Object Oriented Programming

Week 1 Part 3
Writing Java with Eclipse and JUnit

Today's Lecture

- Test Driven Development Review (TDD)
- Building up a class using TDD

Adding a Class using Test Driven Development in Eclipse

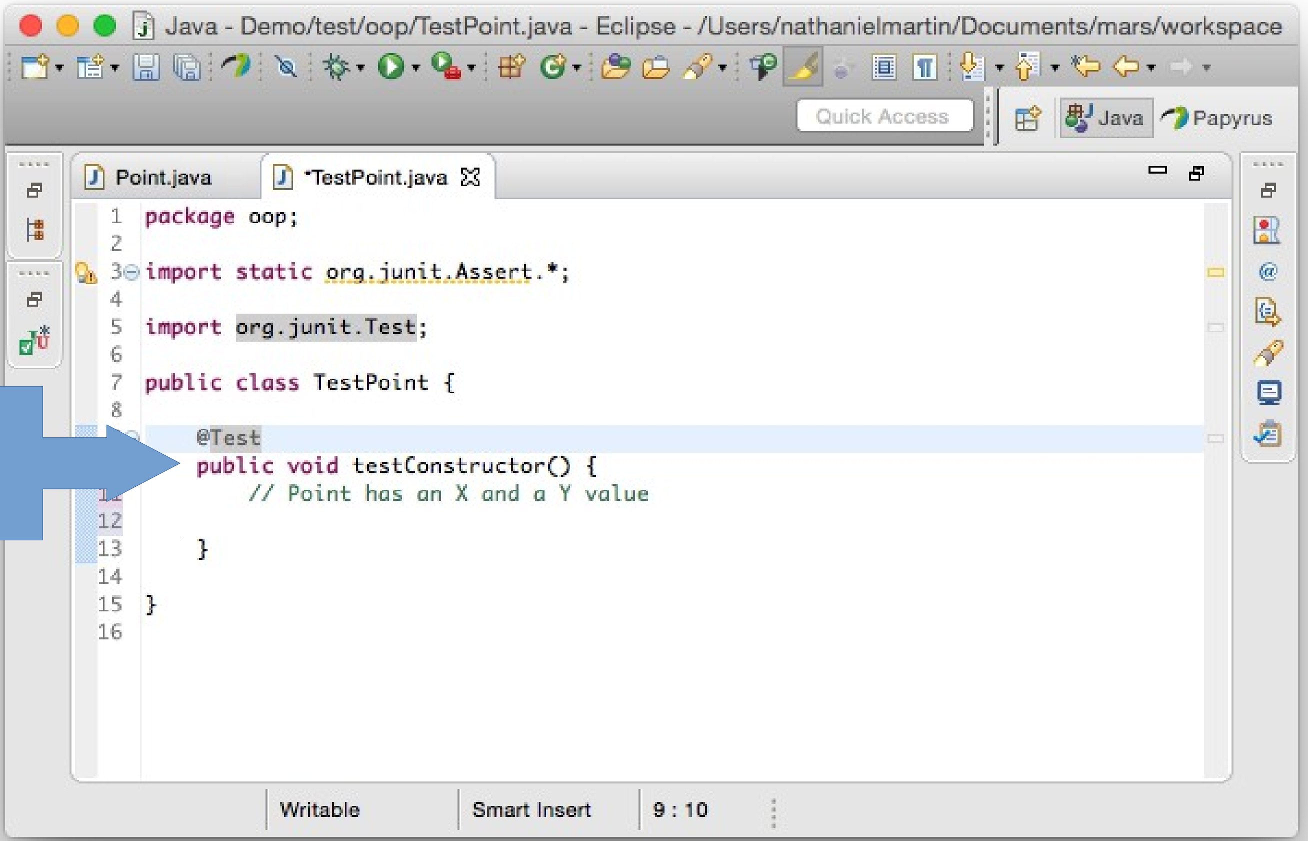
Test Driven Development

1. Write a test
2. See test fail
3. Write code
4. See test succeed
5. Refactor code

Add a test to the test case

Test the constructor

- @Test pragma needed
- Method begins with “test”

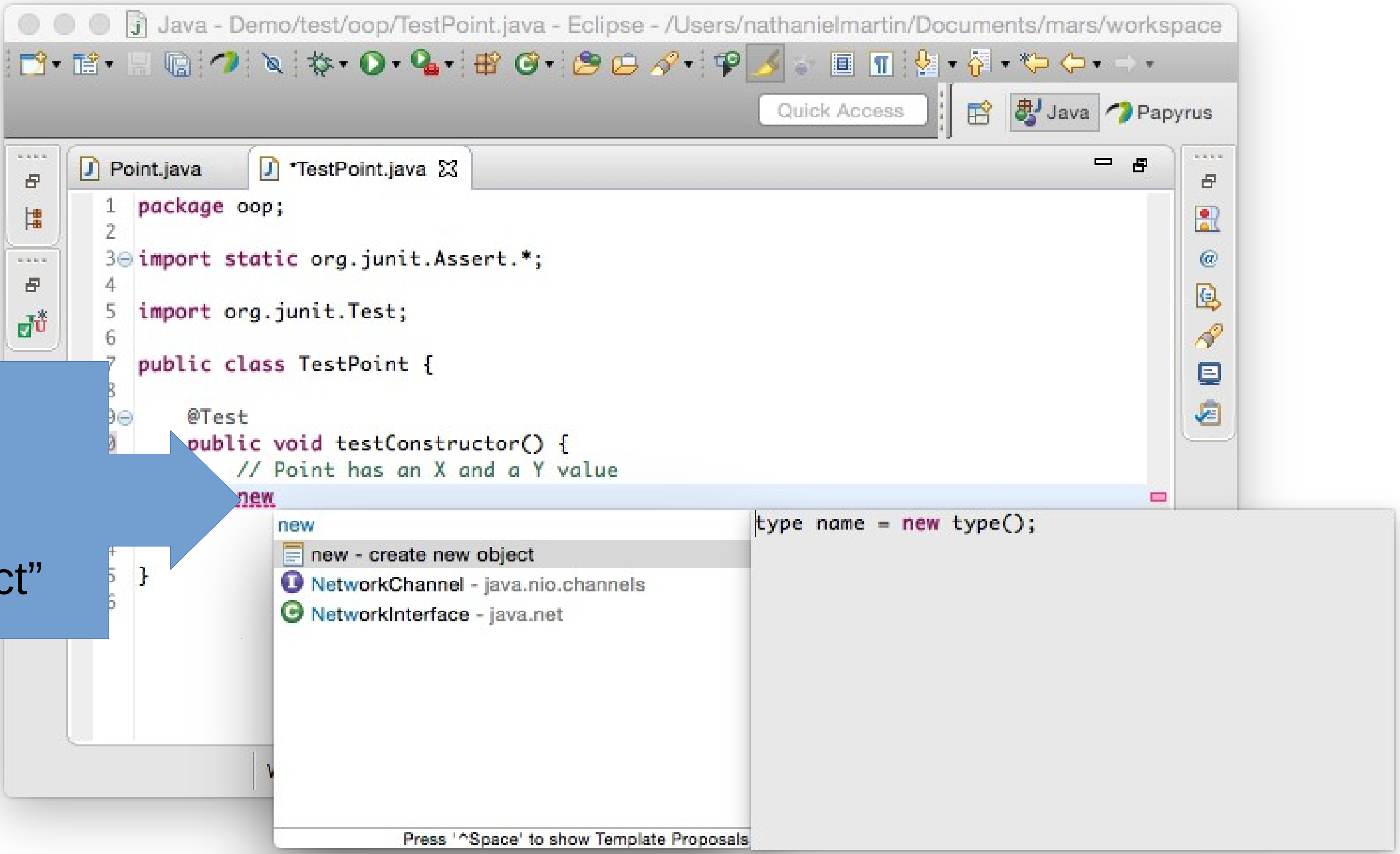

Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace
Quick Access
Java
Papyrus
Point.java
*TestPoint.java
Writable
Smart Insert
9 : 10

```
package oop;

import static org.junit.Assert.*;

import org.junit.Test;

public class TestPoint {

    @Test
    public void testConstructor() {
        // Point has an X and a Y value

    }

}
```

Eclipse Shortcuts (Ctrl-<space>)

- Control-<space> adds code
  - E.g., If you type new Control-space, it offers to add a new object
    - If you select “Create object” it will put in the code *“type name* = new *type*();
    - You fill in the type, and it puts the same type in as the constructor
    - You fill in the name
    - You can add parameters to the constructor

Add the constructor



Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
package oop;

import static org.junit.Assert.*;

import org.junit.Test;

public class TestPoint {

    @Test
    public void testConstructor() {
        // Point has an X and a Y value
        new
```

Quick Access
Java
Papyrus
Point.java
*TestPoint.java
new
new - create new object
NetworkChannel - java.nio.channels
NetworkInterface - java.net
Press '^Space' to show Template Proposals

```
type name = new type();
```

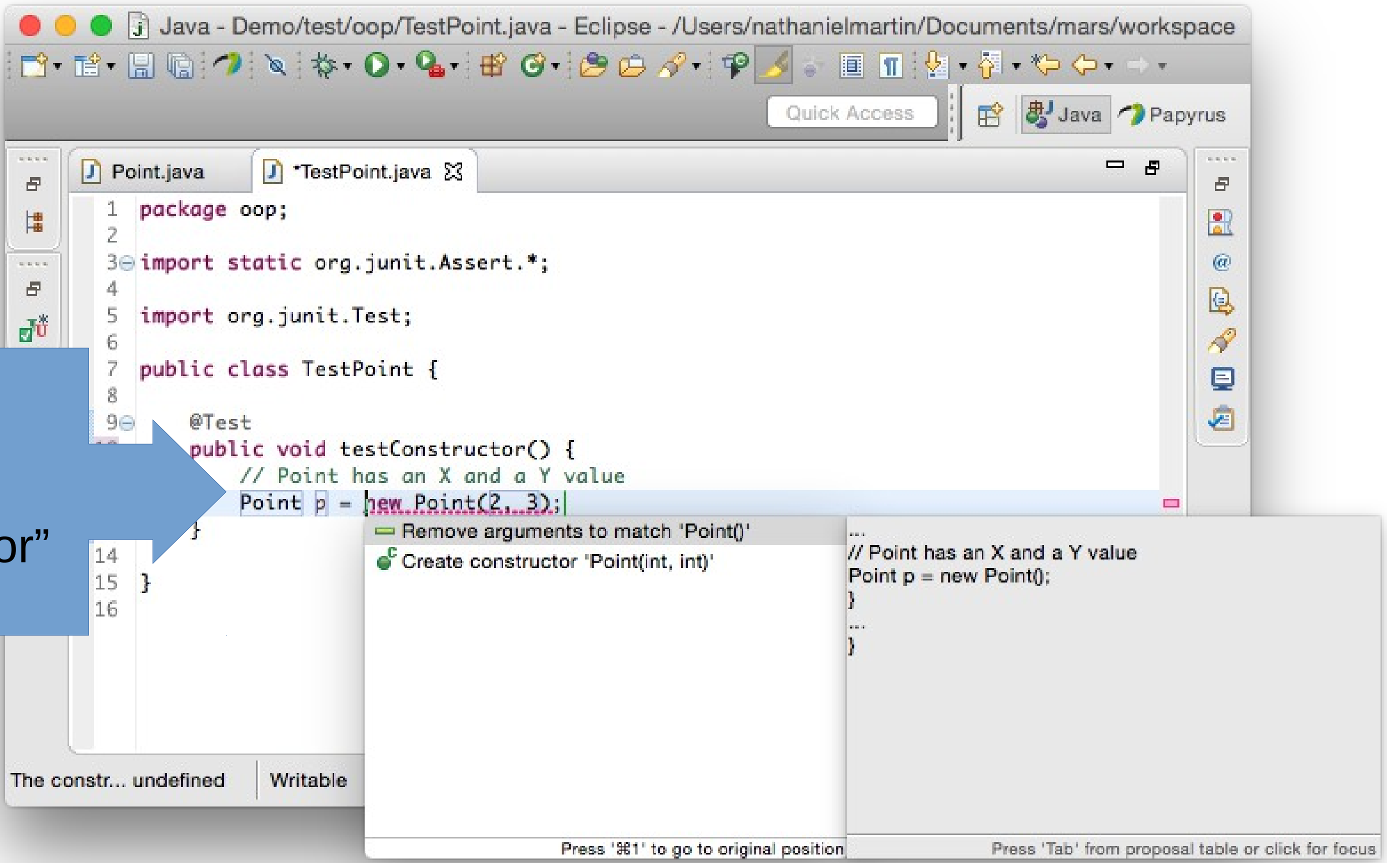
Create the constructor

- Type "new"
- Hit Ctrl-<space>
- Select "create new object"

Eclipse Shortcuts (Ctrl-1)

- Control-1 suggests solutions to compiler errors
  - It will offer to add a constructor when one does not exist
  - It takes to the the file in which the class is defined and puts in a template for the constructor

Fix Compiler Error: no Constructor



Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
package oop;

import static org.junit.Assert.*;

import org.junit.Test;

public class TestPoint {

    @Test
    public void testConstructor() {
        // Point has an X and a Y value
        Point p = new Point(2, 3);
    }

}
```

Remove arguments to match 'Point()'
Create constructor 'Point(int, int)'

```
...
// Point has an X and a Y value
Point p = new Point();
}
...
}
```

Press '⌘1' to go to original position
Press 'Tab' from proposal table or click for focus

