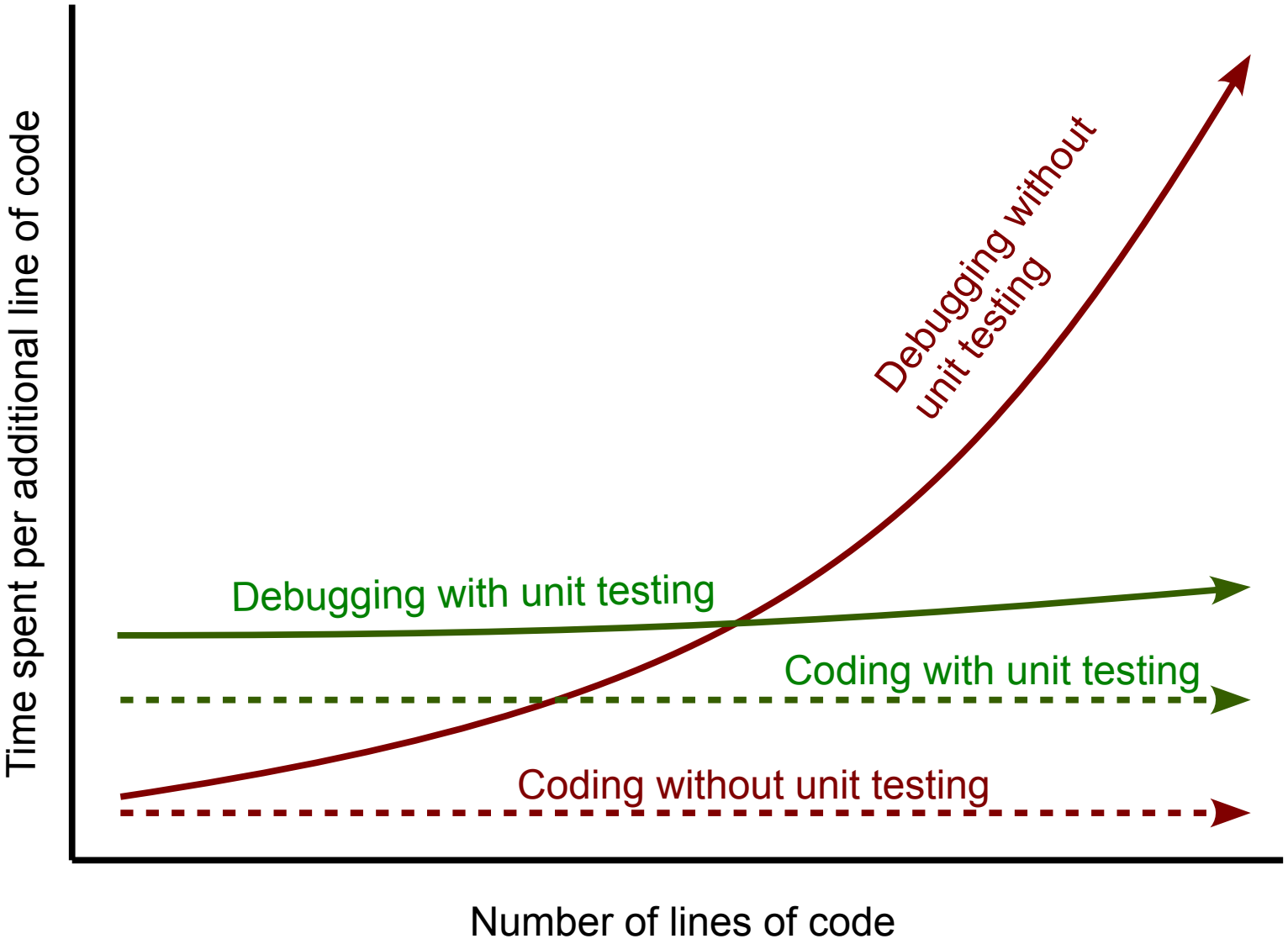
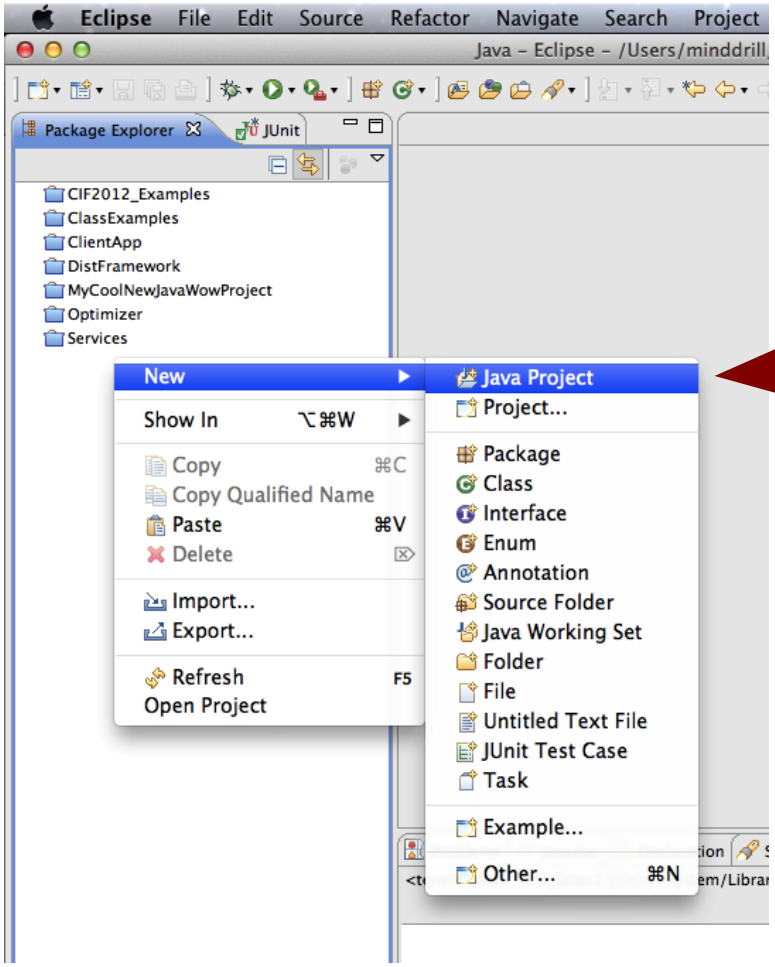
Why do we need testing?



What is Test Driven Development?

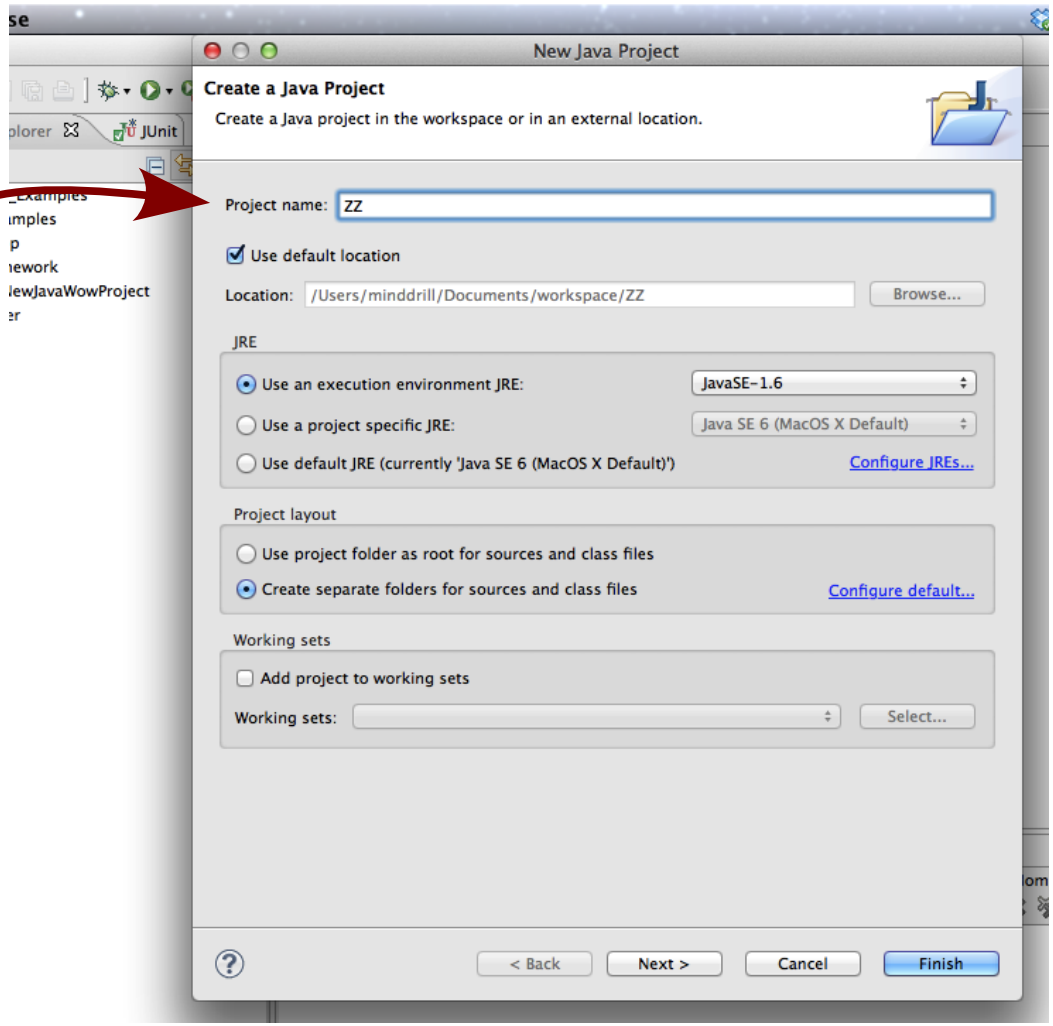
- Test Driven Development is the idea that before we write some piece of the functionality of a class, we want to write the test(s) that we want that piece of functionality to pass
- Why don't we write all of the tests after we write a class?
  - A class may have complex functionality in which the errors produced by a completed class may not be as obvious as errors produced by individual methods of that class
    - To the extent possible, we want to test the functionality of those individual methods
  - There are tests that may pass in error because we are not testing the functionality of a class that we think we're testing
    - We want to write a test first, watch it fail, then write the functionality of a class that makes that test succeed

