Software Architecture

Lab 04

TDD and JUnit Testing

Néstor Cataño
Innopolis University

Spring 2016

Software Testing

- **Testing** is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
  - When you test software, you execute a program using artificial data.
  - You check the results of the test by checking errors, anomalies, etc.

Testing goals

1. To **demonstrate** to the developer and the customer that the software meets its requirements.
2. To **discover** situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification.

Test Driven Development (TDD)

- Production code is written to make a failing **unit test** pass.
  1. we write a unit test that fails because the functionality it is testing does not exist. **Then**,
  2. we write the code that makes that test pass.
- **Test cases and code evolve together**, with the test cases leading the code by a very small fraction.

Unit Testing – Testing an Object

1. Test all operations associated with the object
2. Set and check the value of all attributes associated with the object
3. Put the object into all possible states. This means that one should simulate all events that an object state change

Java Unit (JUnit) Testing in Eclipse

A JUnit Test is Composed of

1. **setup part**
   - inputs and outputs are initialized
   - it plays the same role as a constructor
2. **call part**
   - one calls the object or method to be tested
3. **assertion part**
   - one compares the result of the call with the expected result

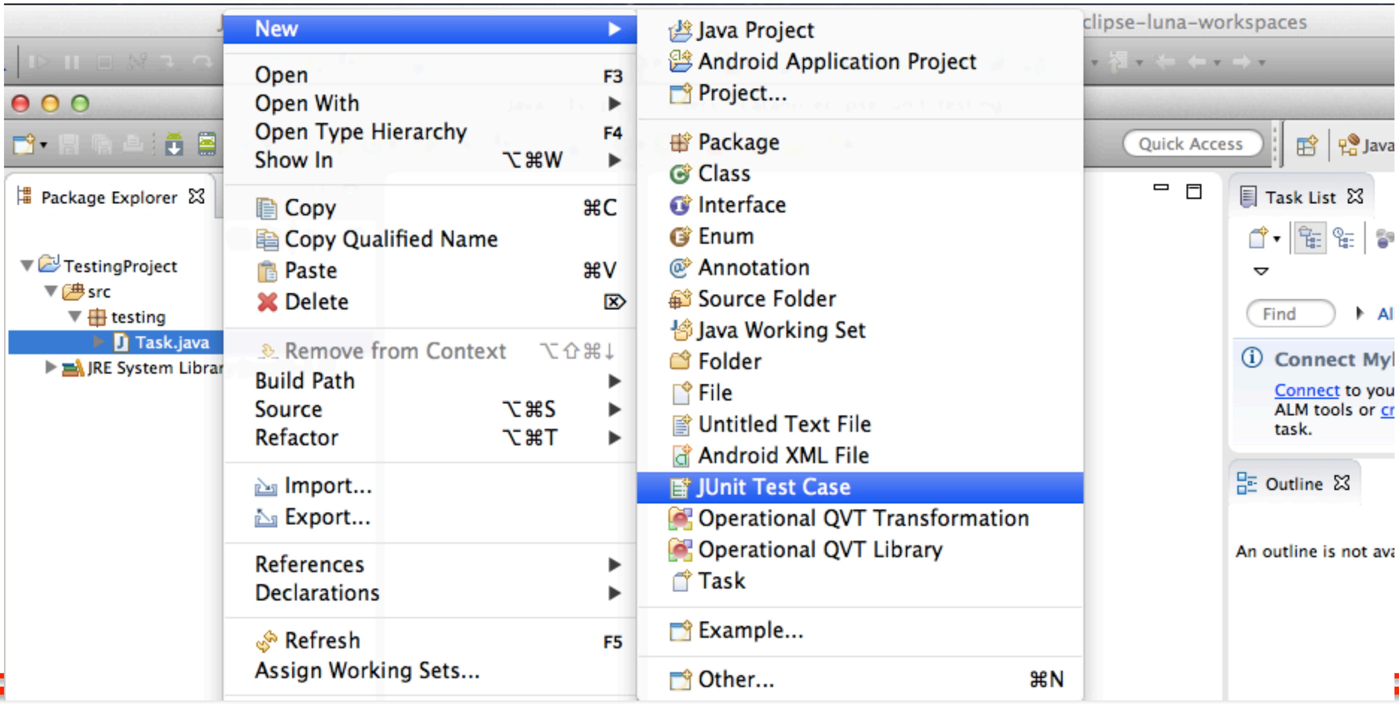
Creating a JUnit Test in Eclipse – Step 0