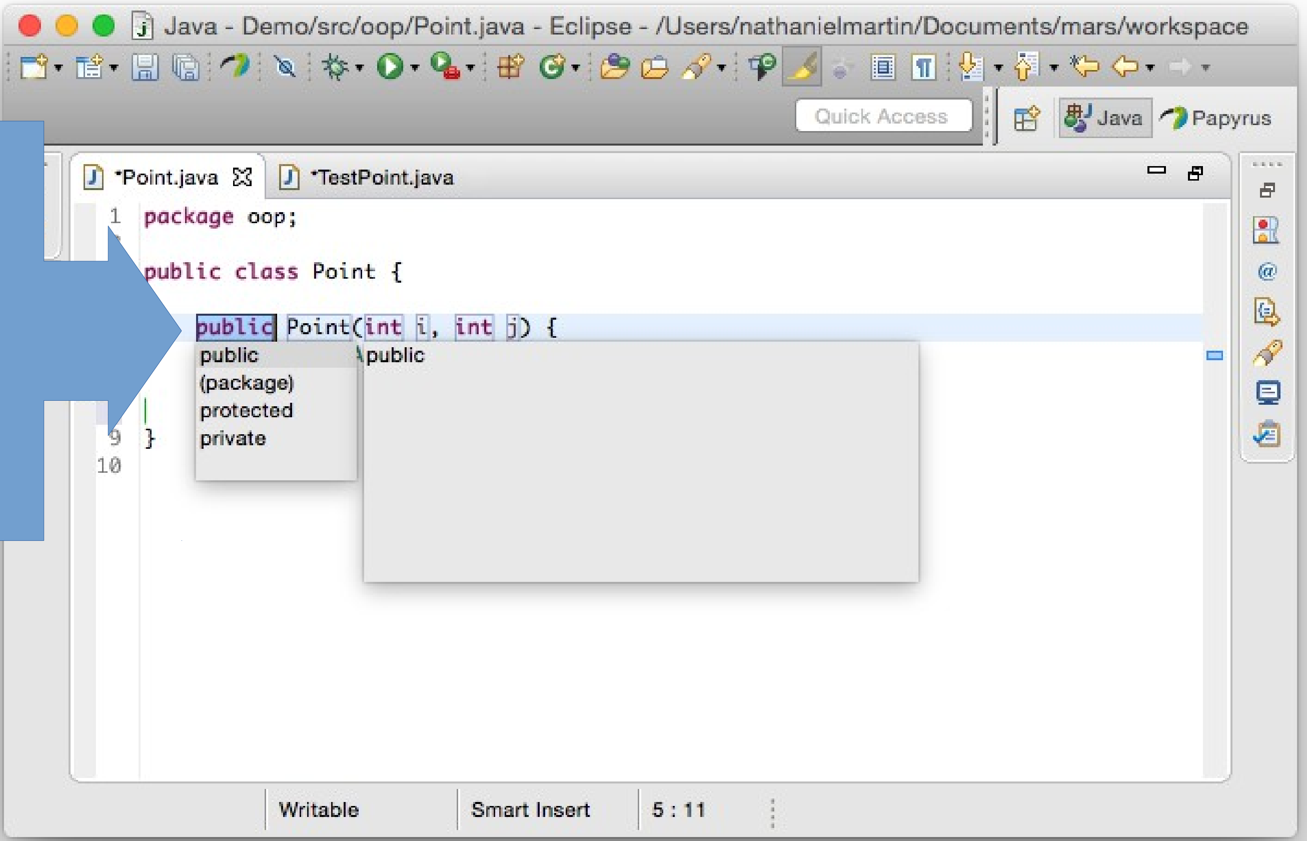
The constr... undefined | Writable

On the line with the error
- Hit Ctrl-1
- Select "Create Constructor"

Add Constructor

When you select
Create Constructor

- Opens class file for object
- Adds function definition for Constructor
- You can tab through to change fields



```
package oop;

public class Point {

    public Point(int i, int j) {

}
```

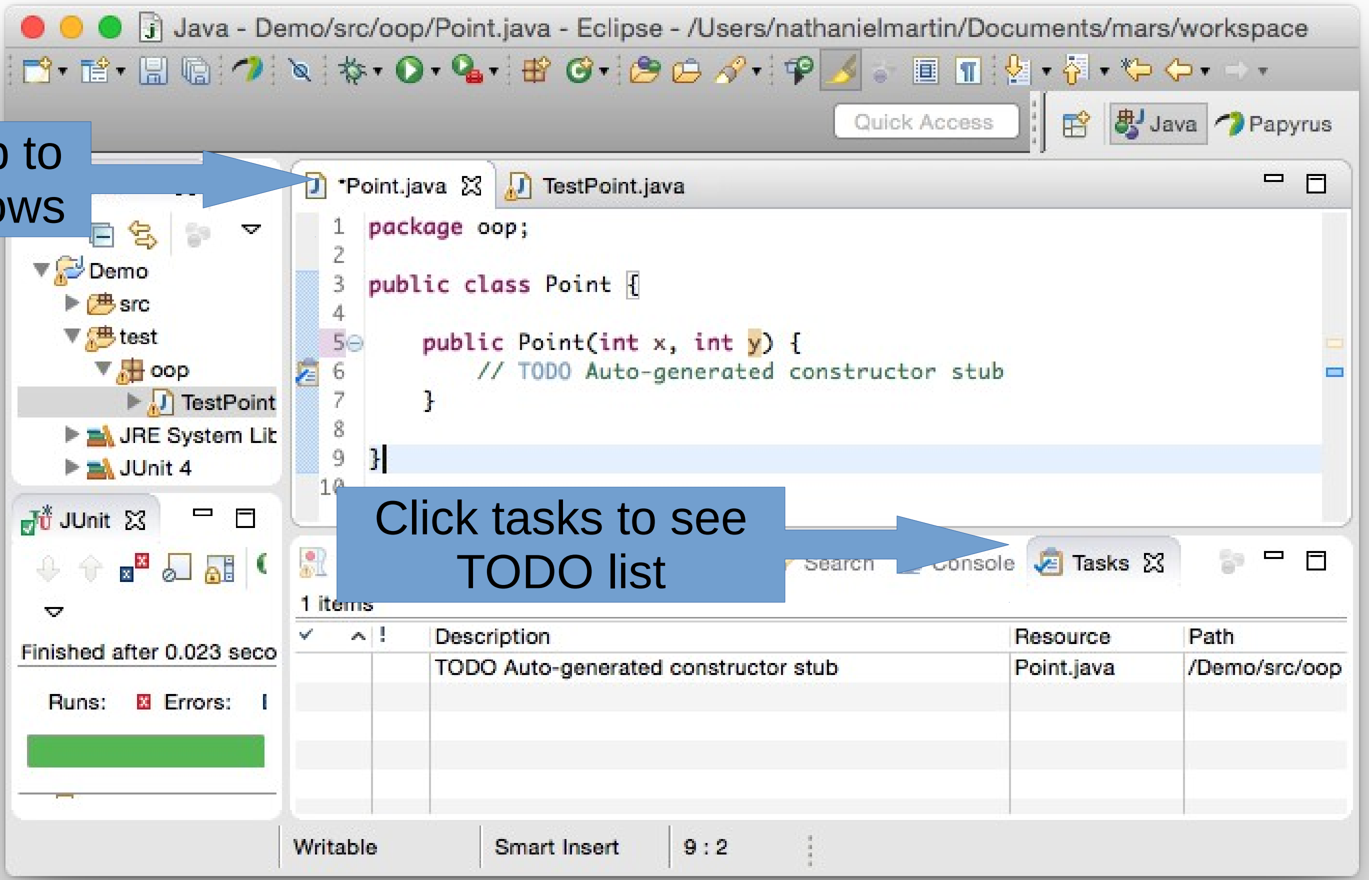
Eclipse TODO

- Eclipse keeps track of all of the lines that start with TODO
  - You can use this work to note places you are working
  - Eclipse puts them in automatically when it adds a function prototype for you.
    - E.g., It adds one when you add your constructor function

Added Constructor w/ TODO

Double click tab to see other windows

Click tasks to see TODO list



Java - Demo/src/oop/Point.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
package oop;

public class Point {

    public Point(int x, int y) {
        // TODO Auto-generated constructor stub
    }

}
```

1 items

✓	!	Description	Resource	Path
		TODO Auto-generated constructor stub	Point.java	/Demo/src/oop

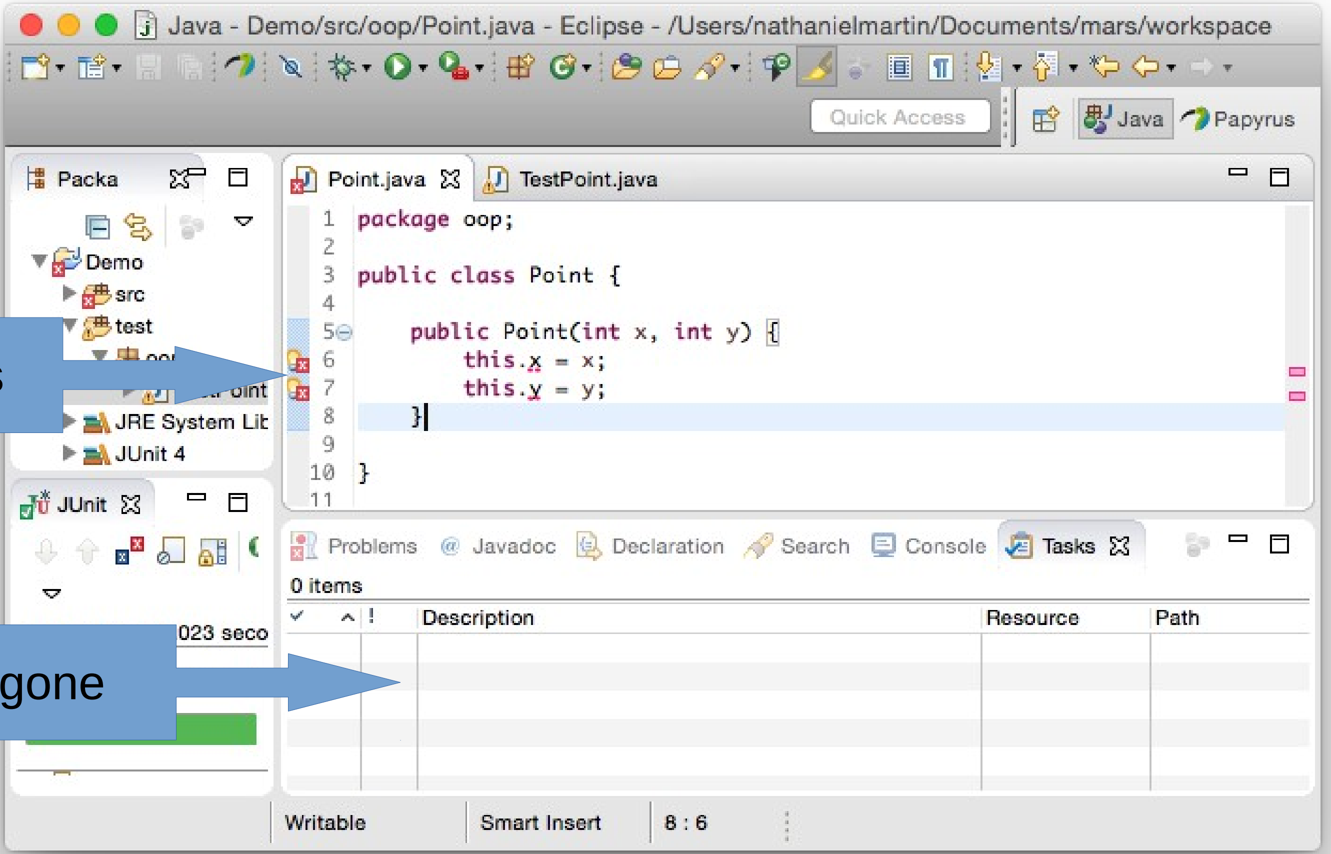
Finished after 0.023 seco

Writable | Smart Insert | 9 : 2

Add the Constructor Body

- The constructor for our point will put in the initial values for the x and y positions of the point.
  - That is `this.x = x;` and `this.y = y;`
    - The expression this.x refers to the instance variable x.
    - The expression x refers to the parameter.

Add Constructor Body

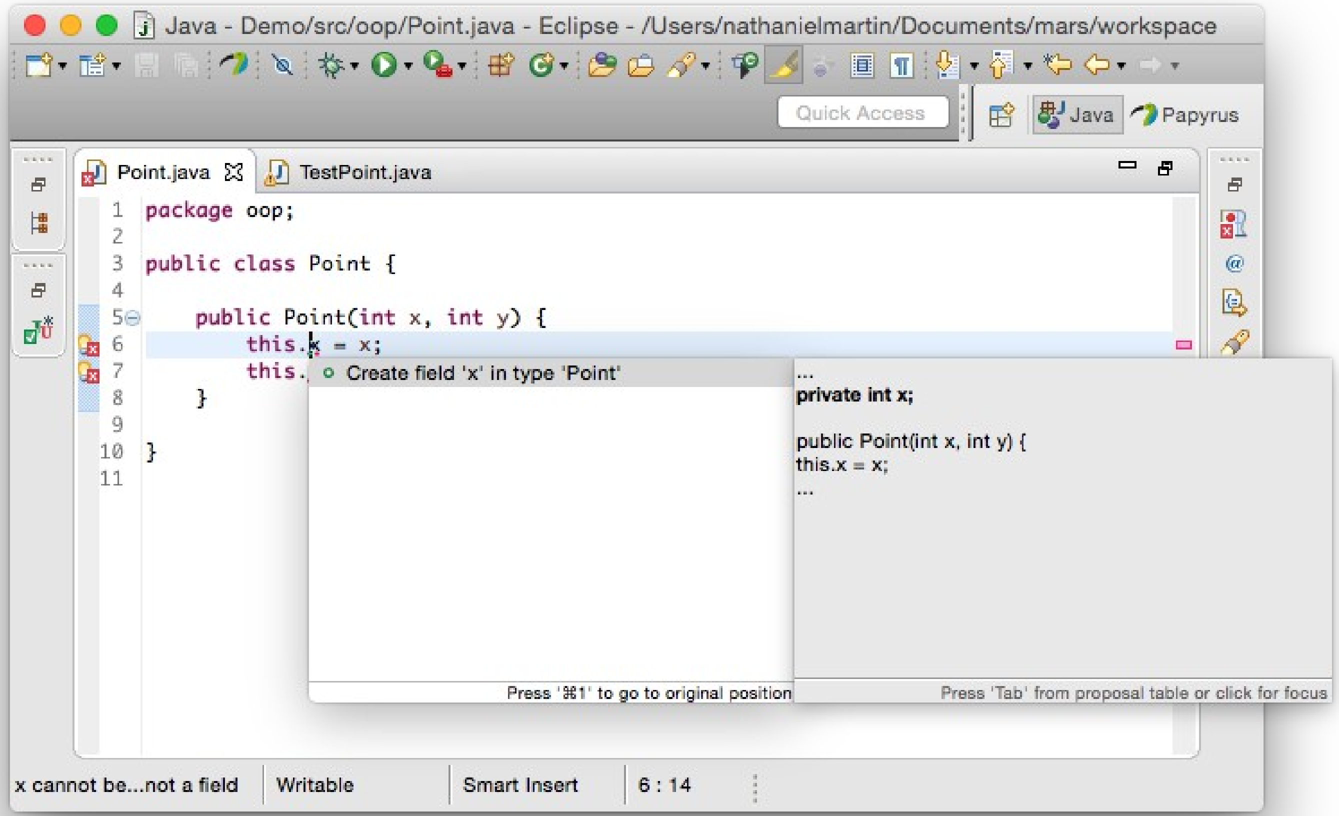