Installing Junit in Eclipse - Example



Create a new Java project

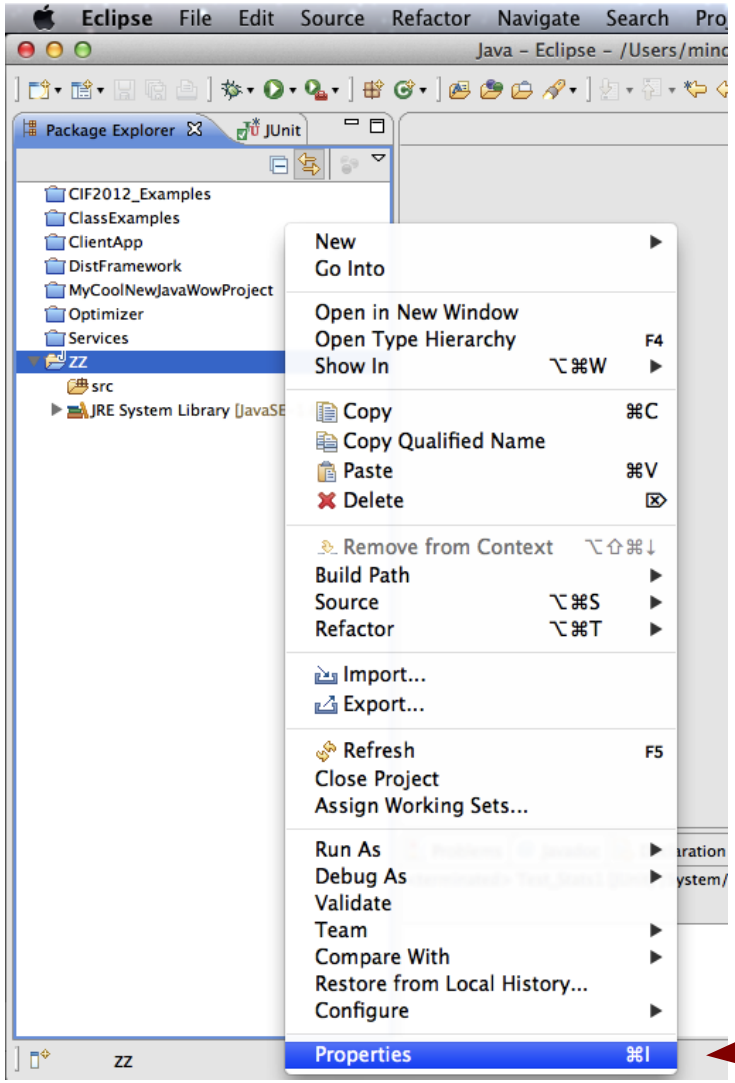
Creating New Java Project



For example, let's call the project ZZ

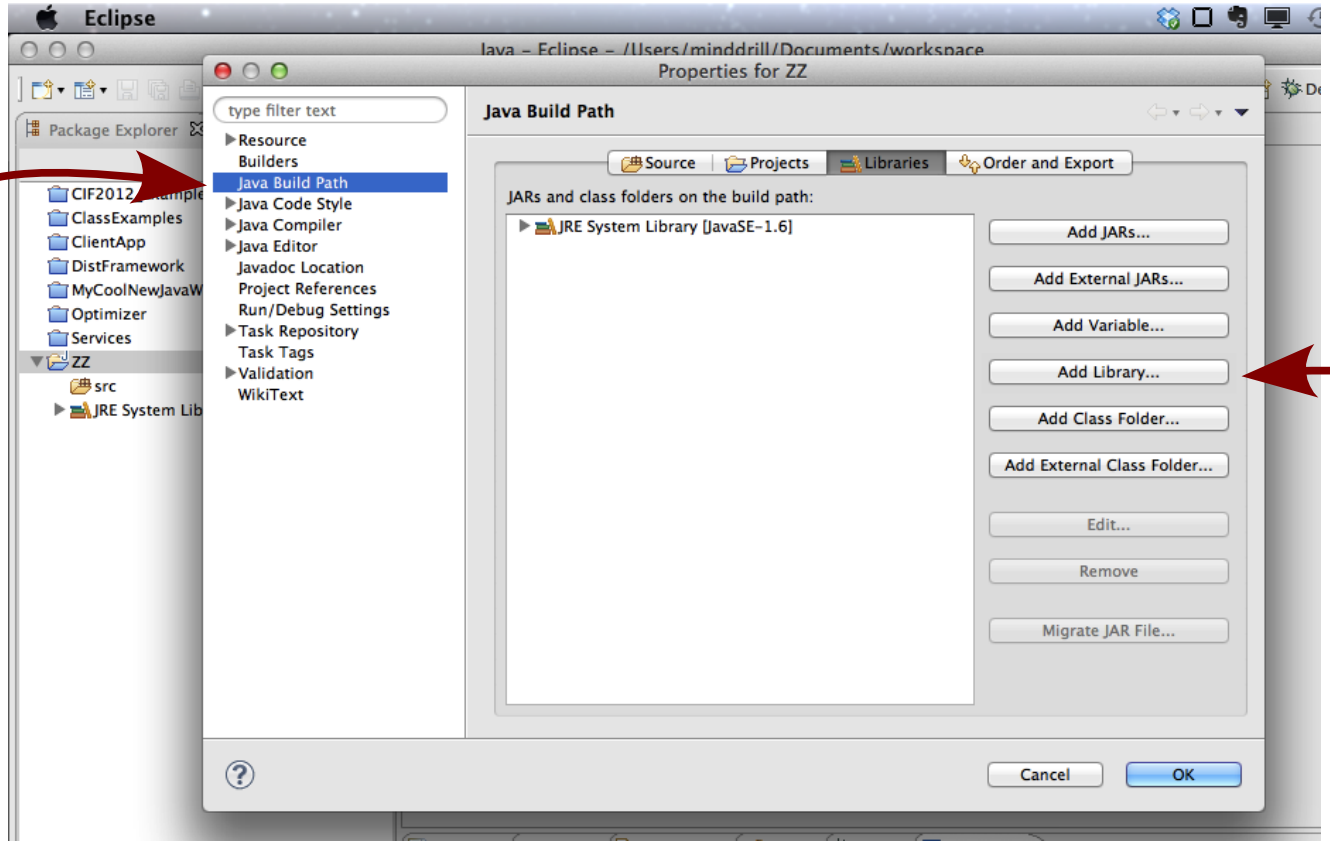
Modifying Project Properties

Project name



Click on *Properties*

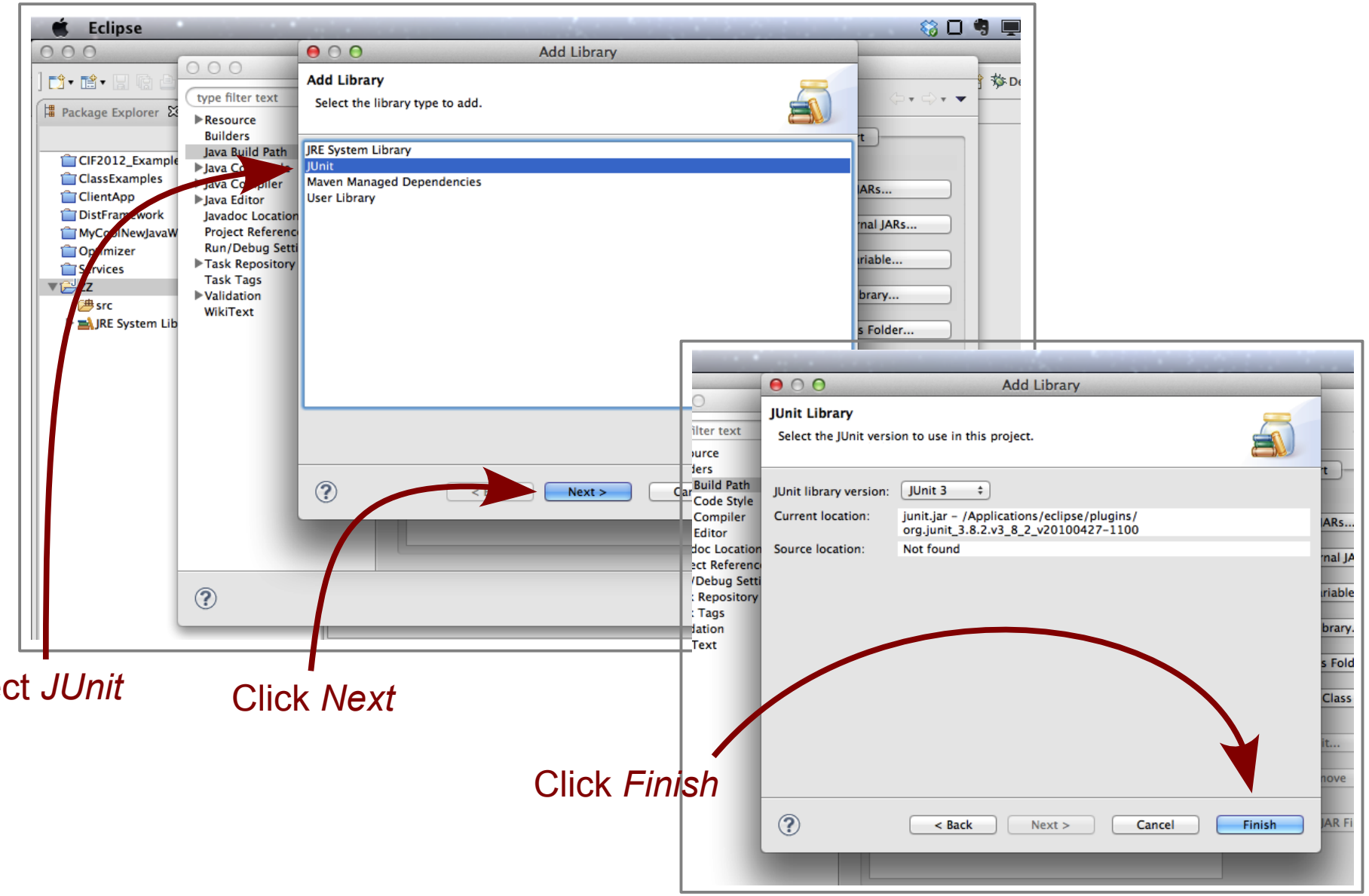
Adding a Library



Select *Java Build Path*

Click on *Add Library*

Selecting JUnit Library

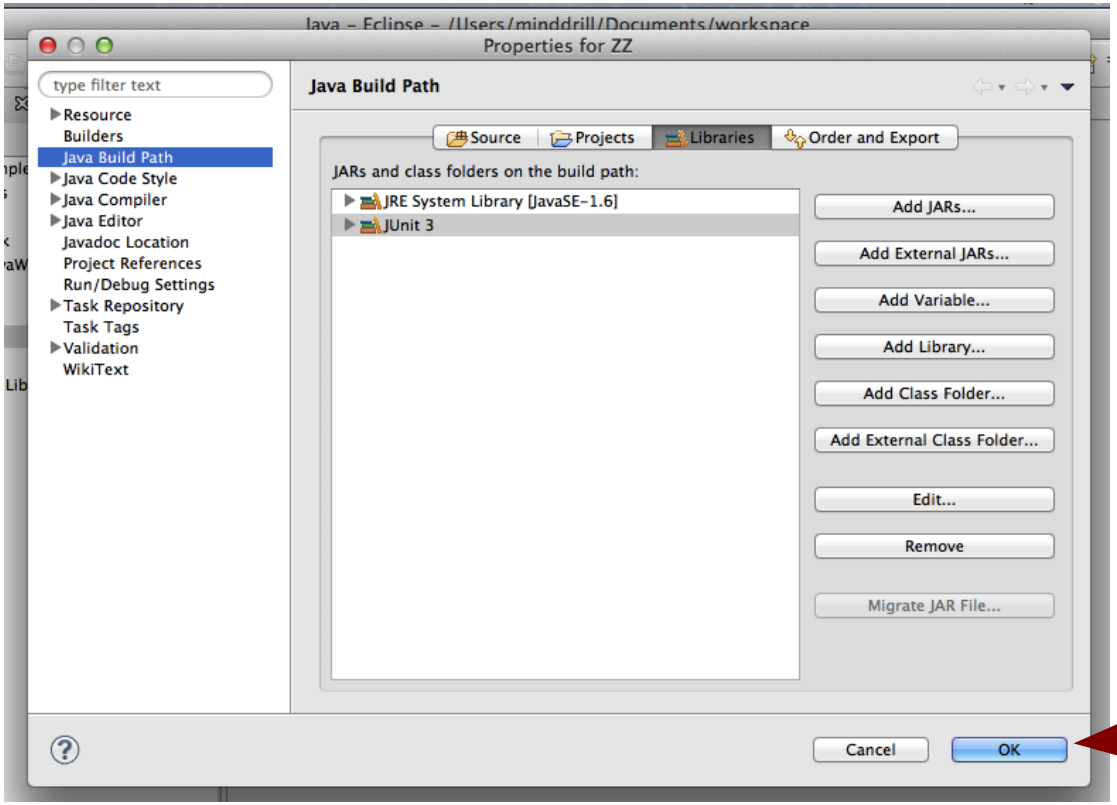


Select *JUnit*

Click *Next*

Click *Finish*

Returning to Package Explorer

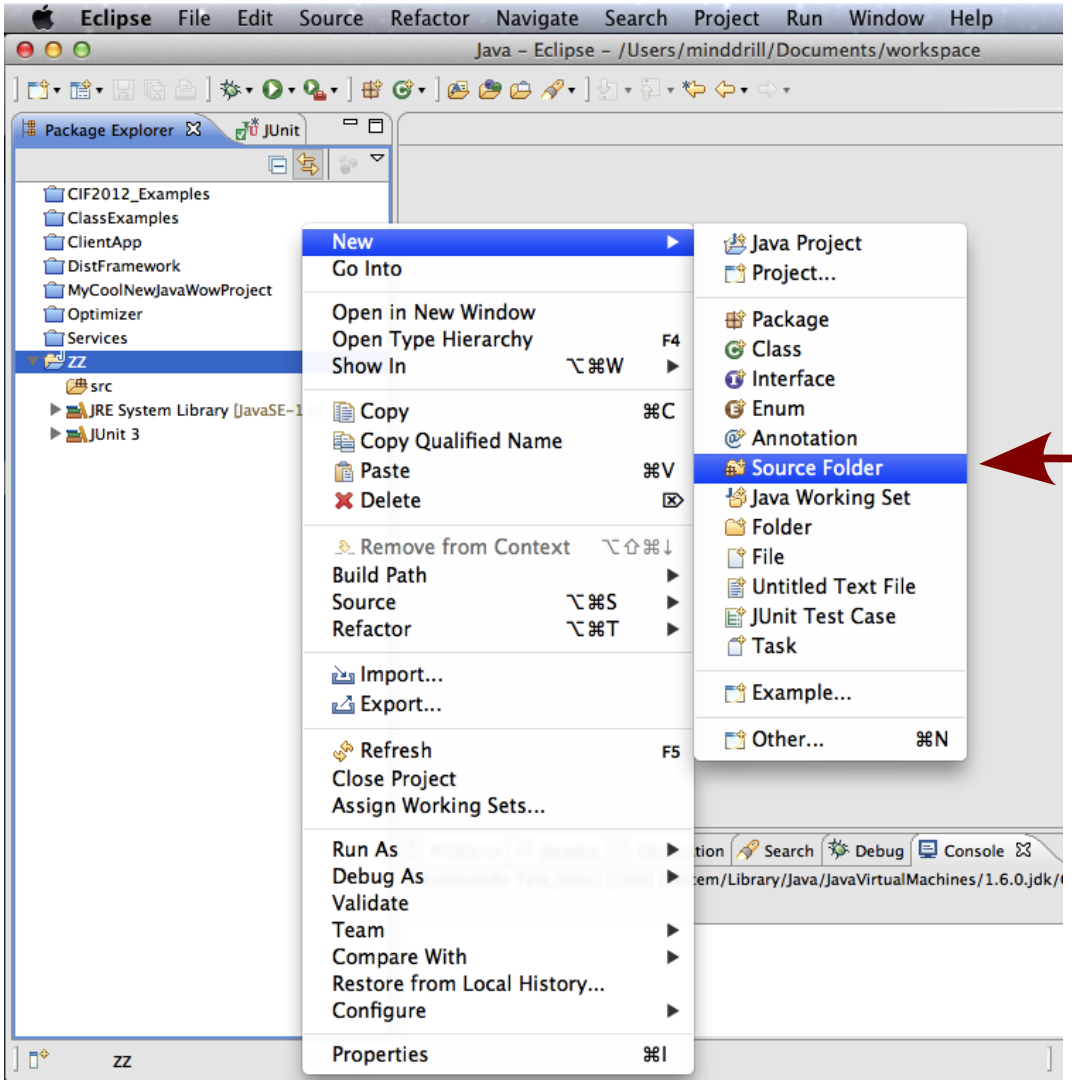


Click on *OK*

Why do we need two source folders?

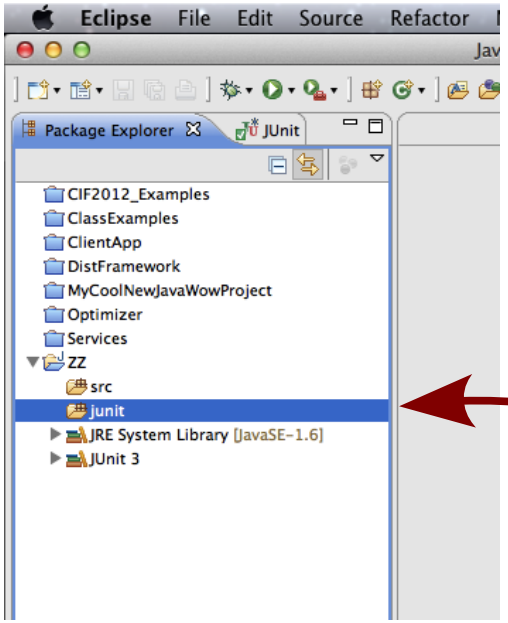
- We want to keep our tests separate from the rest of our code so we can remove the tests when we distribute a new version of our system
- However, we want packages in the *src* directory to correspond to packages in the *junit* directory so we can access both public and protected methods and variables
  - If a package in the junit directory is called *personPkg* and a package in the *src* directory is also called *personPkg*, all classes in the *personPkg* – whether they are in the *junit* source folder or the *src* source folder – will have access to each other's public and protected variables and methods
  - This may be necessary to properly test the functionality of all classes