- Create a Java Project that includes the code below
  - **File**, **New**, **Project**, **Java Project**, **Next,** introduce **TestingProject**, **Finish**
  - **Right click TestingProject**, **New**, **Class**, name your class **Task**, **Finish**

```
public class Task {
 private int psd, pcd;
 private int asd, acd;

 public Task() { psd = pcd = asd = acd = -1;}
 public int getPsd() { return psd; }
 public void setPsd(int psd) { this.psd = psd; }
 public int getPcd() { return pcd; }
 public void setPcd(int pcd) { this.pcd = pcd; }
}
```

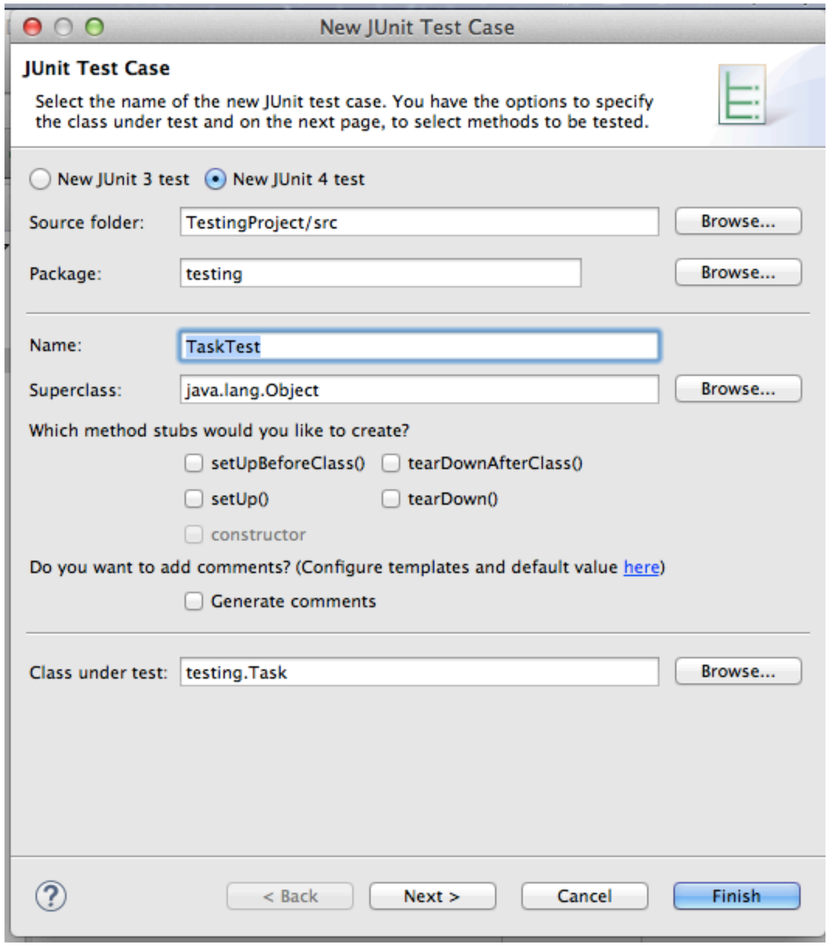
Creating a JUnit Test in Eclipse – Step 1

- In the Package Explorer right-click **Task.java** and select **New**, **Other**, **Java**, **JUnit**, **JUnit Test Case**



Creating a JUnit Test in Eclipse – Step 2

- Select **New Junit 4 test** and click **Next**
- This will import JUnit 4 library to your project



New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

New JUnit 3 test [✓ New JUnit 4 test]

Source folder: TestingProject/src

Package: testing

Name: TaskTest

Superclass: java.lang.Object

Which method stubs would you like to create?

- ☐ setUpBeforeClass()
- ☐ tearDownAfterClass()
- ☐ setUp()
- ☐ tearDown()
- ☐ constructor