```
package oop;

public class Point {

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

}
```

Compiler errors

TODO is gone

Ctrl-1 to add x field


Java - Demo/src/oop/Point.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace
Quick Access
Java
Papyrus
Point.java
TestPoint.java

```
package oop;

public class Point {

    public Point(int x, int y) {
        this.x = x;
        this.
    }

}
```

Create field 'x' in type 'Point'

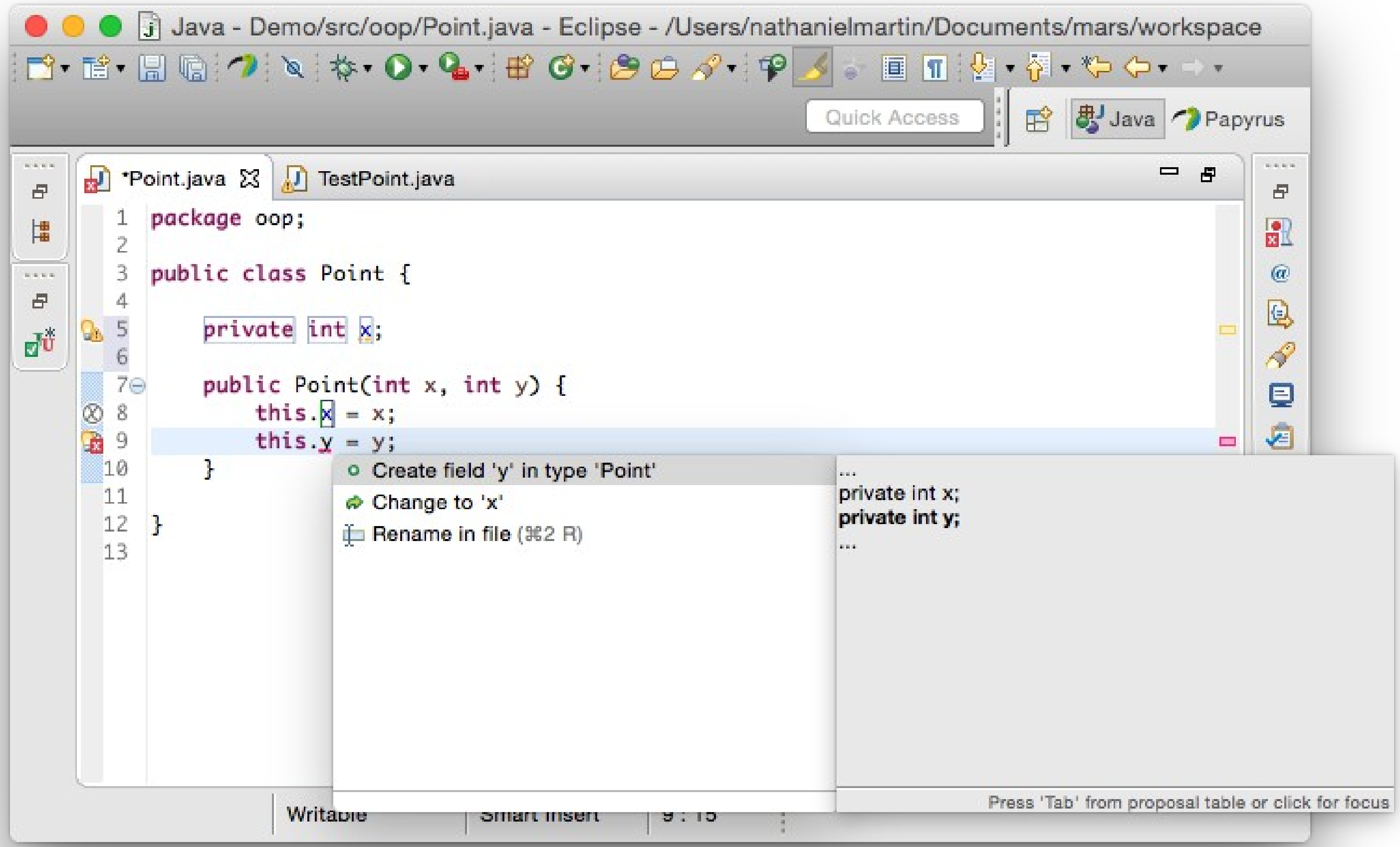
```
...
private int x;

public Point(int x, int y) {
this.x = x;
...
```

Press '⌘1' to go to original position

Press 'Tab' from proposal table or click for focus

x cannot be...not a field | Writable | Smart Insert | 6 : 14

Ctrl-1 again to add y field


Java - Demo/src/oop/Point.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace
Quick Access
Java
Papyrus
*Point.java
TestPoint.java

```
package oop;

public class Point {

    private int x;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

}
```

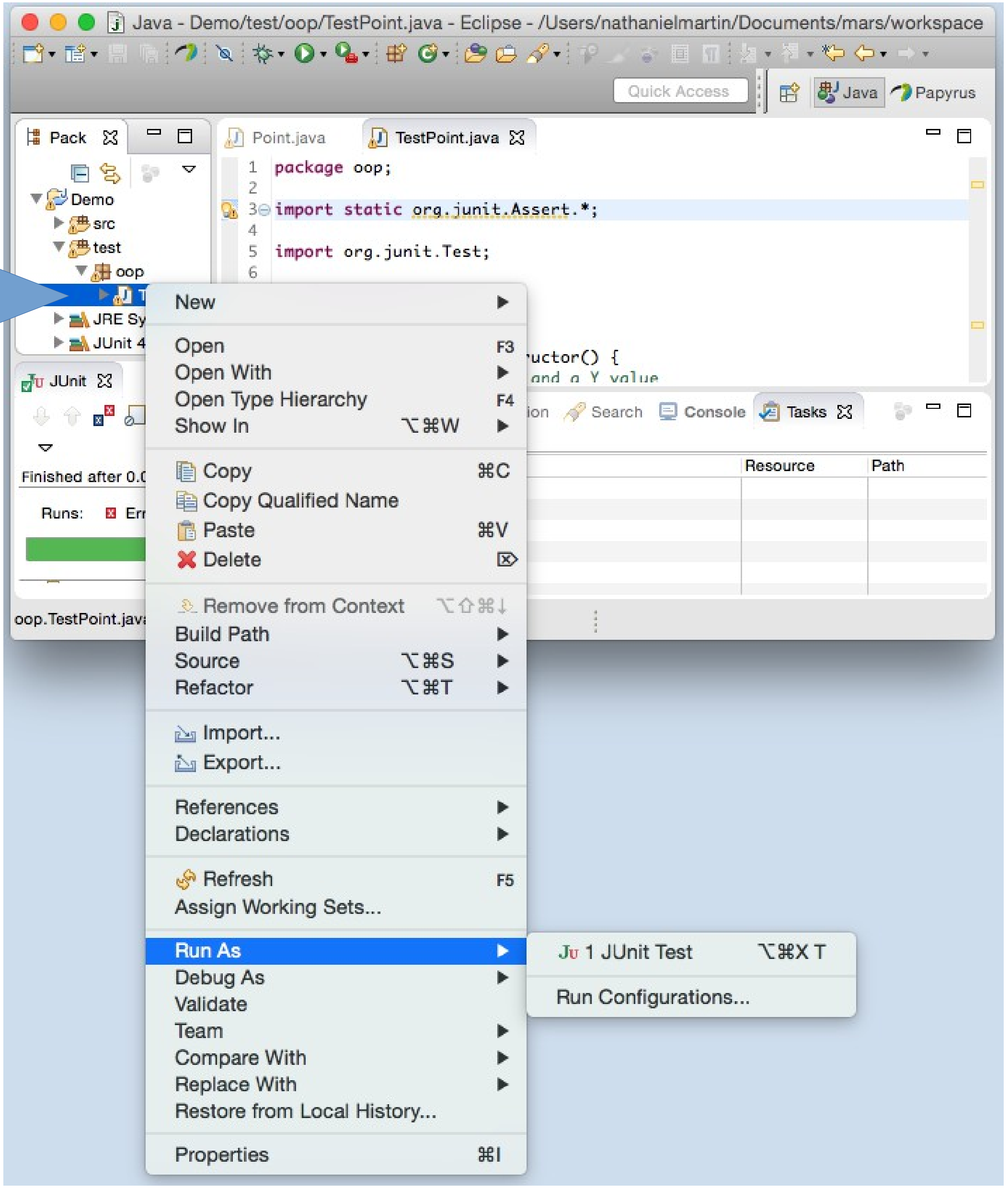
Create field 'y' in type 'Point'
Change to 'x'
Rename in file (⌘2 R)

```
...
private int x;
private int y;
...
```

Press 'Tab' from proposal table or click for focus
Writable
Smart Insert
9 : 15

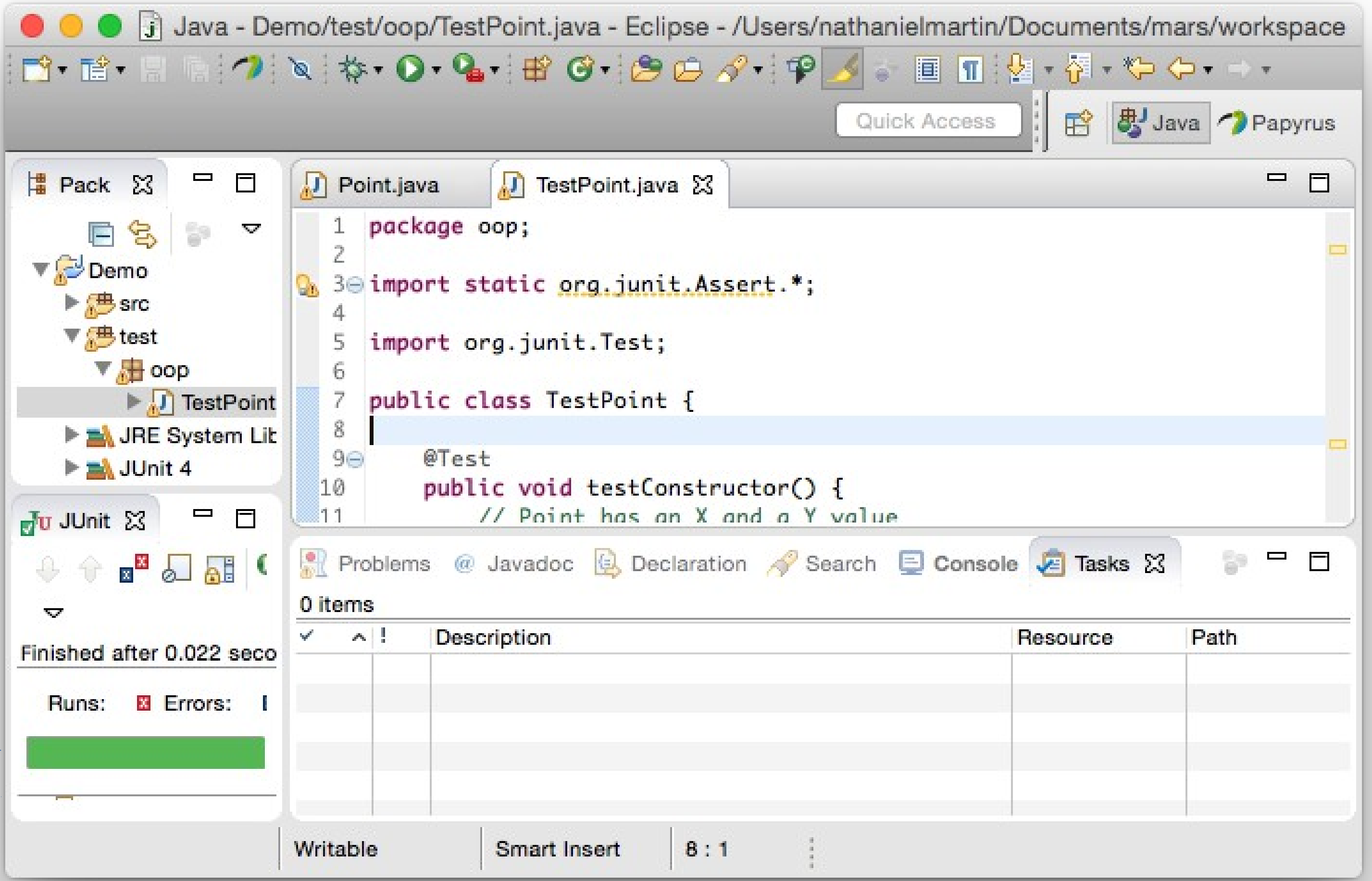
Run JUnit Test

- Double click “TestPoint”
- Select “Run As”
- Choose “JUnit Test”



```
package oop;

import static org.junit.Assert.*;

import org.junit.Test;

uctor() {
and a Y value
```

Oops! Test Passes


Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
package oop;

import static org.junit.Assert.*;

import org.junit.Test;

public class TestPoint {

    @Test
    public void testConstructor() {
        // Point has an X and a Y value
```

Finished after 0.022 seco

Runs: Errors:

0 items

		Description	Resource	Path

Writable | Smart Insert | 8 : 1

Green when test passes

Need a failing test

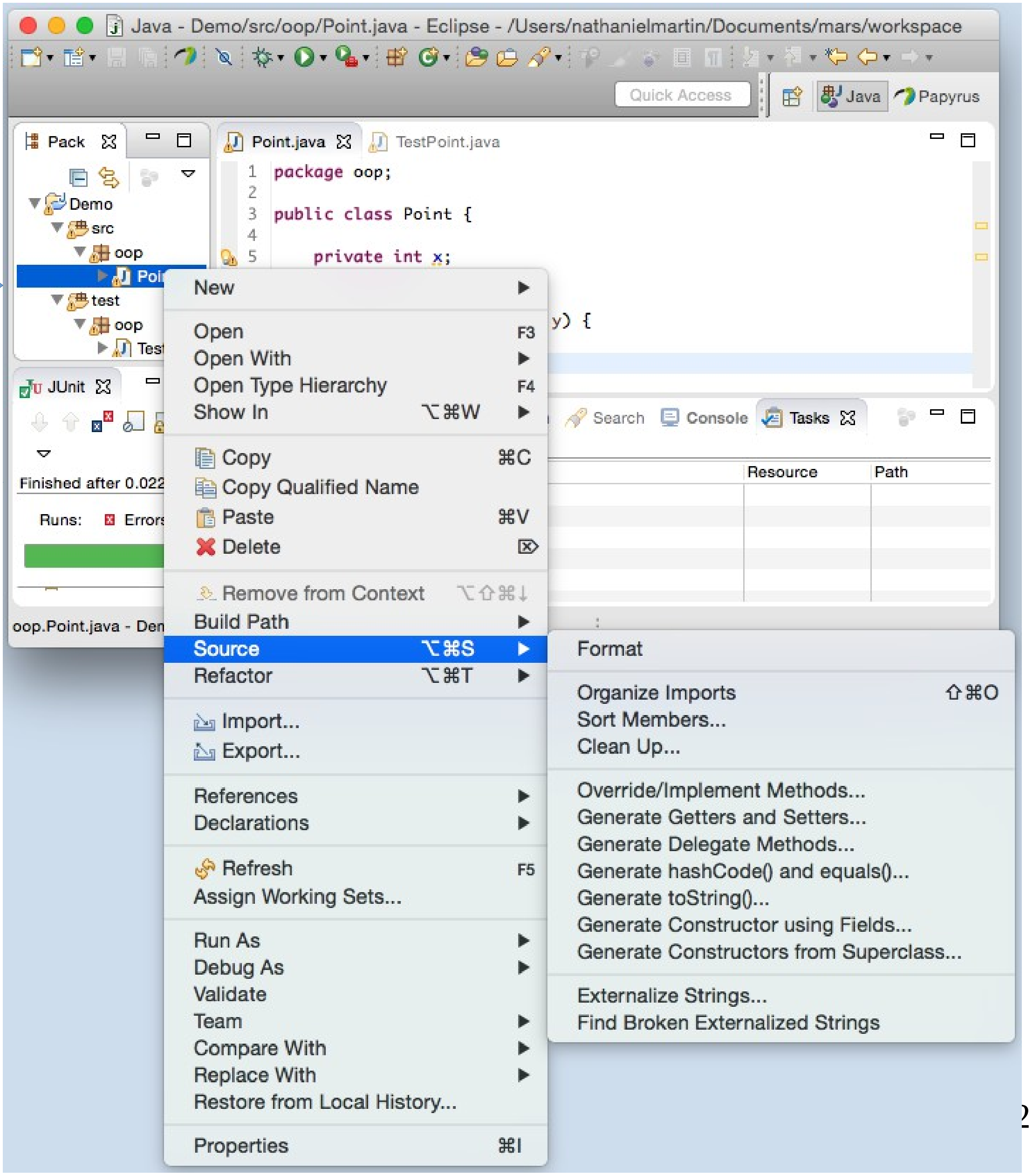
- To add new code, we need a failing test.
- Making sure the test fails before we add code ensures that the test is working
  - If it succeeds before we add code, no code needs to be added
- Here we can test that the instance variables are set correctly.

Checking Instance Variable

- Instance variables are always private
  - They are set by setter methods
  - They are read by getter methods
- Instance variables are private to keep other classes from manipulating the variables directly
  - It allow the class to change the variables while maintaining the interface through the method.
- Fortunately Eclipse will write them for you.

Adding getter methods

1. Right click Point
2. Choose "Source"
3. Select "Generate Getters and Setters"

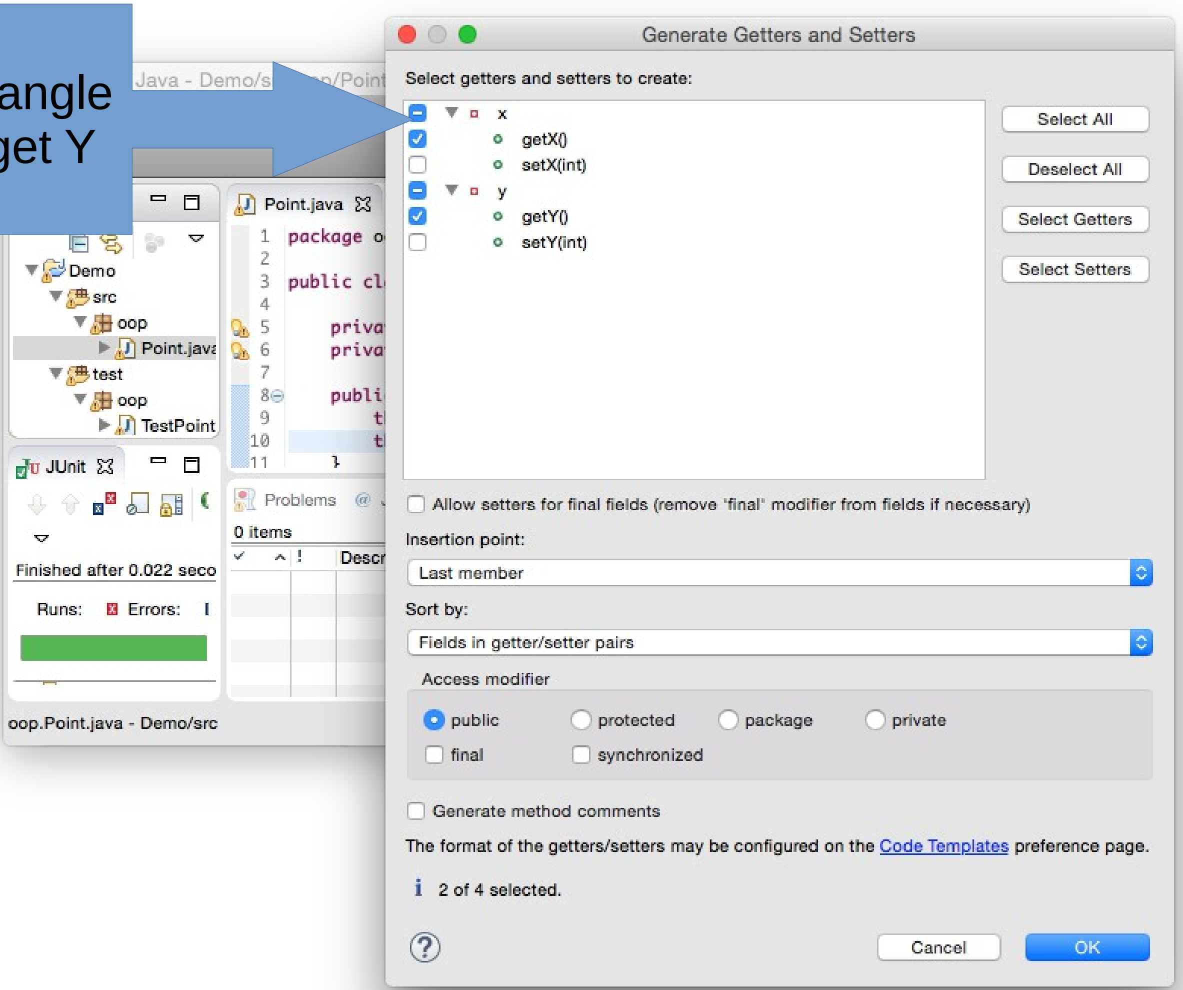


```
package oop;

public class Point {

    private int x;
```

```
y) {
```

Generate only getters

1. Click the black triangle
2. Select getX and get Y



Select getters and setters to create:

- [–] x
  - [✓] getX()
  - [ ] setX(int)
- [–] y
  - [✓] getY()
  - [ ] setY(int)

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: Last member

Sort by: Fields in getter/setter pairs

Access modifier

[✓ public] protected package private

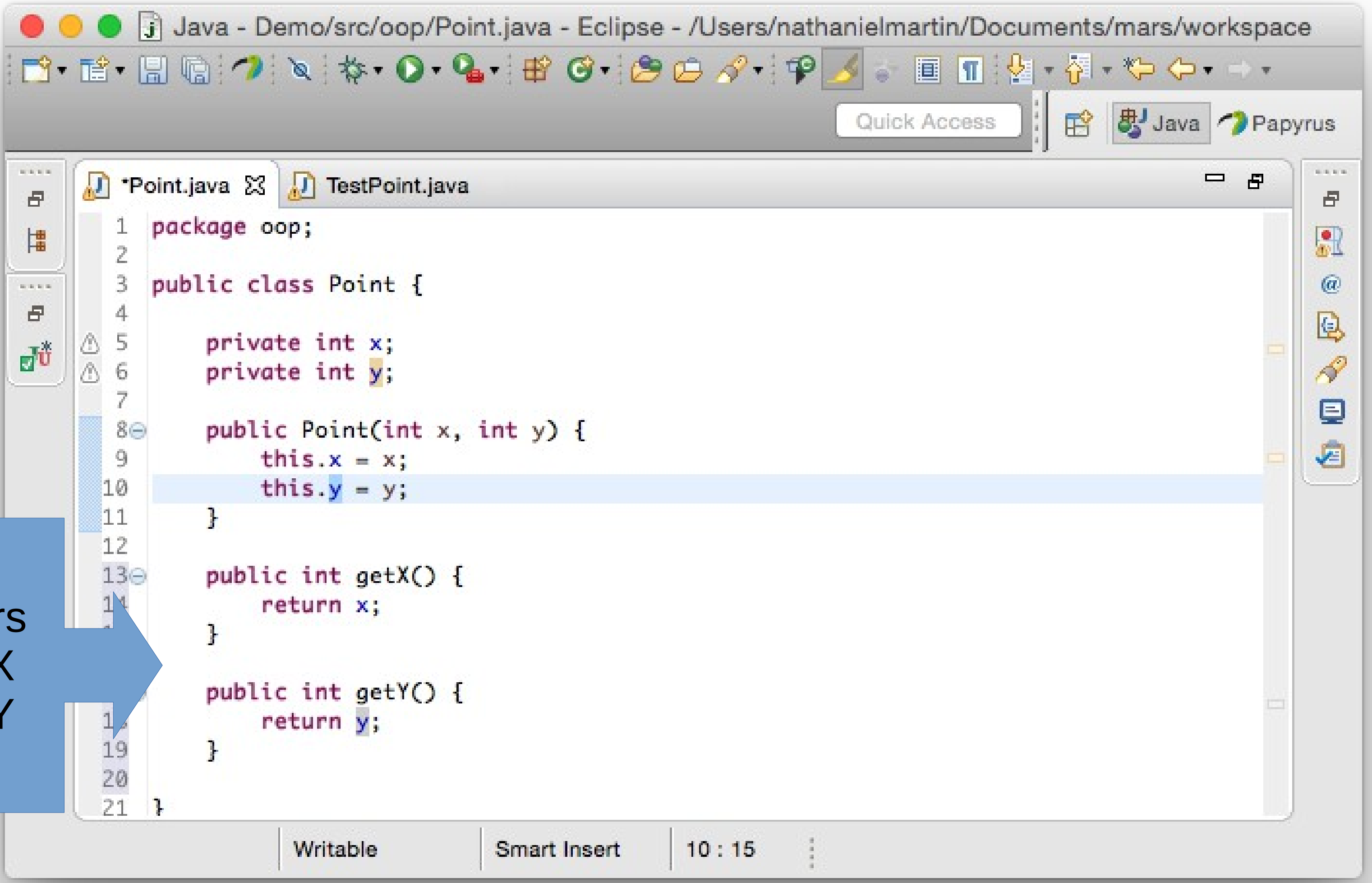
[ ] final [ ] synchronized

[ ] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

2 of 4 selected.