Creating Source Folders



Create a new source folder for our JUnit tests. We'll call it *junit*

Our package explorer after changes

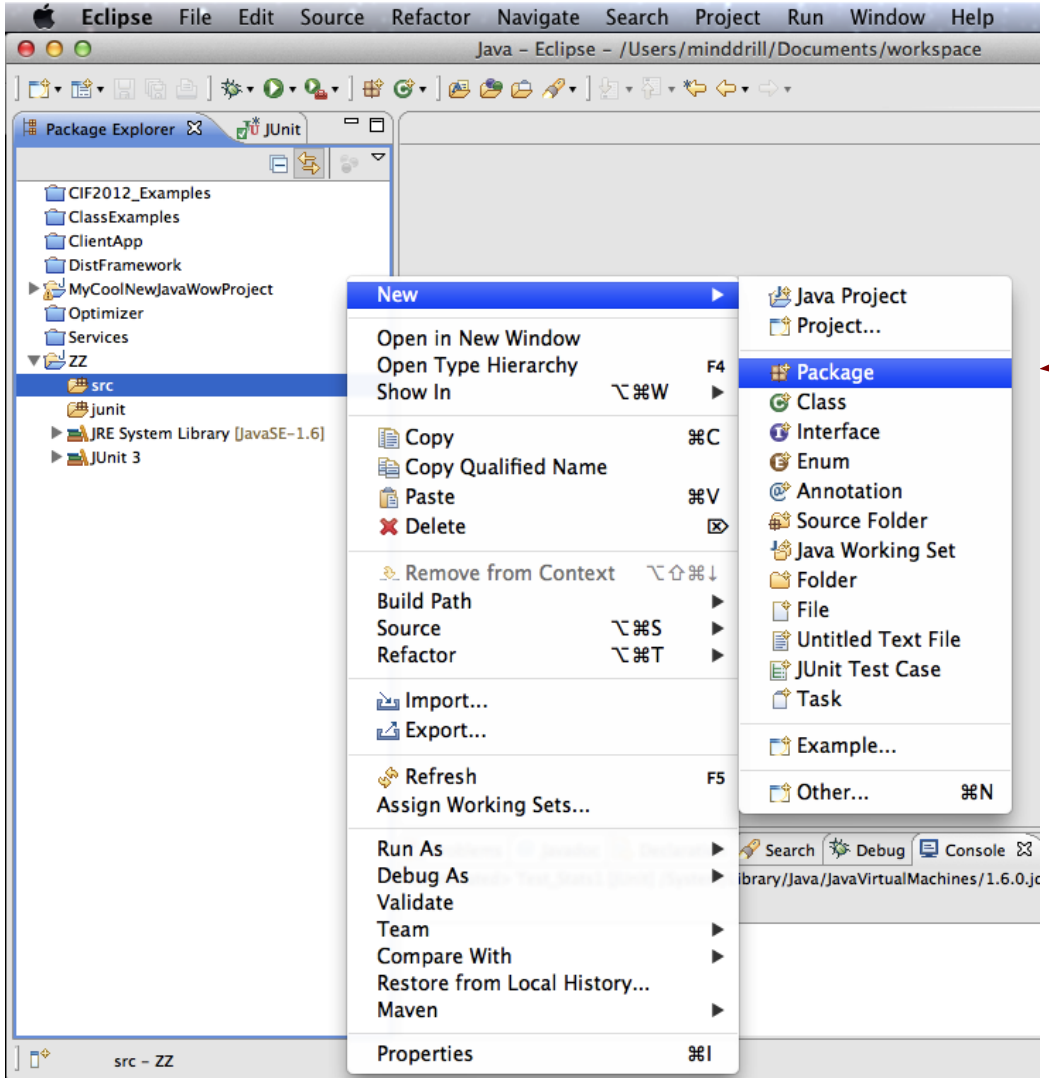


Note the new source folder

Writing Our First Testable Class

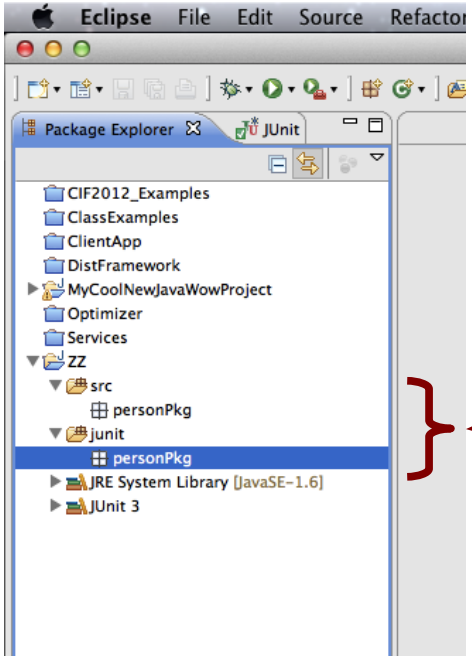
- We are going to write a class, *Person*
  - A *Person* will have the following instance variables
    - A *name* – defined as a String
    - An *age* – defined as an int
  - A *Person* will be able to tell us his/her name and his/her age, and will be able to say whether his/her name and age are the same as that of another person
- We want to write this class using Test Driven Development
- First, we create a package for this class and its tests

Creating our package



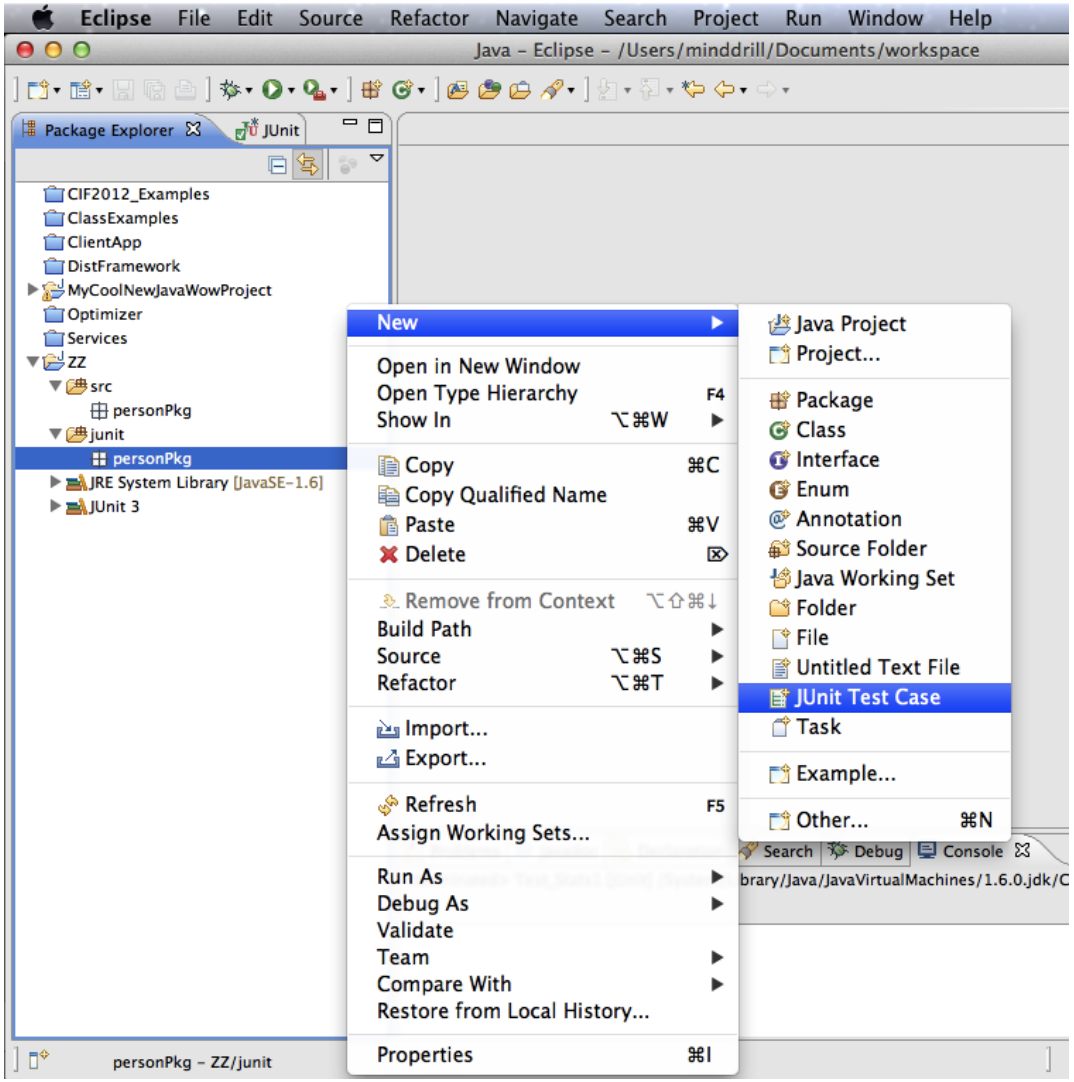
Create a new package called *personPkg*. Do the same for the *junit* source folder