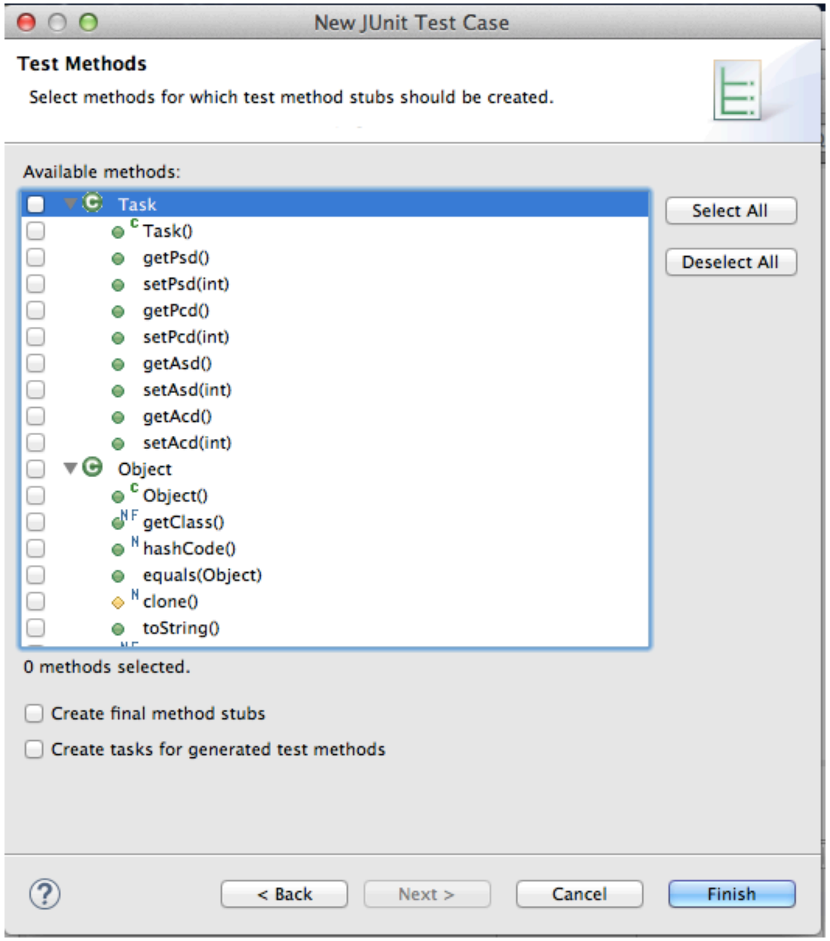
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Class under test: testing.Task

Creating a JUnit Test in Eclipse – Step 3

- Don't select any of the methods → we will create the **JUnit tests manually** based on the software requirements
- Click **Finish**
- Select **Add JUnit 4 library to the build path**



Select methods for which test method stubs should be created.

Available methods:

- [ ] Task
  - [ ] Task()
  - [ ] getPsd()
  - [ ] setPsd(int)
  - [ ] getPcd()
  - [ ] setPcd(int)
  - [ ] getAsd()
  - [ ] setAsd(int)
  - [ ] getAcd()
  - [ ] setAcd(int)
- [ ] Object
  - [ ] Object()
  - [ ] getClass()
  - [ ] hashCode()
  - [ ] equals(Object)
  - [ ] clone()
  - [ ] toString()

0 methods selected.

- [ ] Create final method stubs
- [ ] Create tasks for generated test methods

Class Task

```java
public class Task {
 private int psd, pcd;
 private int asd, acd;

 public Task() { psd = pcd = asd = acd = -1;}
 public int getPsd() { return psd; }
 public void setPsd(int psd) { this.psd = psd; }
 public int getPcd() { return pcd; }
 public void setPcd(int pcd) { this.pcd = pcd; }
}
```

Software Requirement 1

The ``**planned start date**'' for a task is smaller than or equal to its ``**planned completion date**''

@Test – Requirement 1

```java
import org.junit.*; import static org.junit.Assert.*;

public class TaskTest {
 Task task;

 @Before
 public void setUp() throws Exception { task = new Task(); }

 @After
 public void tearDown() throws Exception { ... }

 @Test
 public void testRequirement1() { ... }
}
```

@Before setUP

```java
@Before
public void setUp() { ... }
```

- Runs **before** each `@Test Method`

@After tearDown

```
@After
public void tearDown() { ... }
```

- Runs **after** each `@Test Method`

@Test
Method

```
@Test
public void testMethod() { ... }
```

- **JUnit** @Test **Method**

Software Requirement 1

The ``**planned start date**'' for a task is smaller than or equal to its ``**planned completion date**’’