Getters are added to Class



Java - Demo/src/oop/Point.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

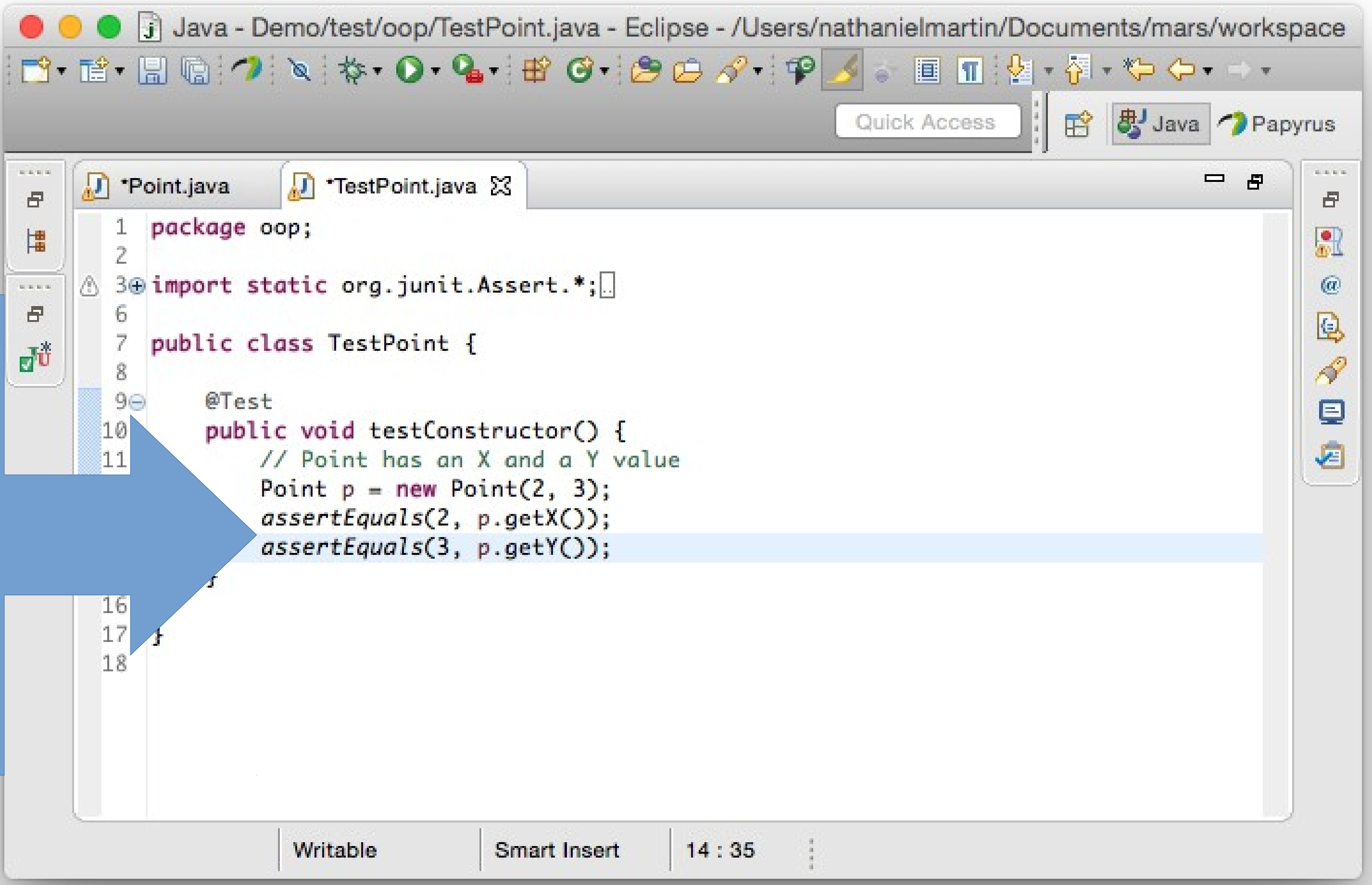
*Point.java TestPoint.java

```java
package oop;

public class Point {

    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

}
```

Writable   Smart Insert   10 : 15

Getters
- getX
- getY

Testing Variable Values



Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

*Point.java | *TestPoint.java

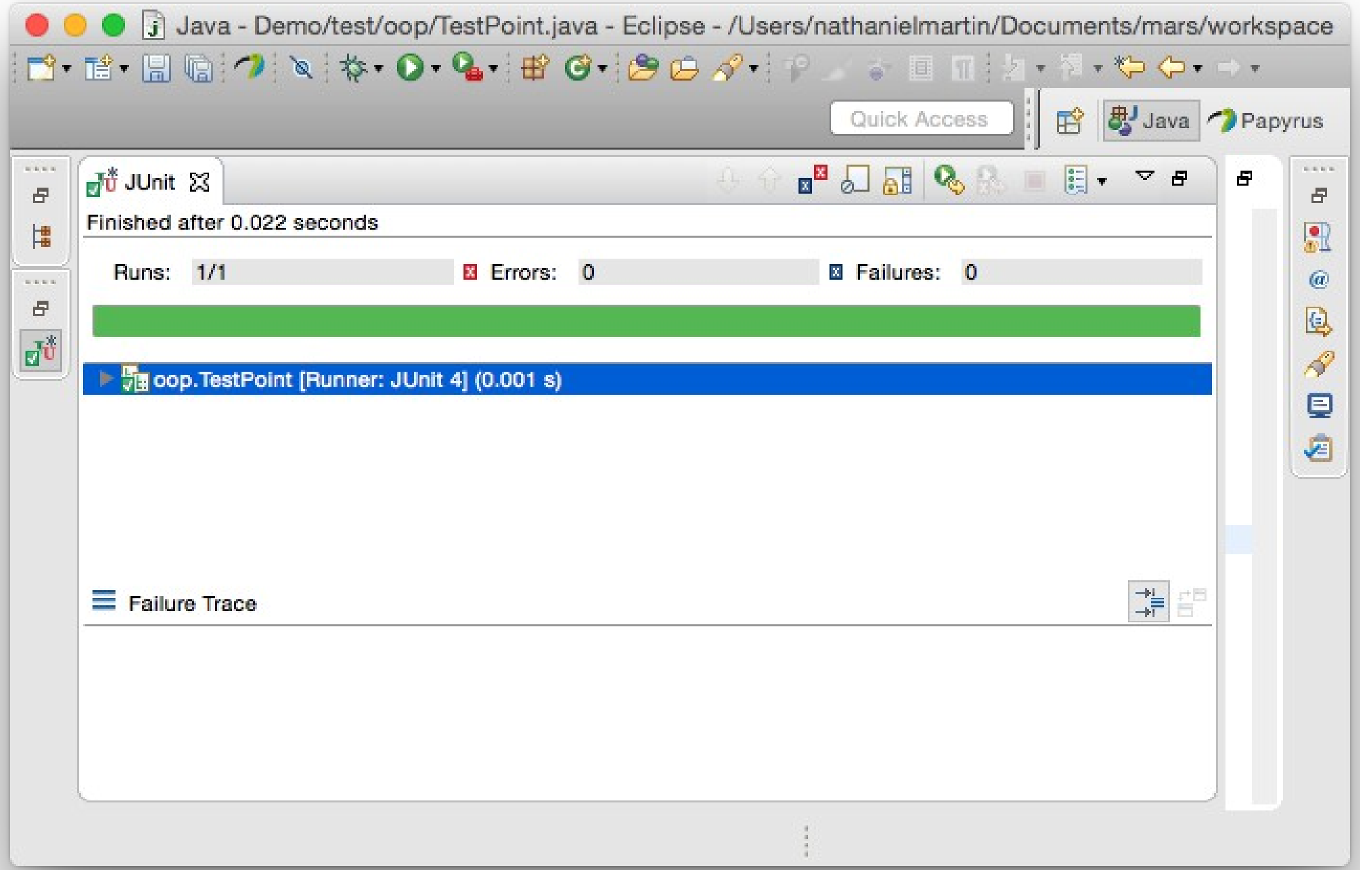
```java
package oop;

import static org.junit.Assert.*;

public class TestPoint {

    @Test
    public void testConstructor() {
        // Point has an X and a Y value
        Point p = new Point(2, 3);
        assertEquals(2, p.getX());
        assertEquals(3, p.getY());
    }

}
```

Writable | Smart Insert | 14 : 35

<obj>.<method>()
p.getX()

AssertEquals
- Succeeds when
  Parameters equal
- Parameters
  - Expected
  - Actual

Oops, Still Passes



Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
Finished after 0.022 seconds

Runs: 1/1    Errors: 0    Failures: 0

oop.TestPoint [Runner: JUnit 4] (0.001 s)
```

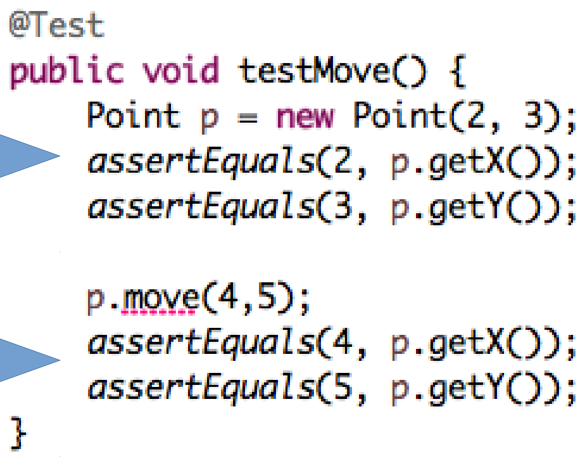
Lets try another test

- Define a function that will move a point to a new location
- It will take two parameters defining the new location of the point

Add the test first

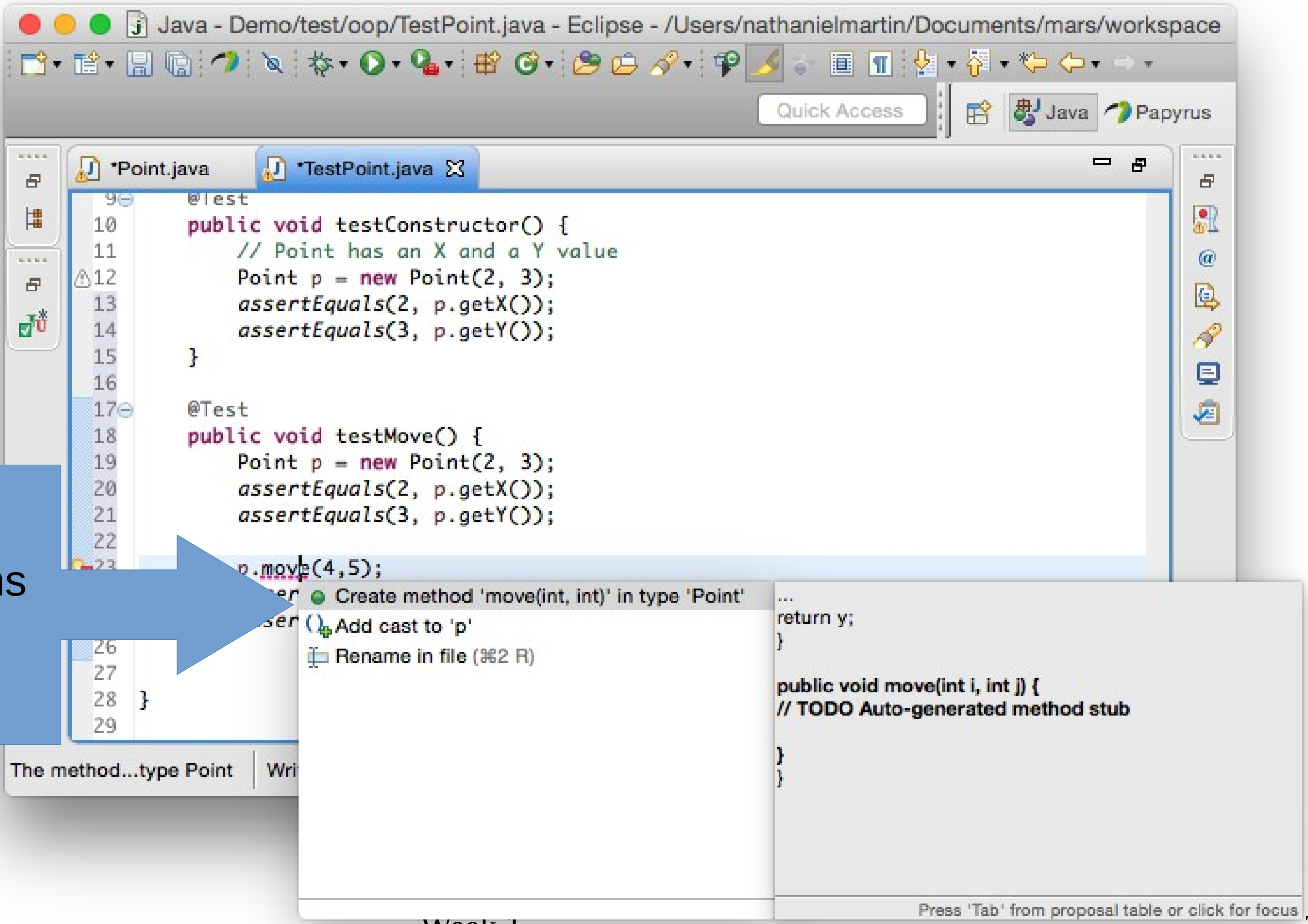
Create a new Point
Check values

Move the Point
Check values



```
@Test
public void testMove() {
    Point p = new Point(2, 3);
    assertEquals(2, p.getX());
    assertEquals(3, p.getY());

    p.move(4,5);
    assertEquals(4, p.getX());
    assertEquals(5, p.getY());
}
```

Fix the compiler error


Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
    @Test
    public void testConstructor() {
        // Point has an X and a Y value
        Point p = new Point(2, 3);
        assertEquals(2, p.getX());
        assertEquals(3, p.getY());
    }

    @Test
    public void testMove() {
        Point p = new Point(2, 3);
        assertEquals(2, p.getX());
        assertEquals(3, p.getY());

        p.move(4,5);
```

```
...
return y;
}

public void move(int i, int j) {
// TODO Auto-generated method stub

}
}
```

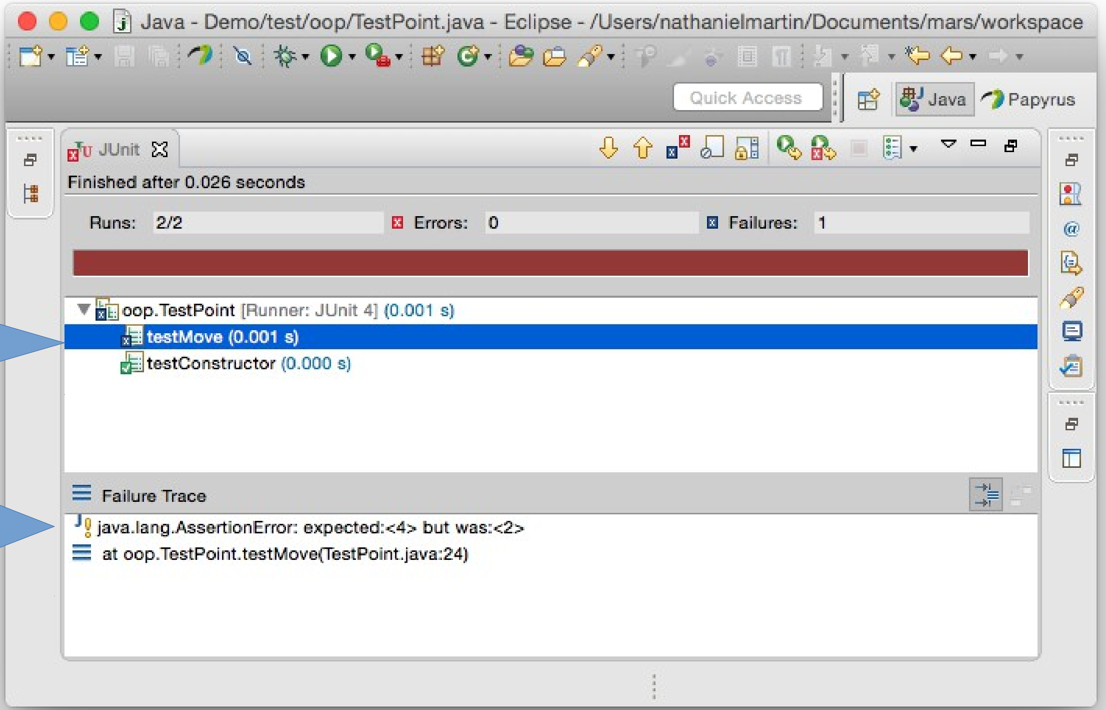
Ctrl-1 for suggestions
Add move(int, int)

