Junit Example

JUnit overview

- Junit (inspired by Sunit) is a simple "testing framework" that provides:
  - classes for writing Test Cases and Test Suites
  - methods for setting up and cleaning up test data ("fixtures")
  - methods for making assertions
  - textual and graphical tools for running tests

JUnit 4 Annotations

- `@BeforeClass`:
  Methods that are called before any test runs. Test parameters can be set and objects can be instantiated here if they do not change in later tests.
- `@AfterClass`:
  Methods that are called at the end of a testsuite after all tests have been run. Resources that have been used during the tests (like network connections or streams) can be freed here.
- `@Before`:
  Methods that are executed before every test.
- `@After`:
  Methods that are executed after every test.
- `@Test`:
  The actual test methods.
- `@Ignore`:
  Methods, testing features that are still to be implemented, can be disabled temporarily.

JUnit 4 Assertions

- `assertEquals(a, b)`
- `assertTrue(a, b)`
- `assertFalse(a, b)`
- `assertNull(a, b)`
- `assertNotNull(a, b)`
- `assertSame(a, b)`
- `assertNotSame(a, b)`

JUnit 4 Example

```
public class SimpleFraction {

      private int numerator,
      private int denominator;

      public SimpleFraction(int num, int den) {
          numerator = num;
          denominator = den;
      }

      public void simplify() {
          long gcd = gcd(denominator, numerator);
          denominator /= gcd;
          numerator /= gcd;
          if(denominator < 0) {
              denominator = -denominator;
              numerator = -numerator;
          }
              }

    private static int gcd(int a, int b) {
          return b == 0 ? a : gcd(b, a % b);
    }
```

```
public int getDenominator() {
   return denominator;
 }

 public int getNumerator() {
   return numerator;
 }

 public void setDenominator(int i) {
   denominator = i;
 }

   public void setNumerator(int i) {
      numerator = i;
   }
   public static void main(String[] args) {
      SimpleFraction fract = new SimpleFraction(10, 3);
      System.out.println(fract.getNumerator() +" "
+fract.getDenominator());
   }
}
```

```
import org.junit.Test;

public class SimpleFractionTest{

    private SimpleFraction f1, f2;

  @Before
  protected void setUp(){
      f1 = new SimpleFraction(15, 25);
      f2 = new SimpleFraction(-27, 6);
  }

  @Test
  public void getDenominator() {
   int result = f1.getDenominator();
   assertTrue("getDenominator() returned" +result +" instead of 25.",
                          result==25);
   result = f2.getDenominator();
   assertEquals(6, result);}
```

```
    @Test
public void simplify() {
        f1.simplify();
        assertEquals(3, f1.getNumerator());
        assertEquals(5, f1.getDenominator());

        f2.simplify();
        assertEquals(-9, f2.getNumerator());
        assertEquals(2, f2.getDenominator());
    }
}
```

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(value=Suite.class)
@SuiteClasses(value={SimpleFractionTest.class})
public class AllTests {

}
```