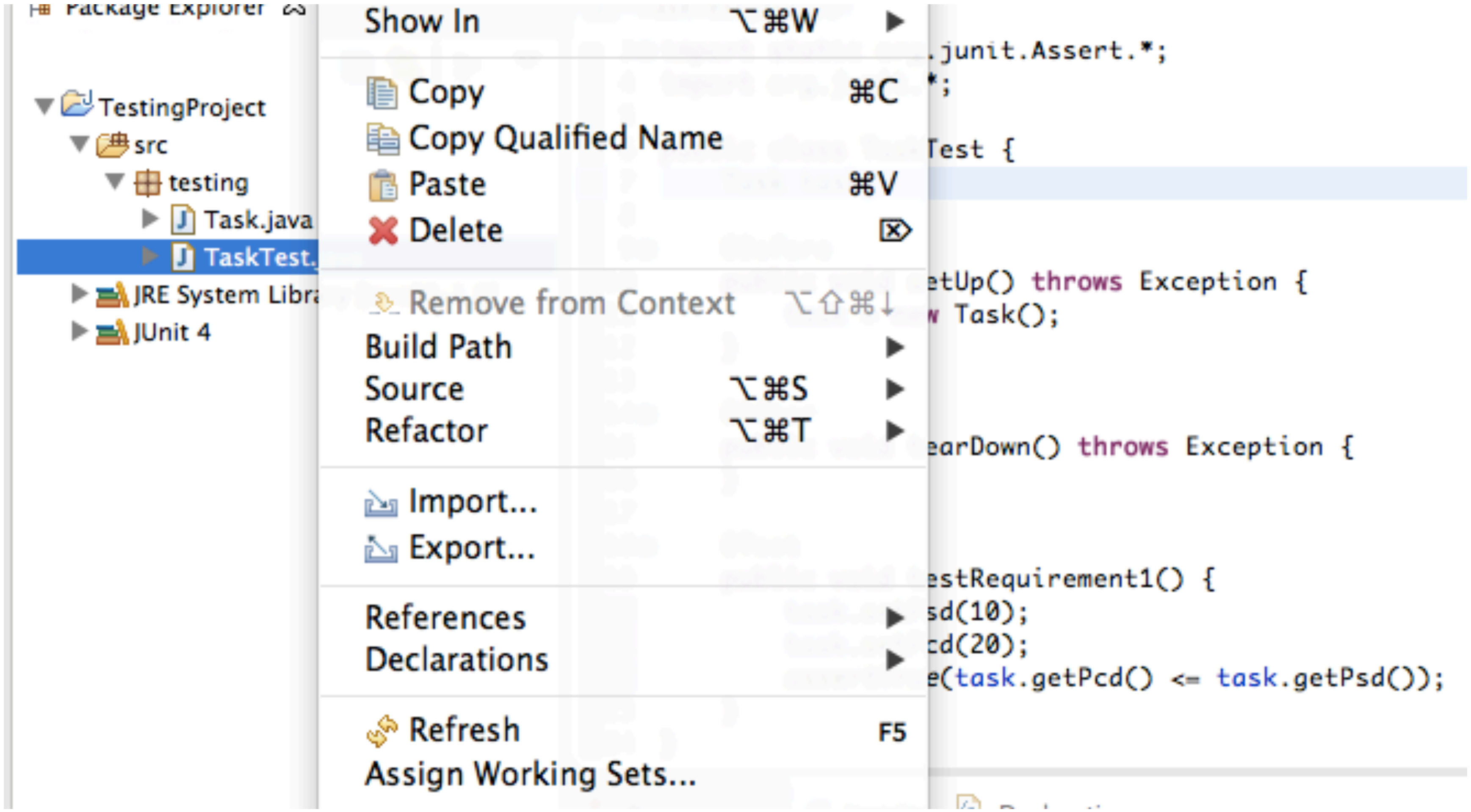
@Test testRequirement1

```
@Test
public void testRequirement1() {
 task.setPsd(20);
 task.setPcd(10);
 assertTrue(task.getPsd() <= task.getPcd());
}
```

We want to make the test fail !