Run the test; It fails

```
@Test
public void testMove() {
    Point p = new Point(2, 3);
    assertEquals(2, p.getX());
    assertEquals(3, p.getY());

    p.move(4,5);
    assertEquals(4, p.getX());
    assertEquals(5, p.getY());
}
```

testMove fails

Expected <4>
But was <2>


Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace
Quick Access
Java
Papyrus
JUnit

Finished after 0.026 seconds

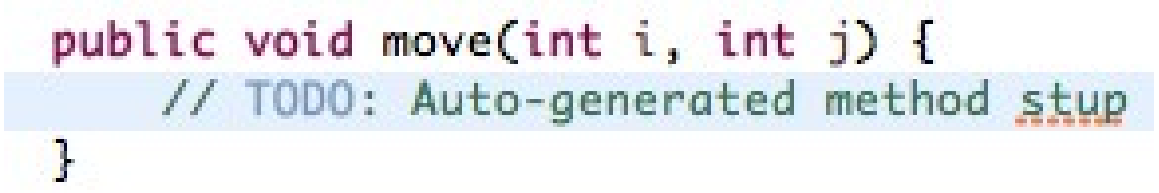
Runs: 2/2 Errors: 0 Failures: 1

```
oop.TestPoint [Runner: JUnit 4] (0.001 s)
    testMove (0.001 s)
    testConstructor (0.000 s)
```

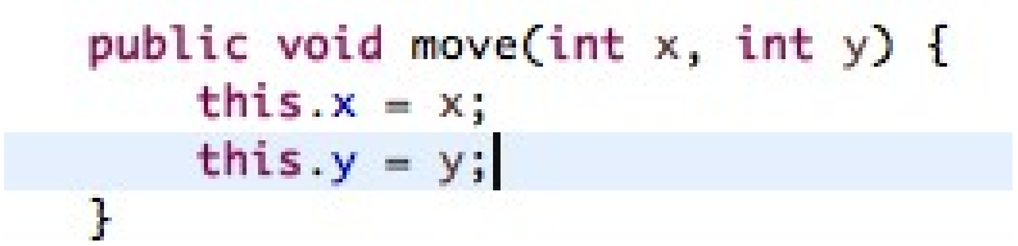
Failure Trace

```
java.lang.AssertionError: expected:<4> but was:<2>
at oop.TestPoint.testMove(TestPoint.java:24)
```

Fix the code

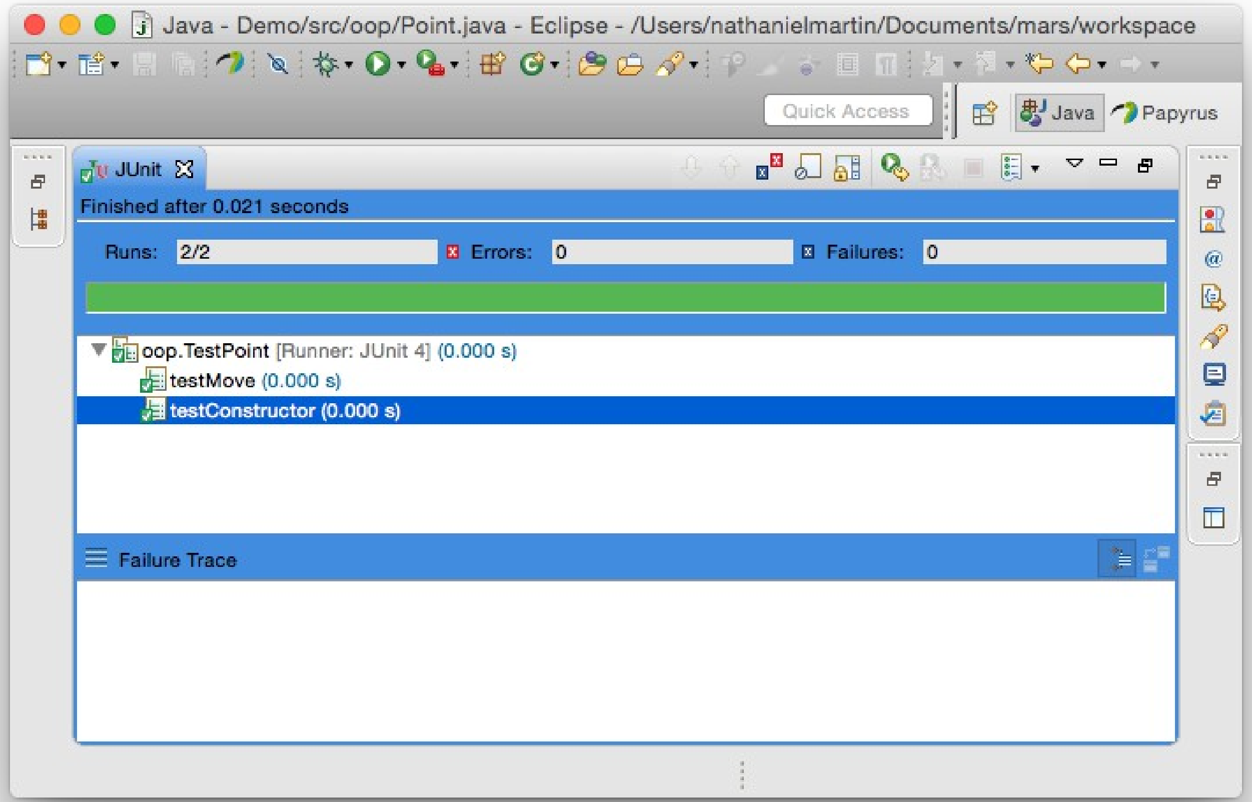


```
public void move(int i, int j) {
    // TODO: Auto-generated method stup
}
```



```
public void move(int x, int y) {
    this.x = x;
    this.y = y;
}
```

Now it works



Java - Demo/src/oop/Point.java - Eclipse - /Users/nathanielmartin/Documents/mars/workspace

```
Finished after 0.021 seconds

Runs: 2/2     Errors: 0     Failures: 0

oop.TestPoint [Runner: JUnit 4] (0.000 s)
    testMove (0.000 s)
    testConstructor (0.000 s)
```

Recap

- We created a Java project
- We added a Class to the project (Point)
- We added a test folder to the project
- We added a Test Case to the test folder (TestPoint)
- We added a test of the Constructor to the test folder.
- We build up the Constructor by correcting compiler errors
- We build a move method by correcting a failing test

Using a Java Object

Using an Object

- Make the object available by importing it's package.
  - Objects are defined in packages to avoid name collision.
  - Our Point is different from other points.
- Create the object using the constructor function
- Call the object's methods by giving the object and the method.

Using Point

- We have created a class called Point.
  - It has an x and y position
- To use the point we can
  - Create a new Point object, which sets x and y
    - E.g., `Point p = new Point(2, 3);`
  - Retrieve x and y
    - E.g., `int x = p.getX();`
  - Move the Point by changing x and y
    - E.g., `p.move(5, 6);`

Running a Class

- A Class is not a program, it is a way of creating kinds of objects
- You can turn a Class into a program by adding a main() function.
  - As in C, the main() function is the starting place for the program.

Testing a Program

- When building a class, we use *unit testing*
  - Unit testing tests the class
  - Unit testing checks the class for the programmer
- When testing a program, we use *system testing*
  - System testing tests the entire program
  - System testing is also called end-to-end testing
  - System testing checks that the program does what the user wants

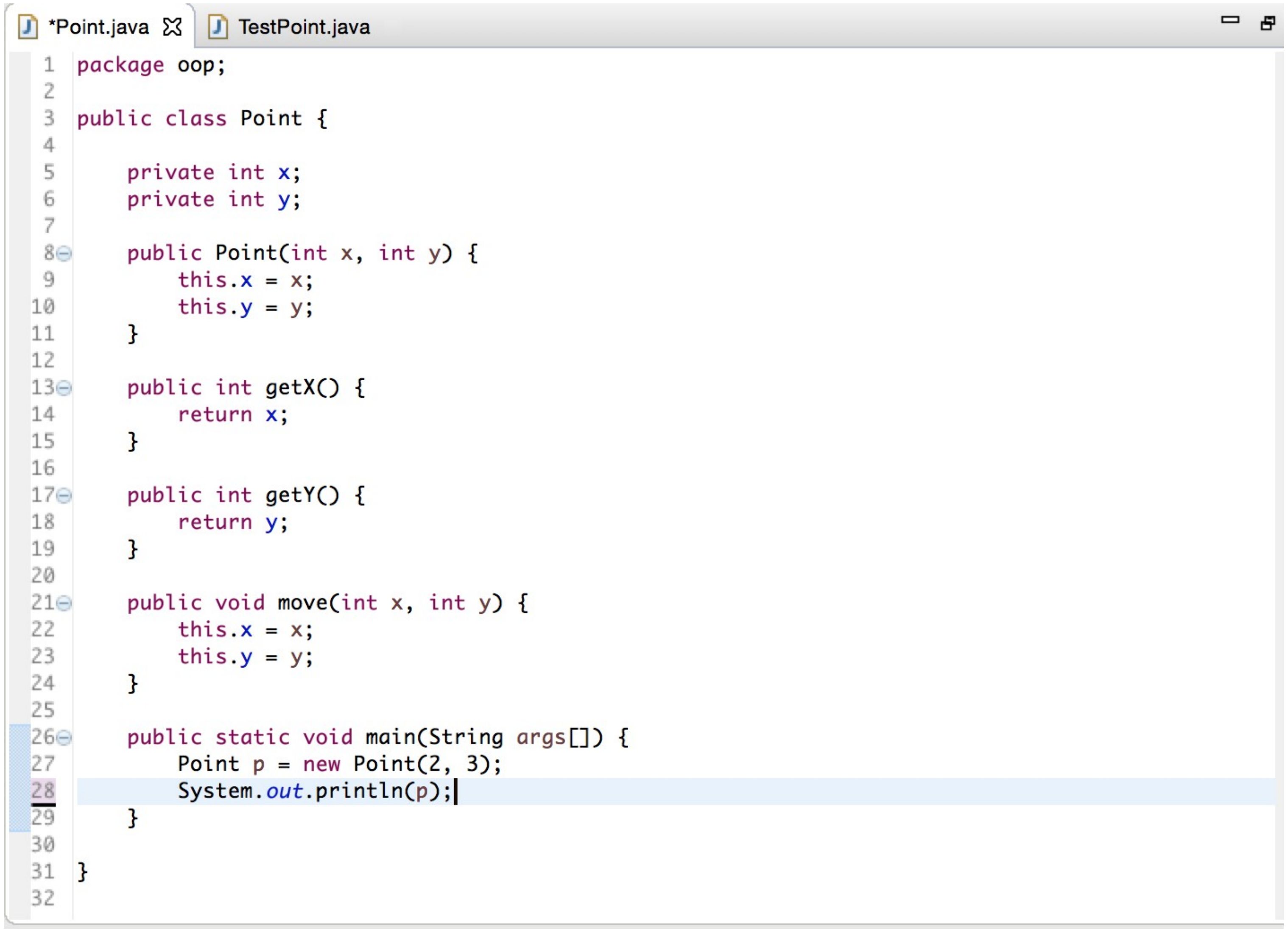
System Testing

- In system testing, we need input and output.
- We will create a point then print it out.

Output

- In java, we can print using System.out.println()
  - System is the name of a class that is included by default
  - The instance variable “out” is a in System.
  - The method println() is a method defined on out, which takes a single string as a parameter.