What you should see

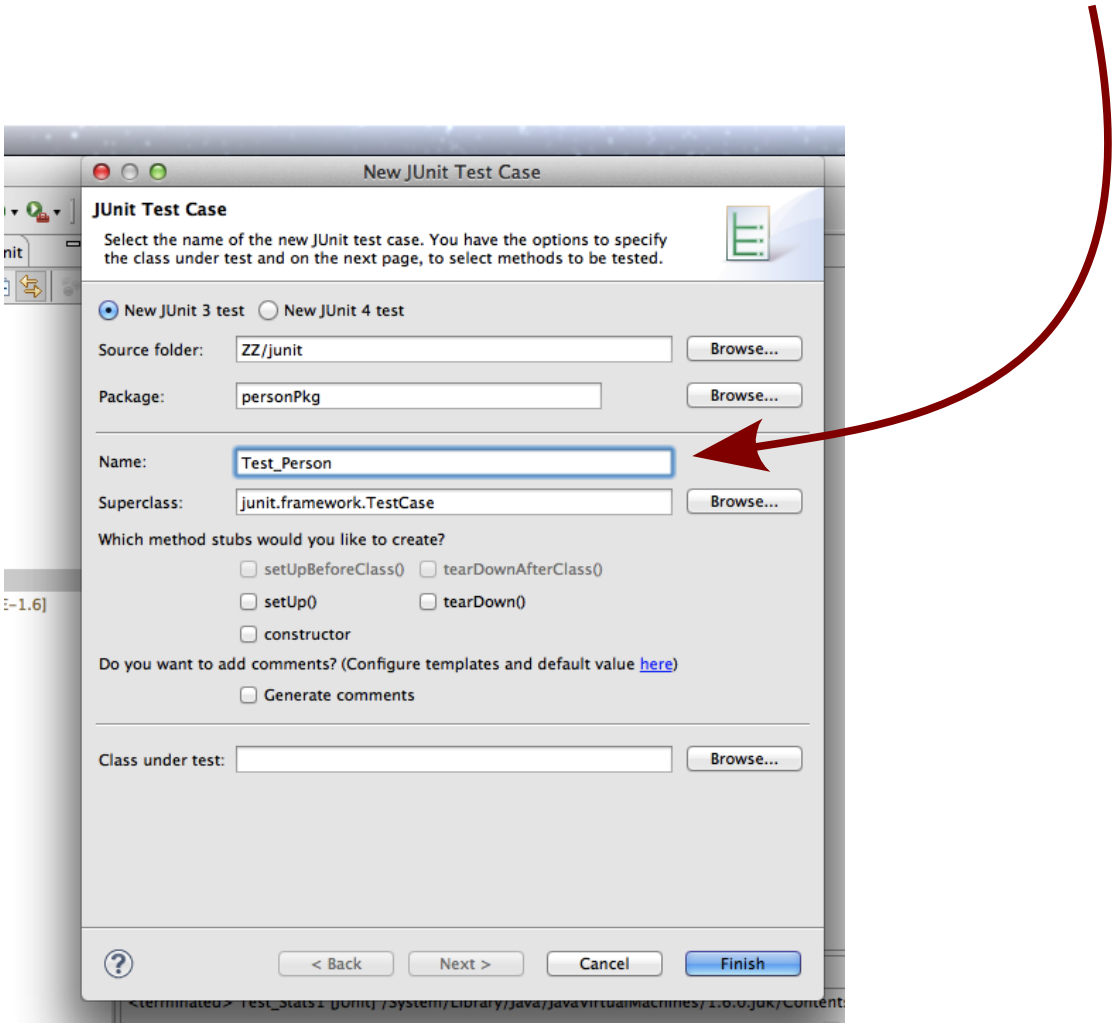


Both source folders should have the same packages

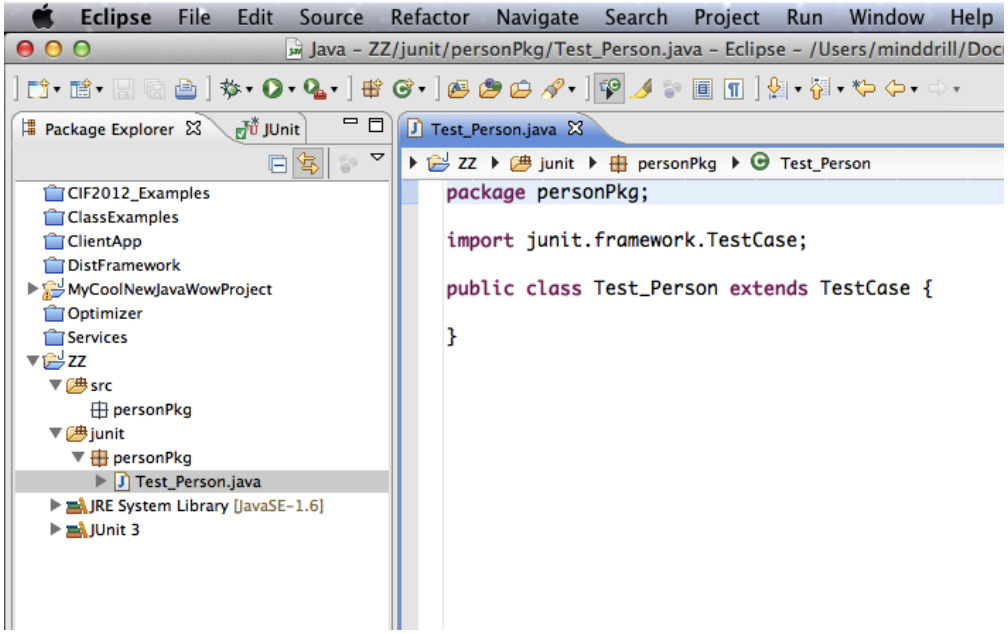
Write JUnit for Person



Call this JUnit *Test_Person*



What you should see



Add code to test *Person* constructor

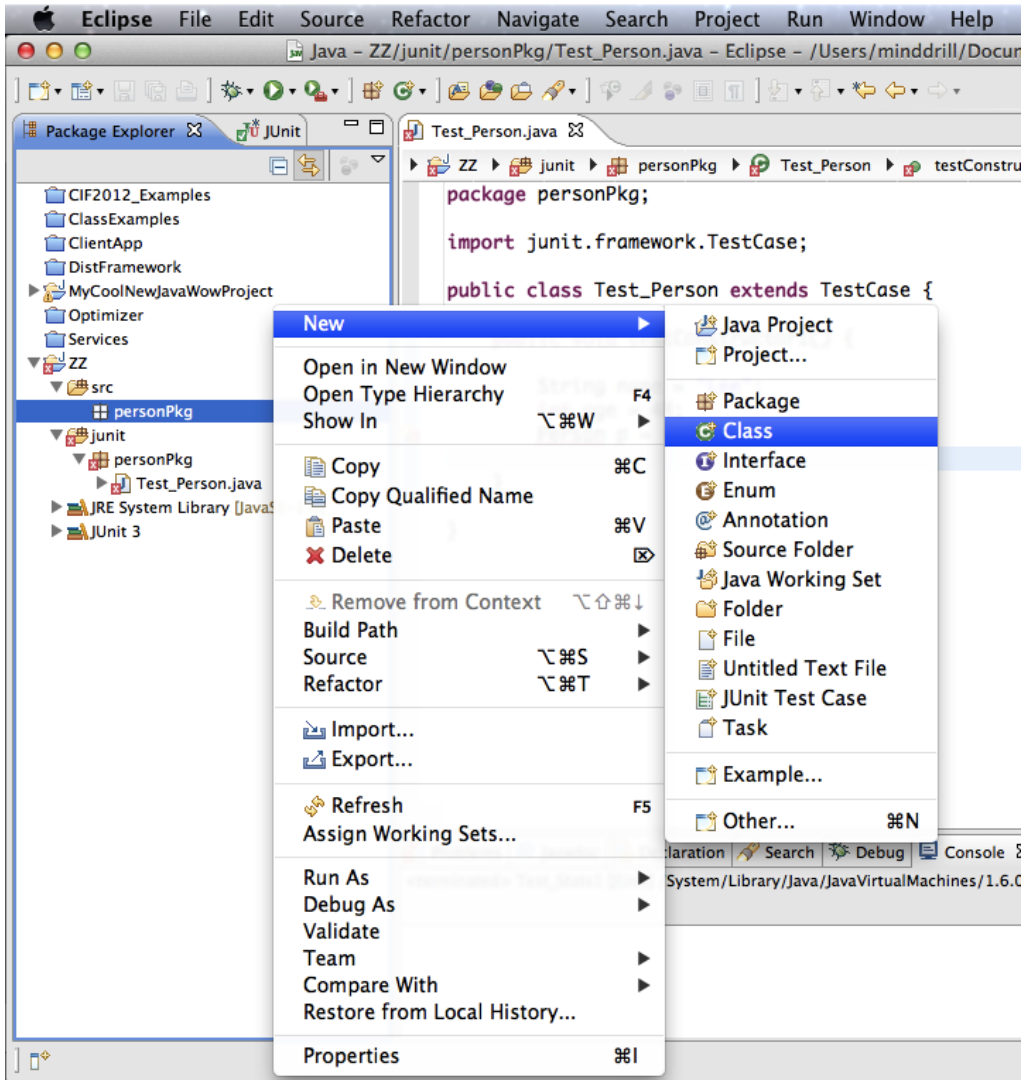
```
package personPkg;

import junit.framework.TestCase;

public class Test_Person extends TestCase {

    public void testConstructor1() {

        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );

    }

}
```

This class is derived from the class TestCase, which is part of the JUnit framework