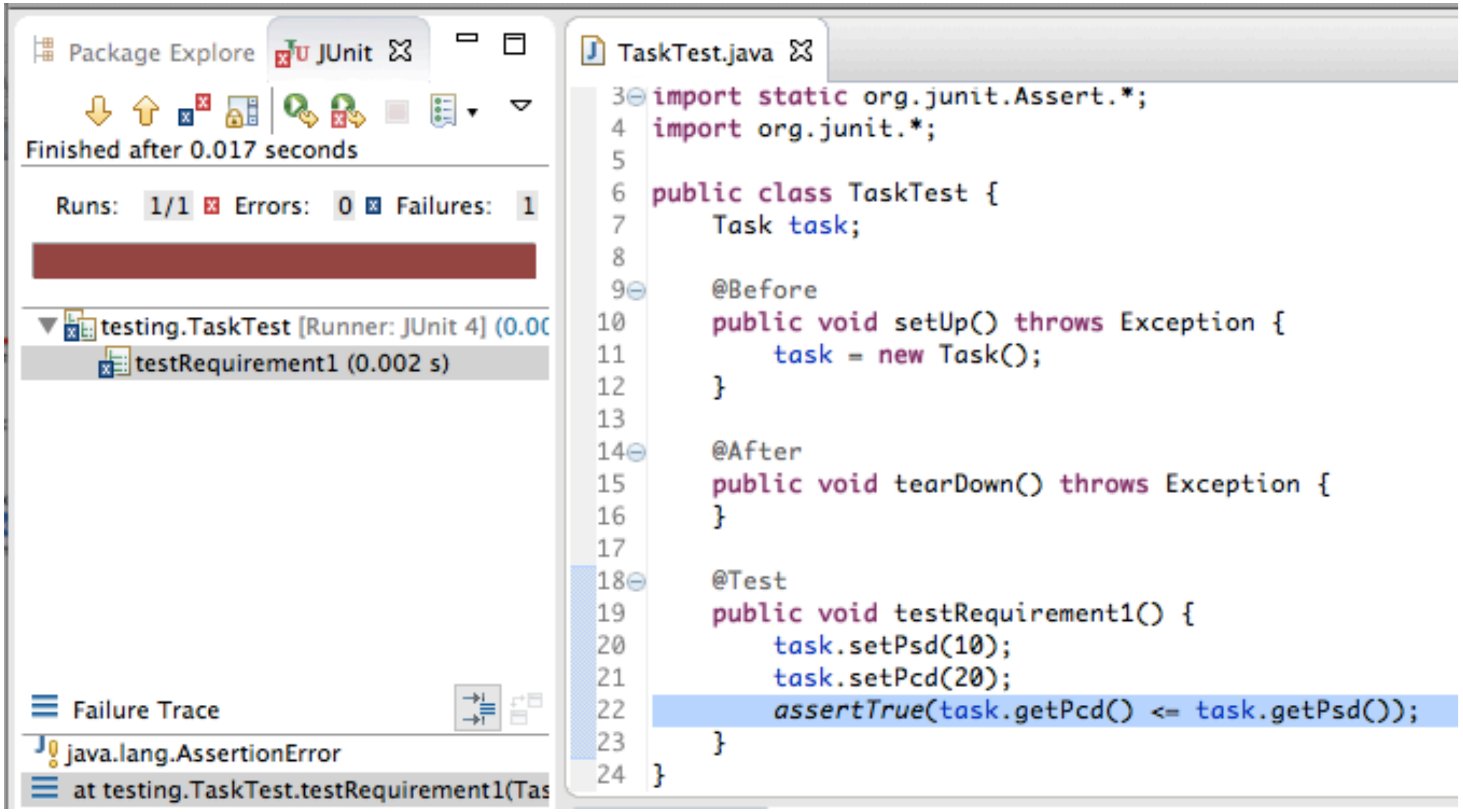
Notice that we're only considering positive values here

@Test
Right Click TaskTest ... Run JUnit Test



```
.junit.Assert.*;
*;

Test {


etUp() throws Exception {
w Task();


earDown() throws Exception {


estRequirement1() {
sd(10);
cd(20);
e(task.getPcd() <= task.getPsd());
```

@Test
Failure



```
Finished after 0.017 seconds
Runs: 1/1  Errors: 0  Failures: 1

testing.TaskTest [Runner: JUnit 4] (0.0
  testRequirement1 (0.002 s)

java.lang.AssertionError
at testing.TaskTest.testRequirement1(Tas
```

```java
import static org.junit.Assert.*;
import org.junit.*;

public class TaskTest {
    Task task;

    @Before
    public void setUp() throws Exception {
        task = new Task();
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void testRequirement1() {
        task.setPsd(10);
        task.setPcd(20);
        assertTrue(task.getPcd() <= task.getPsd());
    }
}
```

