracket notation when creating a new variable of an array type; you cannot use it to update the values stored in an existing variable. For example:

```
//This is okay; create new array with 4 elements
long[] longArray = {1L, 2L, 3L, 4L};

//This is wrong; you cannot update values in an array this way
longArray = {1L, 2L, 3L, 4L};
```

You might like to try this to see the error from the compiler. Square brackets are used to retrieve or update values stored in the array. By convention, the first element in the array has an index value of zero, not one. Here are some examples:

```
//create a new array with five elements, each initialised to zero
int[] intArray = new int[5];

//set the value of the FIRST element to 2
intArray[0] = 2;

//set the value of the THIRD element to 9
intArray[2] = intArray[0] + 7;

//increment the value stored in the FOURTH element to 1
intArray[3]++;
```

The square bracket notation only permits a single array value to be updated at a time. Consequently it is a common to use a for loop to update values stored in an array. Here is an example:

```
int[] numbers = new int[5];
for(int i=0; i<numbers.length; i++)
```

```
 numbers[i] = i;
```

In the above example the for loop is used to update each value of the "array of ints" numbers in turn with the current value stored in the variable i.

Notice the use of the phrase numbers.length; the value stored in the length field is a read-only value of the size of the array. It is a good idea to use the length field when using Java arrays, rather than using literal values.

Task

Create a class called FibonacciCache.

```
public class FibonacciCache {
 //TODO: Test your program with values other than 20 as given
here
 public static long[] fib = new long[20];

 public static void store() {
  //TODO: using a for loop, fill "fib" with the Fibonacci numbers
  //      e.g. if length of "fib" is zero, store nothing; and
  //           if length is six, store 1,1,2,3,5,8 in "fib"
 }

 public static void reset() {
  //TODO: using a for loop, set all the elements of fib to zero
 }

 public static long get(int i) {
  //TODO: return the value of the element in fib found at index i
  //      e.g. "get(3)" should return the fourth element of fib
  //
  //Note: your code should check that i is within the bounds of
fib
  //      and if it is outside this range return the literal "-
1L"
 }
}
```

You will probably want to test your code by writing a "main" method and including some test code which uses your reset, store and get implementations and checks they function correctly. Printing out the contents of fib after performing each method call will probably help you find errors.