Let's Write: FizzBuzz non-modular

What is software testing?
- Writing code to evaluate the correctness of code
- Set *expectations* for a program's output given an input
- Developing a specification for the program defined by its I/O

Two types of tests:
- Unit tests: "white-box" tests, evaluate individual methods or classes
- Integration tests: "black-box" tests, evaluate complete programs and processes

What does a test look like?
- Object interaction (construction, method calls, etc)
- "Assertions" that evaluate outcomes
    - assertEquals
    - assertTrue
    - fail
- Dependency mocking (optional)
    - convert unpredictable network request into dummy response function

Let's write: FizzBuzz modular
See/write: FizzBuzzTest

See: Sorter
See: test.SorterTest

Some languages like JavaScript separate the above three components:
- Test Runner (runs tests in sequence, reports output)
- Assertion library
- Mocking library

jUnit contains the first two. "Mockito" is a Java method mocking library (but we won't need it)

See: How do we add jUnit to our projects?
- jUnit is included in all project code

****

BIG IDEA: Abstraction - what is it?

What are Classes?
 - Blueprints for Objects
 - Containers of "state", composed of variables inside
 - Expose methods for manipulating state

C Programming "Procedural" vs Java OOP Programming
C structs, functions that modify structs
Java Objects, methods that modify object state
Information hiding (encapsulation)

See: Circle, a very simple container class
Let's write: Rectangle, a simple container class with doubleDimensions() method
Observe: doubleDimensions() performs operation without exposing actual stored state (encapsulation, abstraction)

What are Interfaces?
 - Describes the methods a class is *expected* to expose
 - Describes method names, but not implementation

Implement: Shape2D
See: Shape2DTest

Let's talk about Project 0...
https://www.briancui.com/csc-143/projects/project0.html

```java
package interview;

public class FizzBuzz {
    /*
        FizzBuzz is often considered a meme/joke interview question,
        but there are several online anecdotes of interviewers who are
        surprised to find how few applicants are able to get past what
appears
        to be a simple programming problem. If you can write FizzBuzz,
congratulations!
        You're in the top 1%.

        Write a program that prints the numbers 1 to 100, inclusive.
        Except, if the number is divisible by 3, print "Fizz".
        If the number is divisible by 5, print "Buzz".
        If the number is divisible by both, print "FizzBuzz".
     */

    public static void main(String args[]) {
        for (int i = 1; i <= 100; ++i) {
            System.out.println(fizzbuzz(i));
        }
    }

    public static String fizzbuzz(int i) {
        if (i % 3 == 0 && i % 5 == 0) {
            return "FizzBuzz";
        } else if (i % 3 == 0) {
            return "Fizz";
        } else if (i % 5 == 0) {
            return "Buzz";
        }

        return Integer.toString(i);
    }
}
```

```java
package test;

import interview.FizzBuzz;
import org.junit.Test;

import static junit.framework.TestCase.assertEquals;
import static junit.framework.TestCase.assertTrue;

public class FizzBuzzTest {
    String fizz = "Fizz";
    String buzz = "Buzz";
    String fizzbuzz = fizz + buzz;

    @Test
    public void DivisibleByThreeTest() {
        assertEquals(fizz, FizzBuzz.fizzbuzz(3));
        assertEquals(fizz, FizzBuzz.fizzbuzz(6));
        /* ... */

        for (int i = 3; i <= 100; i += 3) {
            assertTrue(FizzBuzz.fizzbuzz(i).contains(fizz));
        }
    }

    @Test
    public void DivisibleByFiveTest() {
        assertEquals(buzz, FizzBuzz.fizzbuzz(5));
        assertEquals(buzz, FizzBuzz.fizzbuzz(10));
        /* ... */

        for (int i = 5; i <= 100; i += 5) {
            assertTrue(FizzBuzz.fizzbuzz(i).contains(buzz));
        }
    }

    @Test
    public void DivisibleByBothTest() {
        assertEquals(fizzbuzz, FizzBuzz.fizzbuzz(15));
        assertEquals(fizzbuzz, FizzBuzz.fizzbuzz(30));
        /* ... */

        for (int i = 15; i <= 100; i += 15) {
            assertEquals(fizzbuzz, FizzBuzz.fizzbuzz(i));
        }
    }

    @Test
    public void IndivisibleTest() {
        for (int i = 1; i <= 100; i++) {
            if (i % 3 == 0 || i % 5 == 0) continue;

            assertEquals(Integer.toString(i), FizzBuzz.fizzbuzz(i));
        }
    }
}
```

```java
package scratch;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

public class Sorter {
    public static class SimpleComparator implements Comparator<Integer> {
        @Override
        public int compare(Integer o1, Integer o2) {
            //return o1.compareTo(o2);
            int a = o1.intValue();
            int b = o2.intValue();

            return a - b;
        }
    }

    public static List<Integer> sort(List<Integer> c) {
        c.sort(new SimpleComparator());
        return c;
    }

    public static List<Integer> copySort(List<Integer> c) {
        List<Integer> copy = List.copyOf(c);
        sort(c);
        return copy;
    }
}
```

…

```java
public interface Shape2D {
    public double area();
}
```