Point gets main() method

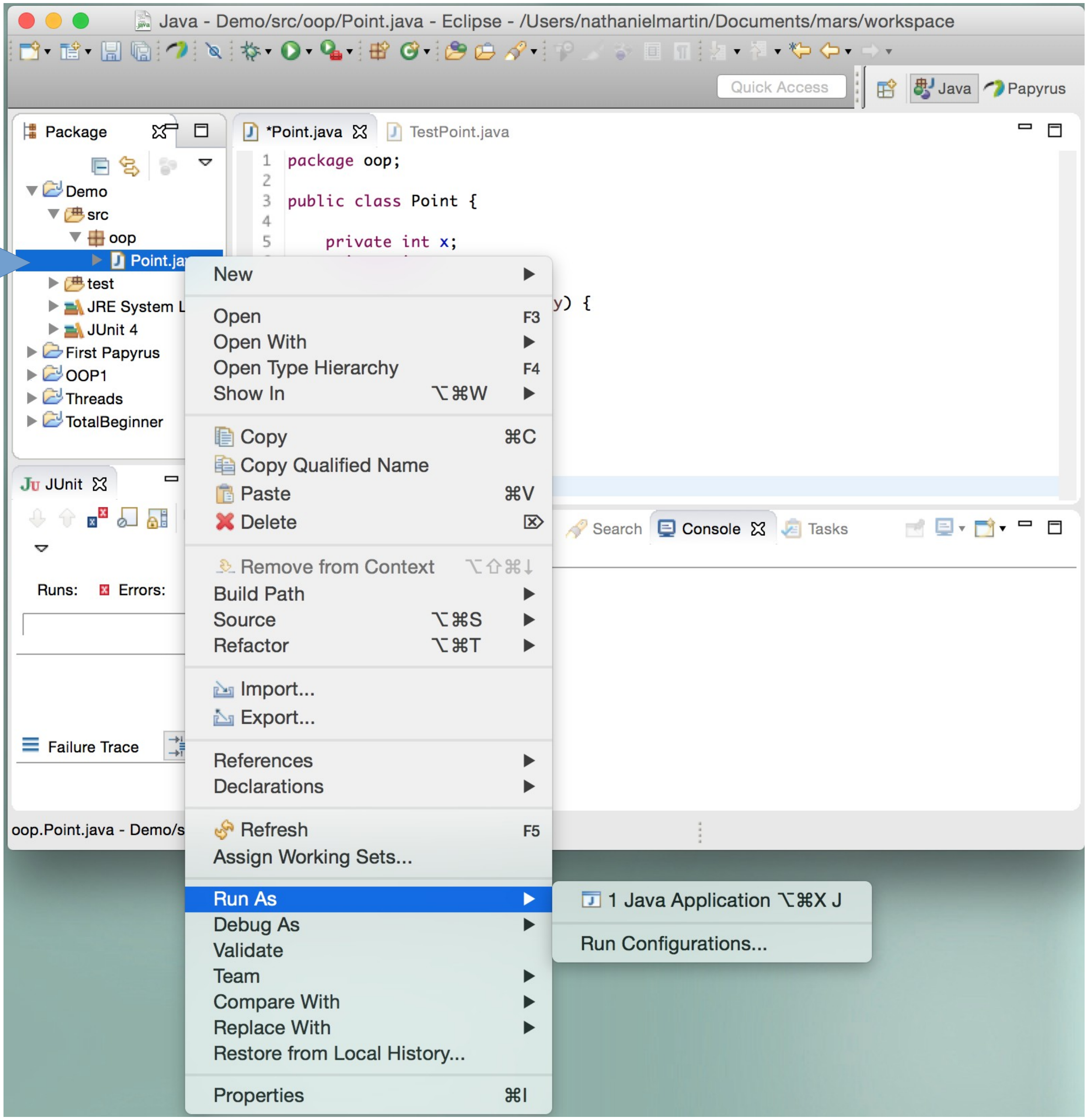


*Point.java TestPoint.java

```java
package oop;

public class Point {

    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public static void main(String args[]) {
        Point p = new Point(2, 3);
        System.out.println(p);
    }

}
```

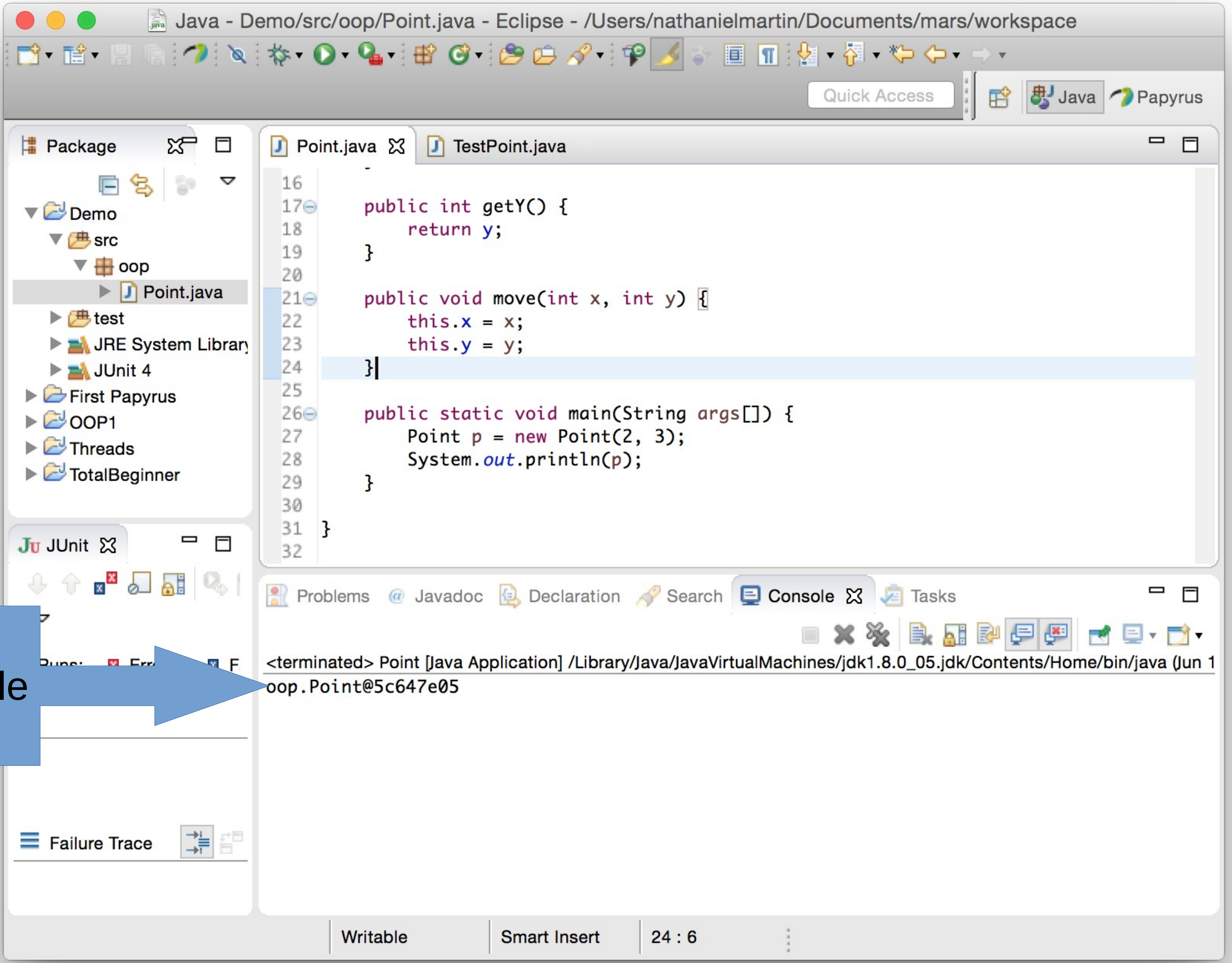
Run the program

1) Double Click on class
2) Select "Run As"
3) Choose Java Application



```
package oop;

public class Point {

    private int x;
```

Results

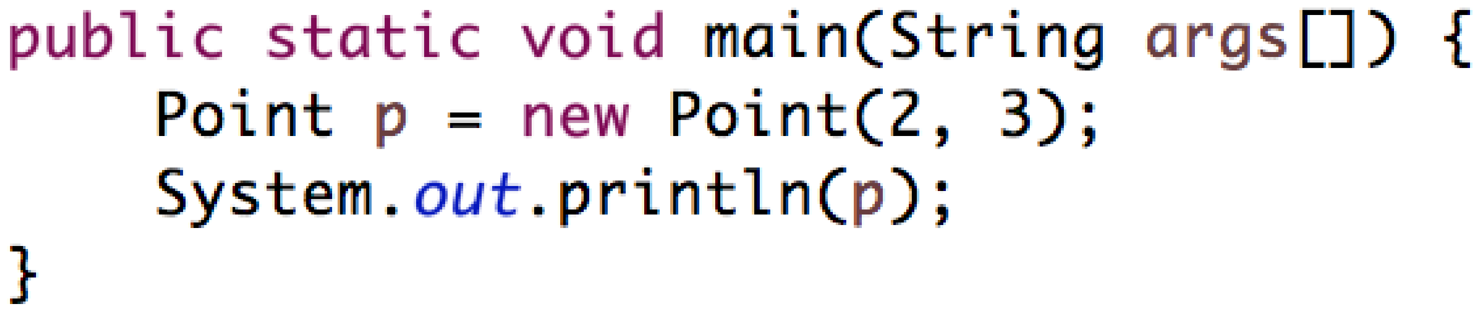


```java
    public int getY() {
        return y;
    }

    public void move(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public static void main(String args[]) {
        Point p = new Point(2, 3);
        System.out.println(p);
    }

}
```

```
<terminated> Point [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/bin/java (Jun 1
oop.Point@5c647e05
```

Output shows in console

System.out.println()



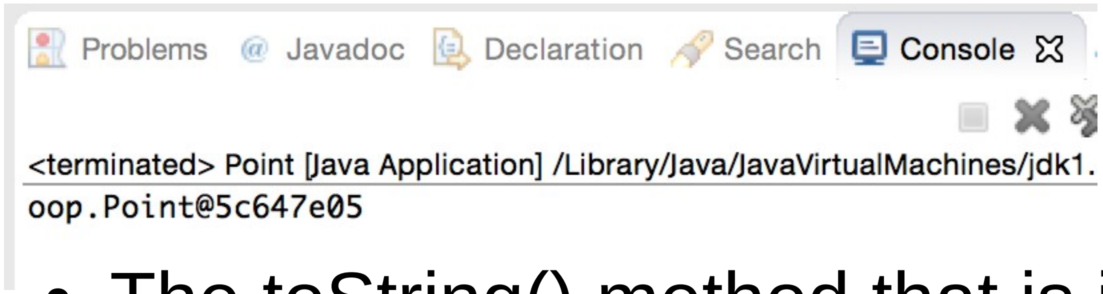
```
public static void main(String args[]) {
    Point p = new Point(2, 3);
    System.out.println(p);
}
```

- The method System.out.println() prints only strings
  - System.out.println(p) changes p from a Point to a string using the toString method.
  - When you use a Class where a string is needed, Java implicitly changes it to a string using toString()

toSting()

- Every object has a toString() method defined for it.
  - The method is *inherited* from the Object class
  - The Object class is the basis of all classes in Java
  - If we add a method with the same name and parameters as an inherited method, we *override* the inherited method
    - That is we redefine the inherited method for our class

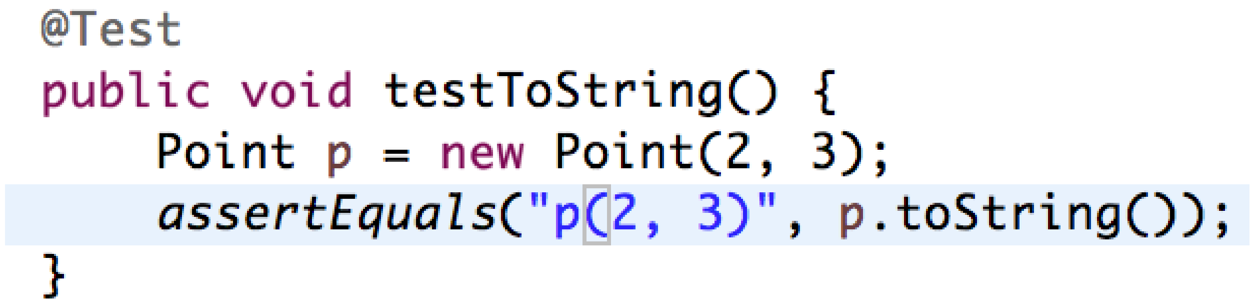
ToString output



```
<terminated> Point [Java Application] /Library/Java/JavaVirtualMachines/jdk1.
oop.Point@5c647e05
```

- The toString() method that is inherited can only provide very generic information
  - oop.Point@5c647e05
  - Prints out the package, class and location
    - Not particularly informative; but always available to Java
- We can override toString() to be more useful
  - We add the toString() method to the class using TDD

Add a test



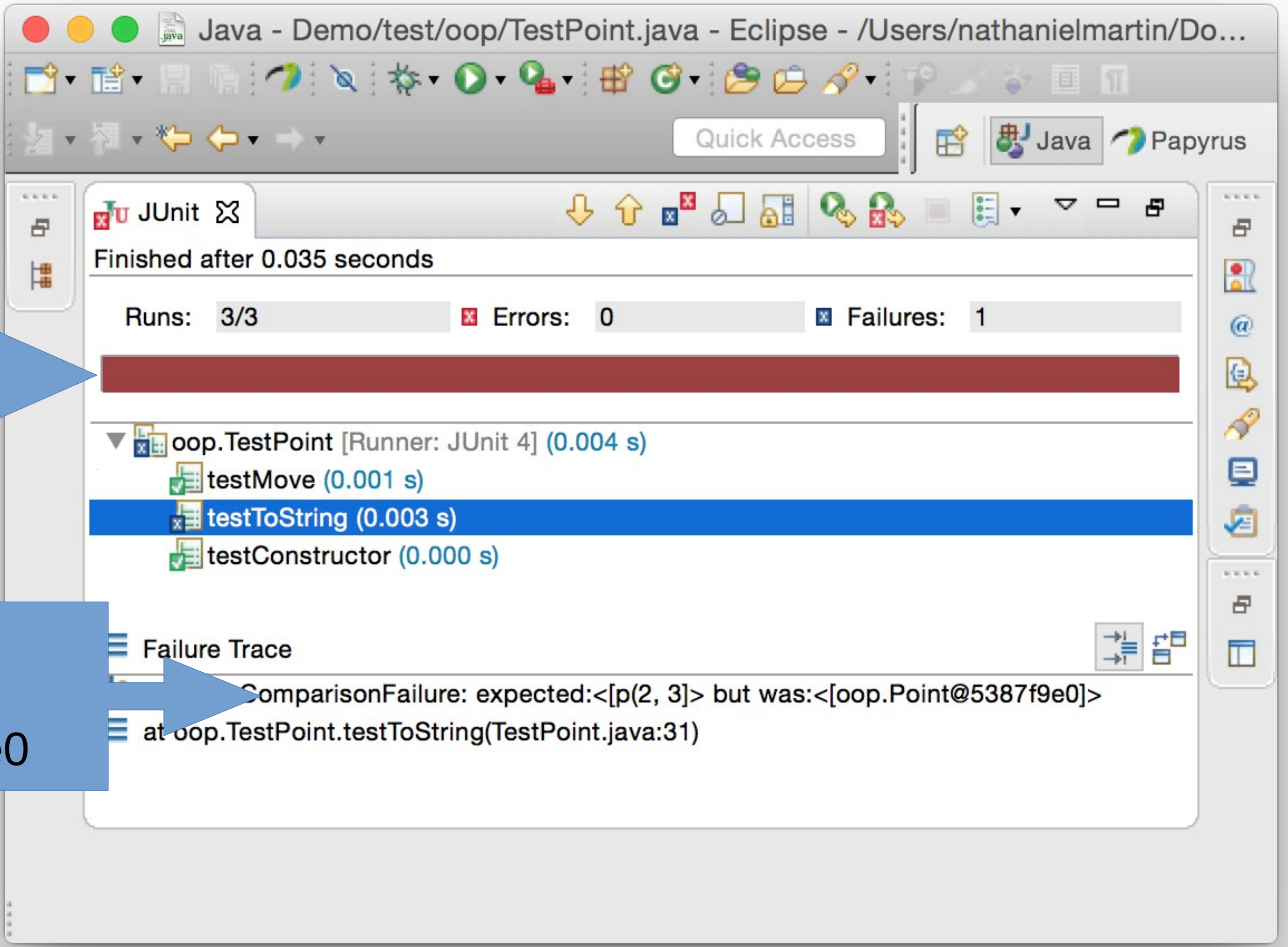
```
@Test
public void testToString() {
    Point p = new Point(2, 3);
    assertEquals("p(2, 3)", p.toString());
}
```

- Create a point and check that toString makes the right thing.
- There are no compiler errors because it is calling the inherited method.
- The test documents what it should produce
  - i.e., "p(2, 3)" for a point at position 2, 3.