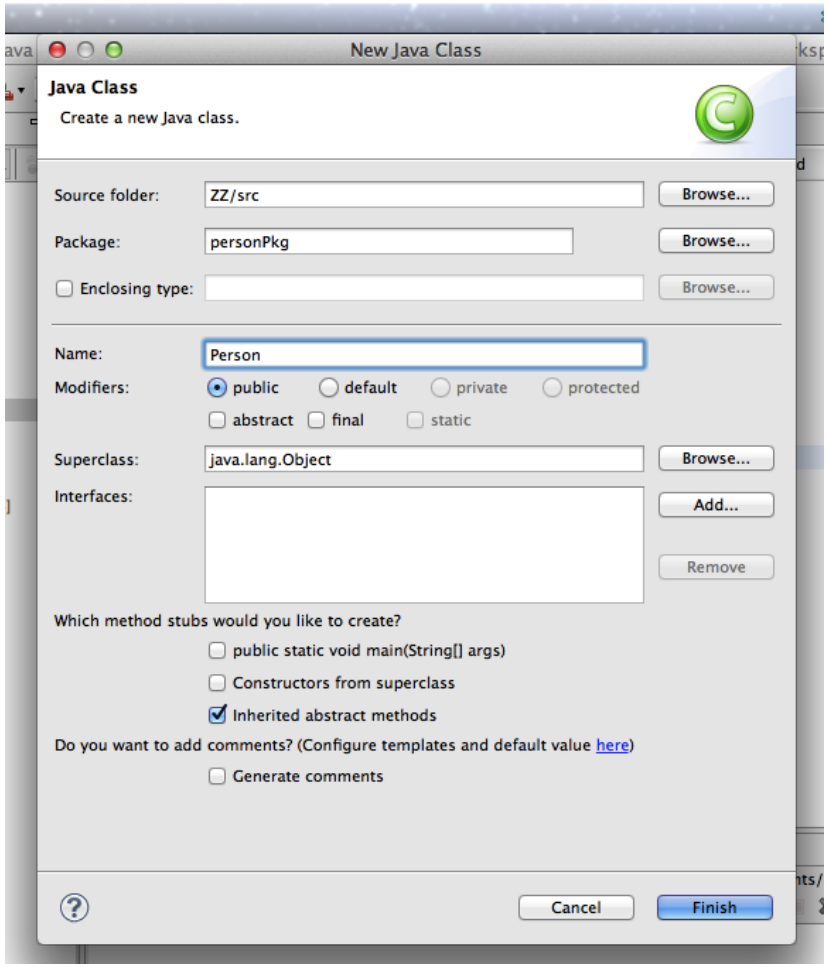
The JUnit framework will automatically run all methods in this class that start with the word *test*

We are testing whether we can create an object of the class *Person* but the IDE is reporting that the class doesn't exist. That's the first thing we have to fix.

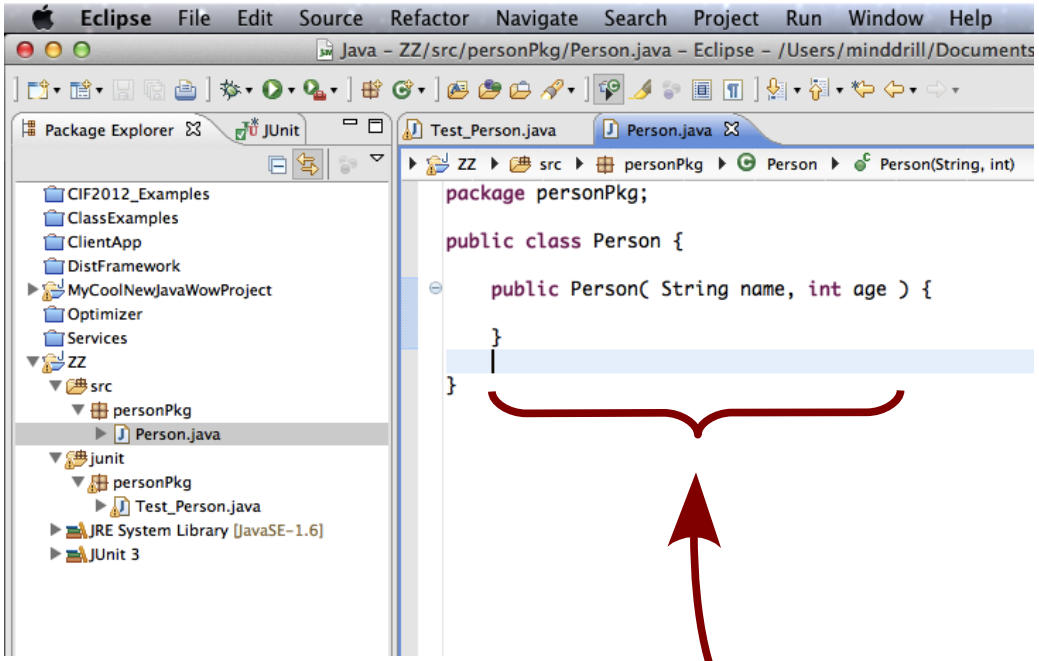
Create class *Person*



Create class *Person*

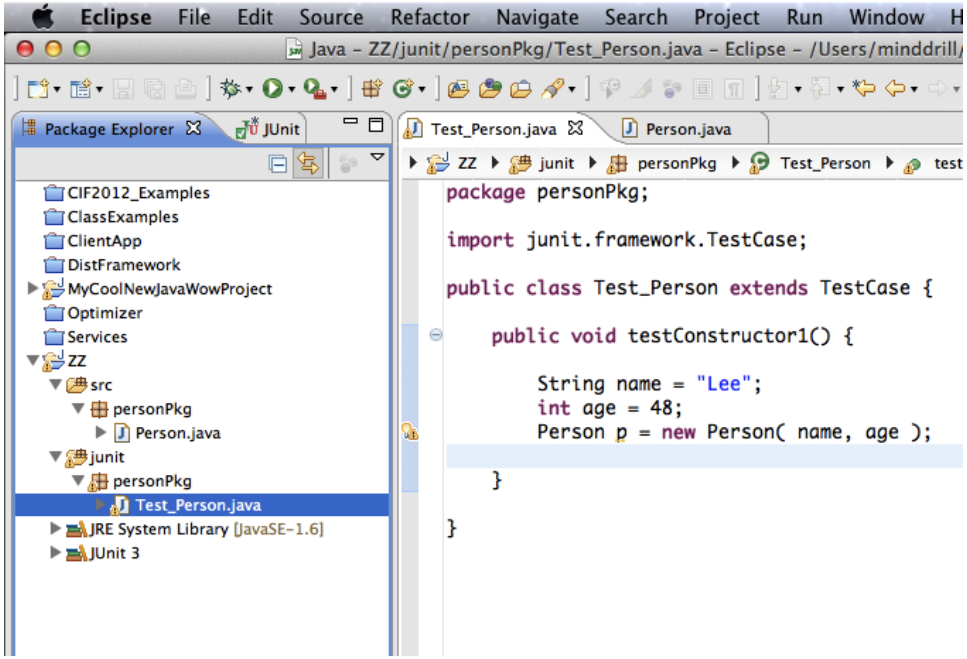


Implement constructor

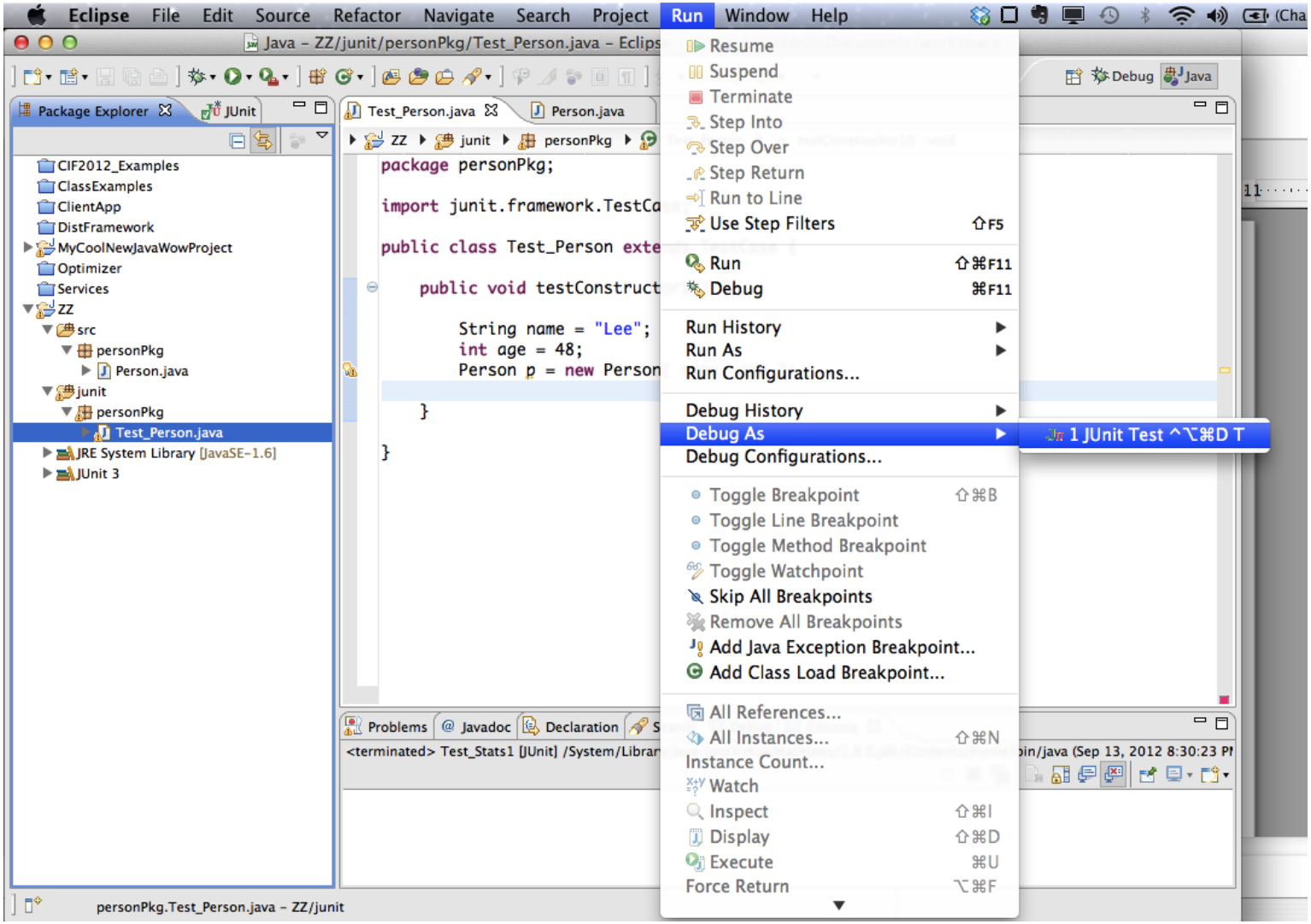


Implement no more than is necessary to pass the test!

