Rational Class with Exceptions and Testing

This lab exercise contains starter files with a skeleton for a class called **`Rational`**. However, you should copy and paste the contents of the methods from last week's lab methods into that file (don't copy and paste the file, copy and paste the contents of each method one by one). Your tasks this week will be to make the Rational class officially immutable, add a **`subtract`** method, implement a copy constructor, add some exception throwing and practice JUnit tests,

Stages once you have transferred your existing code to the new Rational.java file:

(1) Notice that I've added the keyword final in front of the private numerator and denominator fields. Check to see whether anything else in your exist methods code needs to change as a result.

(2) Implement the new **`subtract`** method.

(3) Implement the copy constructor method.

(4) We have given you a file **`StudentTests.java`** in which you should implement the tests specified.

(5) Consider each method of the class and decide whether it needs to have a test for a zero as the denominator of some Rational object that it will create. If it does, add that conditional test with the body being the line of code:

```
throw new ArithmeticException("Divide by Zero");
```

(6) Write some new test cases to see if this exception is thrown when it should be. For example, the following will try to construct an invalid **`Rational`** object. If it succeeds in creating that illegal object, the test fails due to the **`fail();`** statement. If the attempt throws the exception, the test will pass due to no lines of code with assertions that indicate the test failed.

```
@Test
public void testConstructorOnDivideByZero() {
  try {
     Rational rationalNum = new Rational(3, 0);
     fail(); //If you get to this line of code, something went wrong.
  }
  catch (ArithmeticException e) {
     //Nothing really needed here for this test since we
     //  WANT this exception caught for the test to pass.
  }
}
```