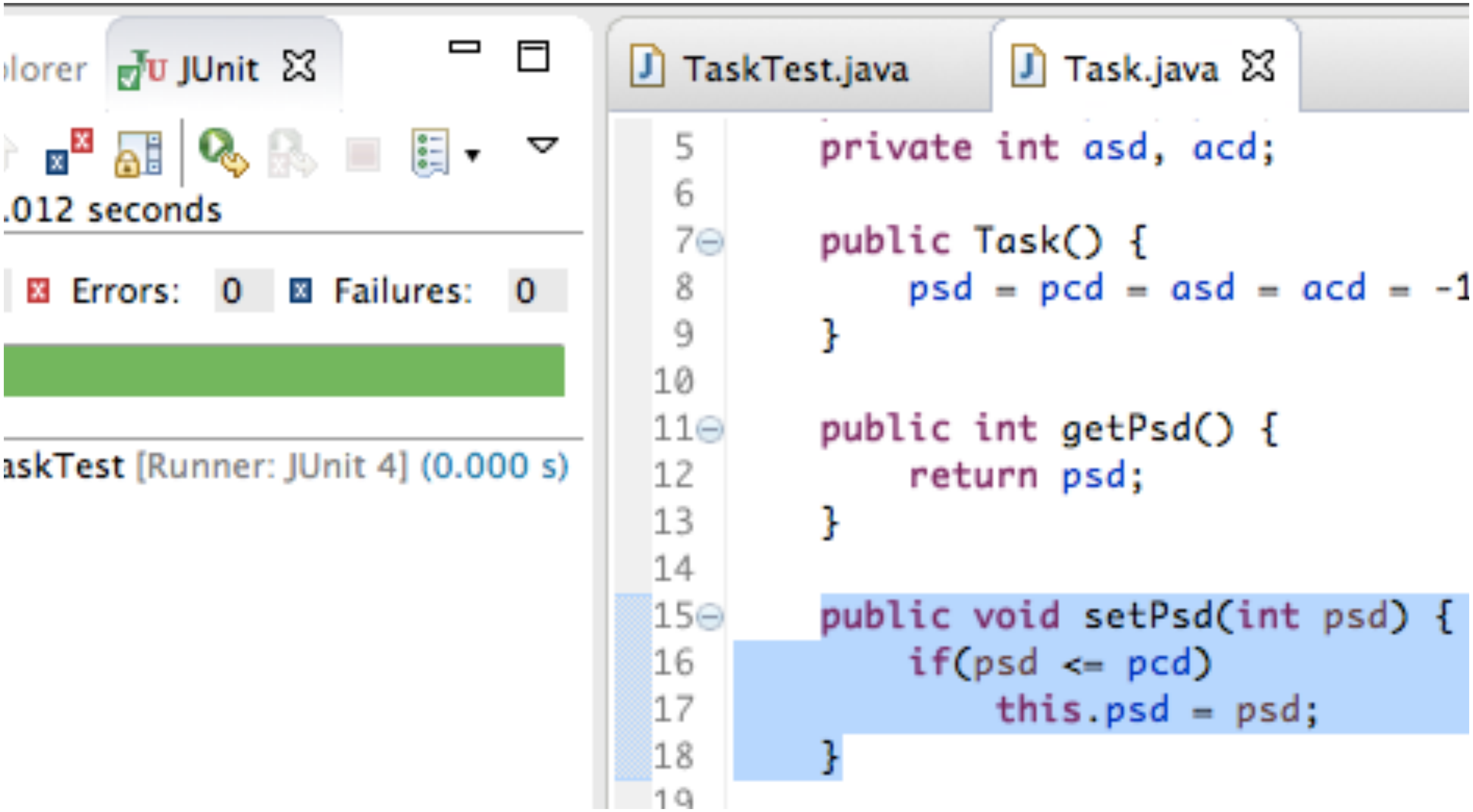
Evolving the Implementation

```
public void setPsd(int psd) {
   this.psd = psd;
}
```

Adapting the Implementation

```
public void setPsd(int psd) {
   if(psd <= pcd)
      this.psd = psd;
}
```

Tess Pass
Re-Running the JUnit Test



.012 seconds

Errors: 0 Failures: 0

askTest [Runner: JUnit 4] (0.000 s)

```
    private int asd, acd;

    public Task() {
        psd = pcd = asd = acd = -1
    }

    public int getPsd() {
        return psd;
    }

    public void setPsd(int psd) {
        if(psd <= pcd)
            this.psd = psd;
    }
```

Now Do It Yourself!!

Test Coverage

Planned Start Date	Planned Completion Date
Negative	Negative
Positive	Positive
Positive	Negative
Negative	Positive