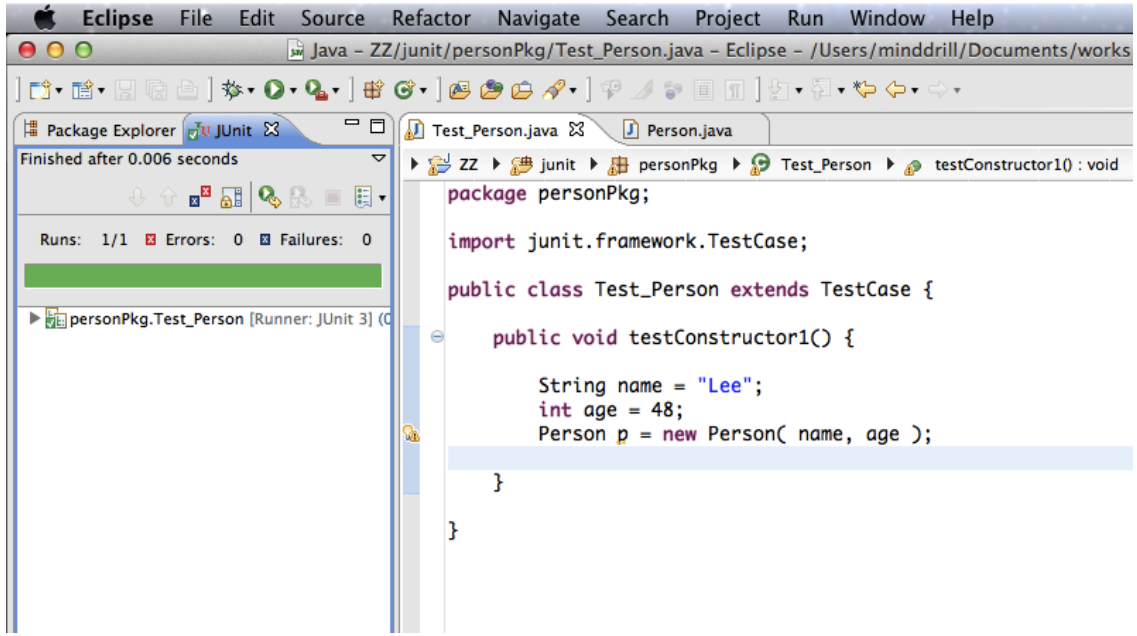
Back in our test, the errors are gone



We can now run our test



Green line means all tests passed



Implement test for name and age

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help

Java – ZZ/junit/personPkg/Test_Person.java – Eclipse – /Users/minddrill/Docume

Package Explorer | JUnit

- CIF2012_Examples
- ClassExamples
- ClientApp
- DistFramework
- MyCoolNewJavaWowProject
- Optimizer
- Services
- ZZ
  - src
    - personPkg
      - Person.java
  - junit
    - personPkg
      - Test_Person.java
  - JRE System Library [JavaSE-1.6]
  - JUnit 3

Test_Person.java | Person.java

ZZ ▸ junit ▸ personPkg ▸ Test_Person ▸ testGetAge() :

```
import junit.framework.TestCase;

public class Test_Person extends TestCase {

    public void testConstructor1() {

        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );

    }

    public void testGetName() {
        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );
        assertEquals( p.getName(), name );
    }

    public void testGetAge() {
        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );
        assertEquals( p.getAge(), age );
    }

}
```

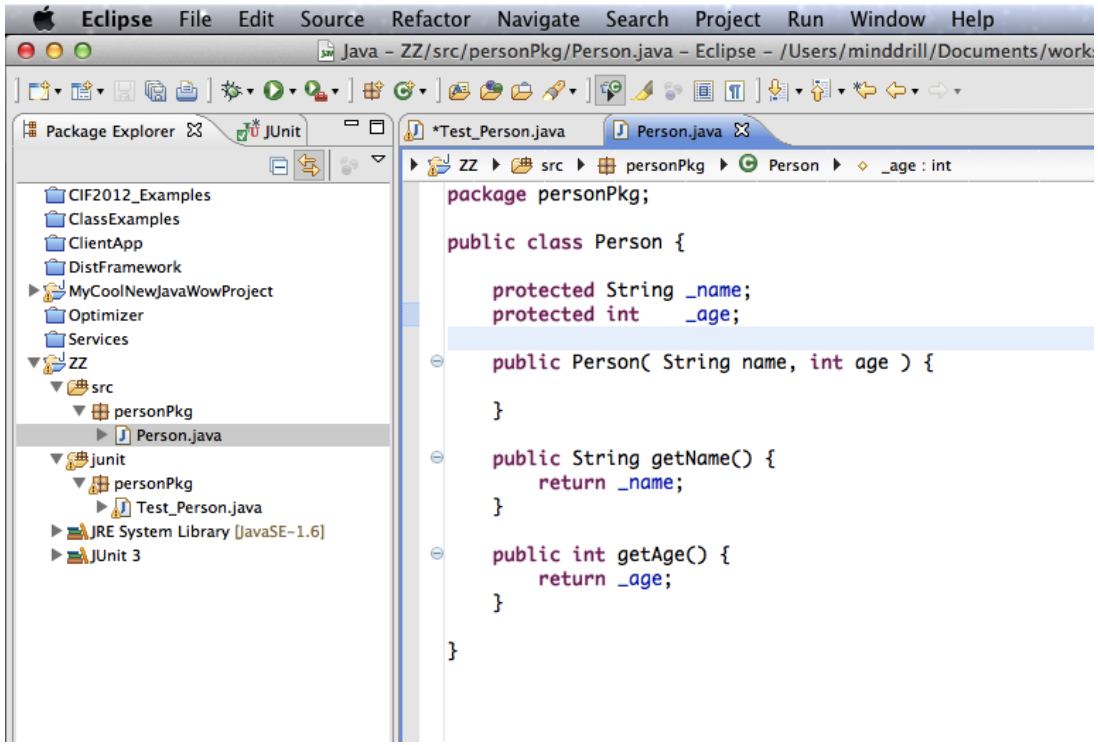
Problems | Javadoc | Declaration | Search | Debug | Console

<terminated> Test_Person (3) [JUnit] /System/Library/Java/JavaVirtualMachines/1.6.

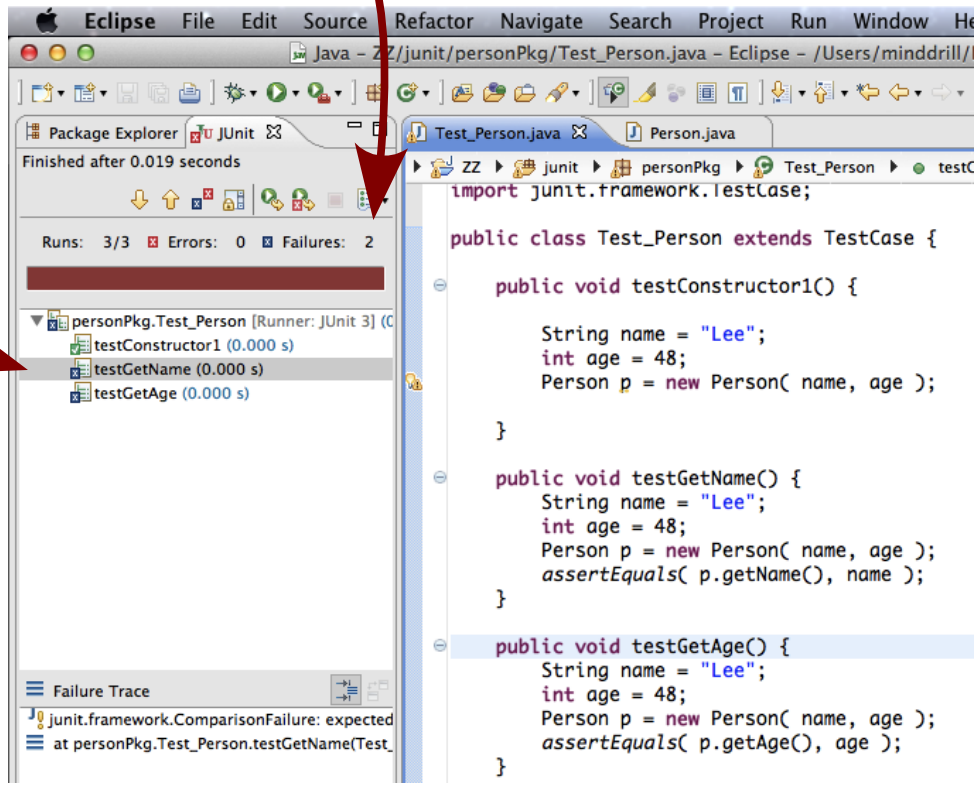
IDE reports that neither method exists, so we have to implement them

Note the use of assert methods, which are part of the TestCase class from which our test class inherits functionality

Implement methods to get name and age

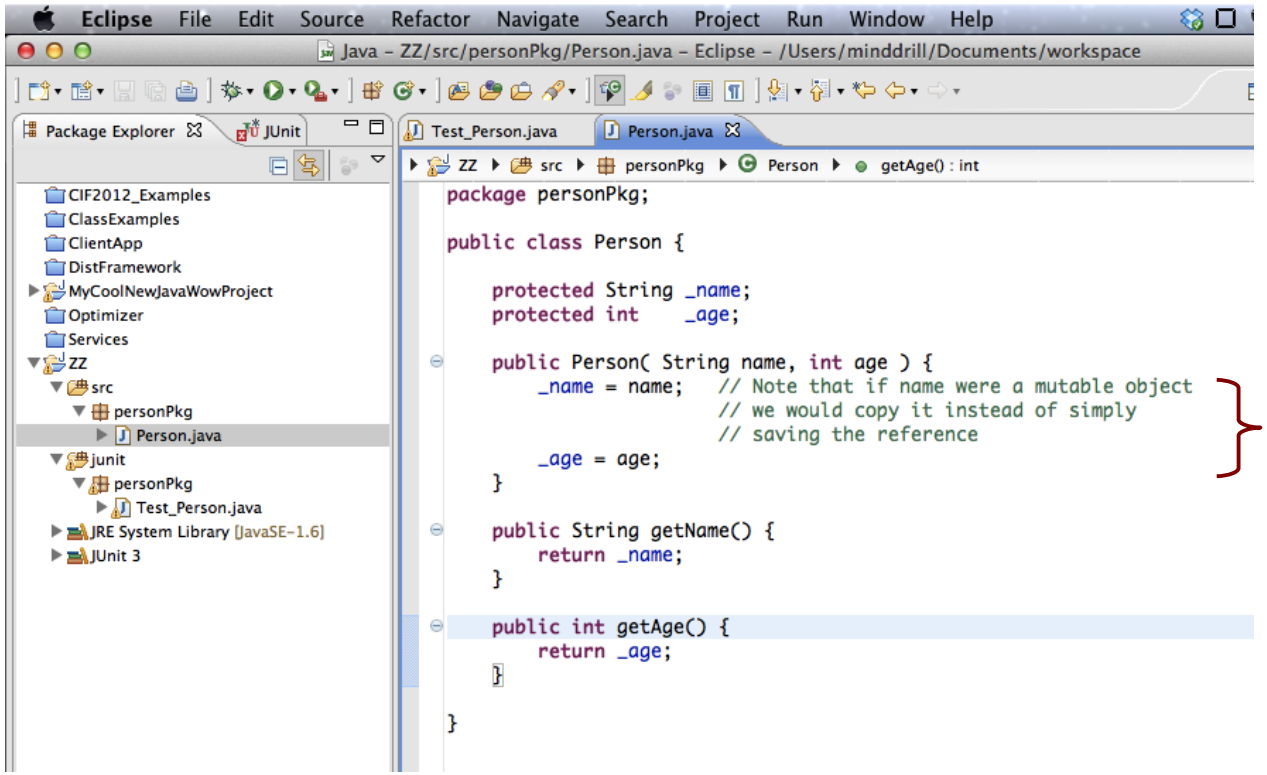


Running tests produces two failures

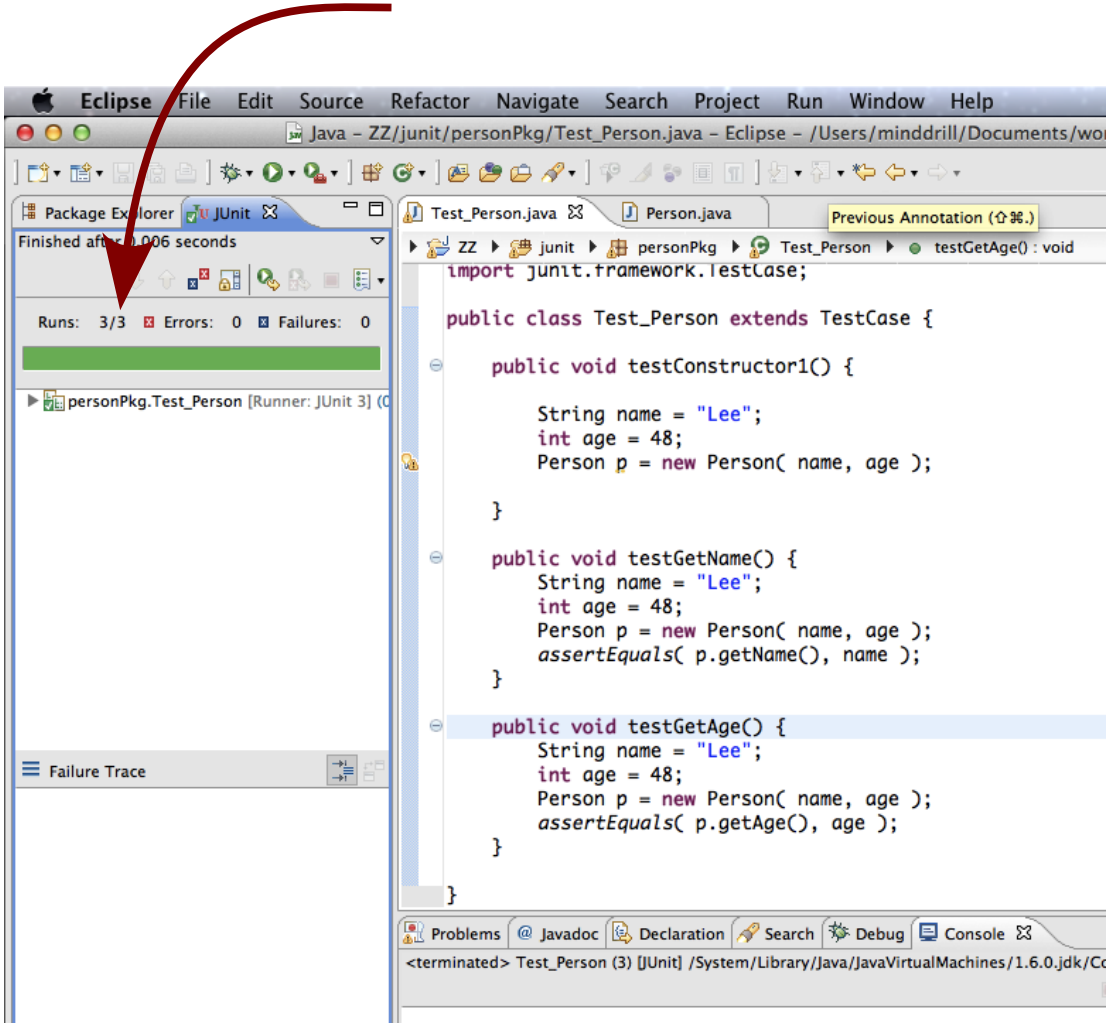


Note that execution did not stop when test 2 failed. All tests were run and the results were reported.

We correct these failures



...and now the tests pass



Default *equals* method fails

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help

Java – ZZ/junit/personPkg/Test_Person.java – Eclipse – /Users/minddrill/Documents/workspace

Package Explorer | JUnit

Finished after 0.012 seconds

Runs: 4/4 Errors: 0 Failures: 1

personPkg.Test_Person [Runner: JUnit 3] (C
- testConstructor1 (0.000 s)
- testGetName (0.000 s)
- testGetAge (0.000 s)
- testEquals (0.000 s)

Failure Trace

junit.framework.AssertionFailedError

at personPkg.Test_Person.testEquals(Test_Pe

Test_Person.java | Person.java

ZZ ▸ junit ▸ personPkg ▸ Test_Person ▸ testEquals() : void

```
    public void testGetName() {
        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );
        assertEquals( p.getName(), name );
    }

    public void testGetAge() {
        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );
        assertEquals( p.getAge(), age );
    }

    public void testEquals() {
        String name = "Lee";
        int age = 48;
        Person p = new Person( name, age );
        String name1 = "Lee";
        int age1 = 48;
        Person p1 = new Person( name1, age1 );
        assertTrue( p.equals( p ) );    // Without our own equals, this should pass
        assertTrue( p.equals( p1 ) );   // ..but this should fail
    }

}
```

Problems | Javadoc | Declaration | Search | Debug | Console

<terminated> Test_Person (3) [JUnit] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Sep 19, 2012 10:10

The default *equals* method – the one that is derived from Object – tests for equality of identity, not equality of value. So if we want to test for equality of value, we must write our own *equals* method.

Our own *equals* method

Eclipse File Edit Source Refactor Navigate Search Project Run Window Help

Java – ZZ/src/personPkg/Person.java – Eclipse – /Users/minddrill/Documents/workspace

Package Explorer | JUnit

- CIF2012_Examples
- ClassExamples
- ClientApp
- DistFramework
- MyCoolNewJavaWowProject
- Optimizer
- Services
- ZZ
  - src
    - personPkg
      - Person.java
  - junit
    - personPkg
      - Test_Person.java
  - JRE System Library [JavaSE-1.6]
  - JUnit 3

Test_Person.java | Person.java

ZZ ▸ src ▸ personPkg ▸ Person ▸ equals(Object) : boolean

```
        _name = name;    // Note that if name were a mutable object
                         // we would copy it instead of simply
                         // saving the reference
        _age = age;
    }

    public String getName() {
        return _name;
    }

    public int getAge() {
        return _age;
    }

    public boolean equals( Object object ) {
        if( this == object ) {
            return true;
        }
        if( ! ( object instanceof Person ) ) {
            return false;
        }
        Person p = (Person) object;
        return ( p.getName().equals( _name ) ) && ( p.getAge() == _age );
    }

}
```

Problems | Javadoc | Declaration | Search | Debug | Console

<terminated> Test_Person (3) [JUnit] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Sep 19, 201