Run the Test (Fails: Yeah!)

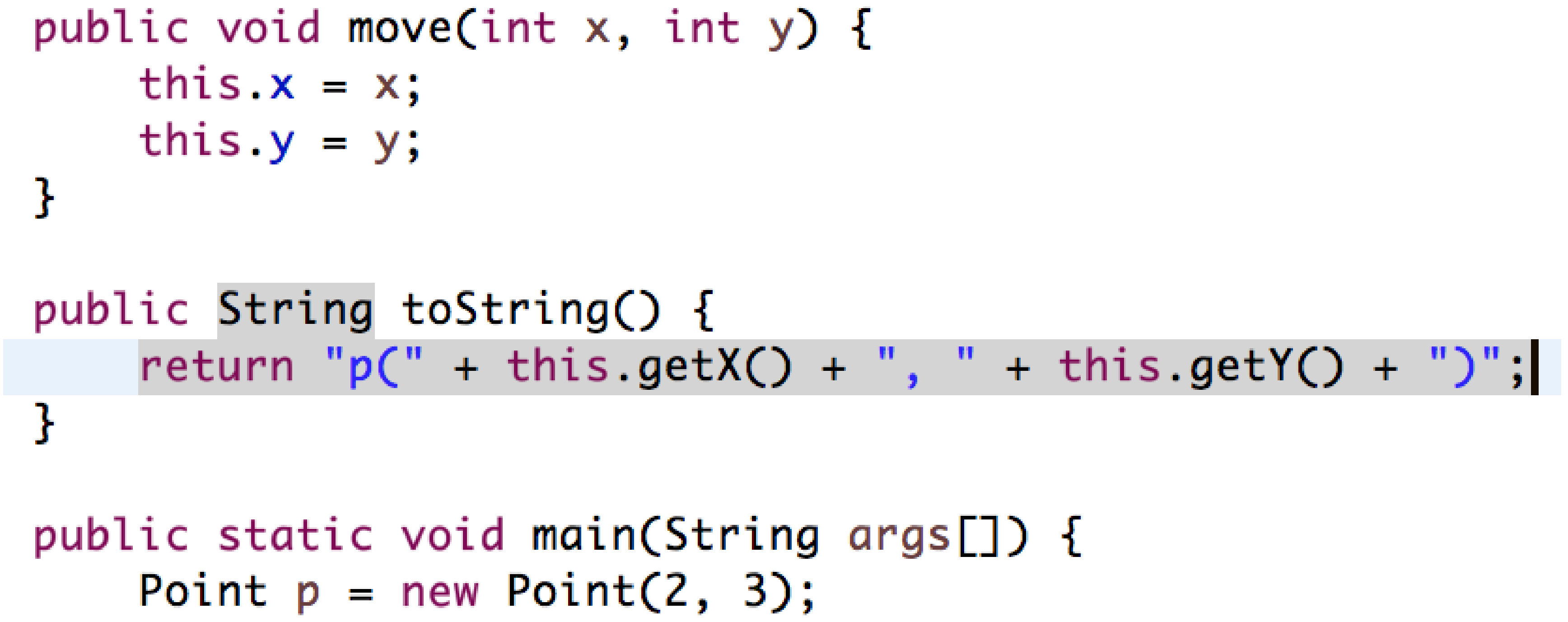
Test Fails

Comparison Failure
Expected p(2, 3)
Was oop.Point@53879e0


Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Do...
Quick Access
Java
Papyrus
JUnit

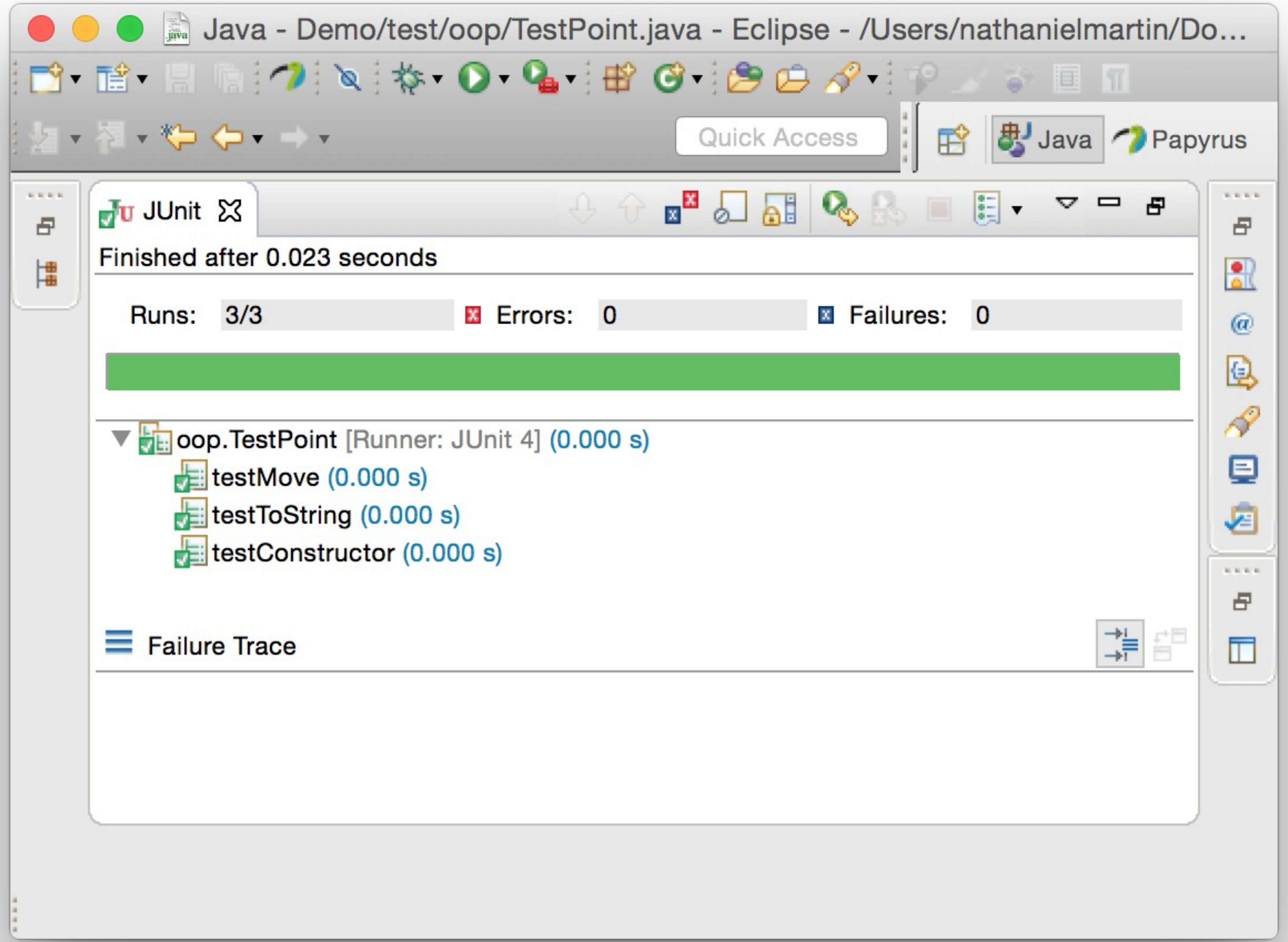
```
Finished after 0.035 seconds

Runs: 3/3    Errors: 0    Failures: 1

oop.TestPoint [Runner: JUnit 4] (0.004 s)
    testMove (0.001 s)
    testToString (0.003 s)
    testConstructor (0.000 s)

Failure Trace
ComparisonFailure: expected:<[p(2, 3]> but was:<[oop.Point@5387f9e0]>
at oop.TestPoint.testToString(TestPoint.java:31)
```

Add the new method



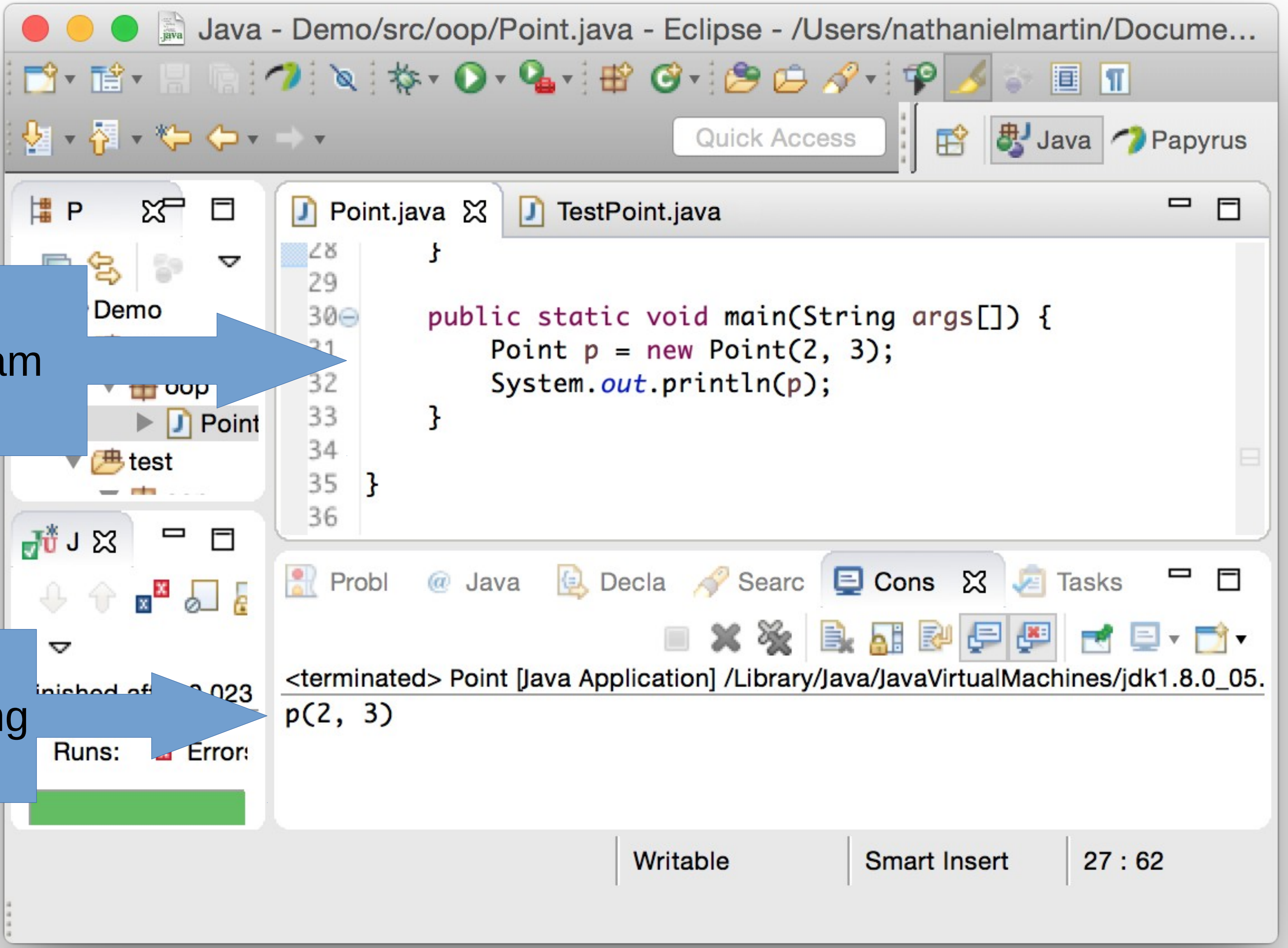
```
public void move(int x, int y) {
    this.x = x;
    this.y = y;
}

public String toString() {
    return "p(" + this.getX() + ", " + this.getY() + ")";
}

public static void main(String args[]) {
    Point p = new Point(2, 3);
```

Run test again (Succeeds!)


Java - Demo/test/oop/TestPoint.java - Eclipse - /Users/nathanielmartin/Do…
Quick Access
Java
Papyrus
JUnit
Failure Trace

```
Finished after 0.023 seconds
Runs: 3/3    Errors: 0    Failures: 0

oop.TestPoint [Runner: JUnit 4] (0.000 s)
    testMove (0.000 s)
    testToString (0.000 s)
    testConstructor (0.000 s)
```

Try running to program again



```
        }

        public static void main(String args[]) {
            Point p = new Point(2, 3);
            System.out.println(p);
        }

    }
```

```
<terminated> Point [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_05.
p(2, 3)
```

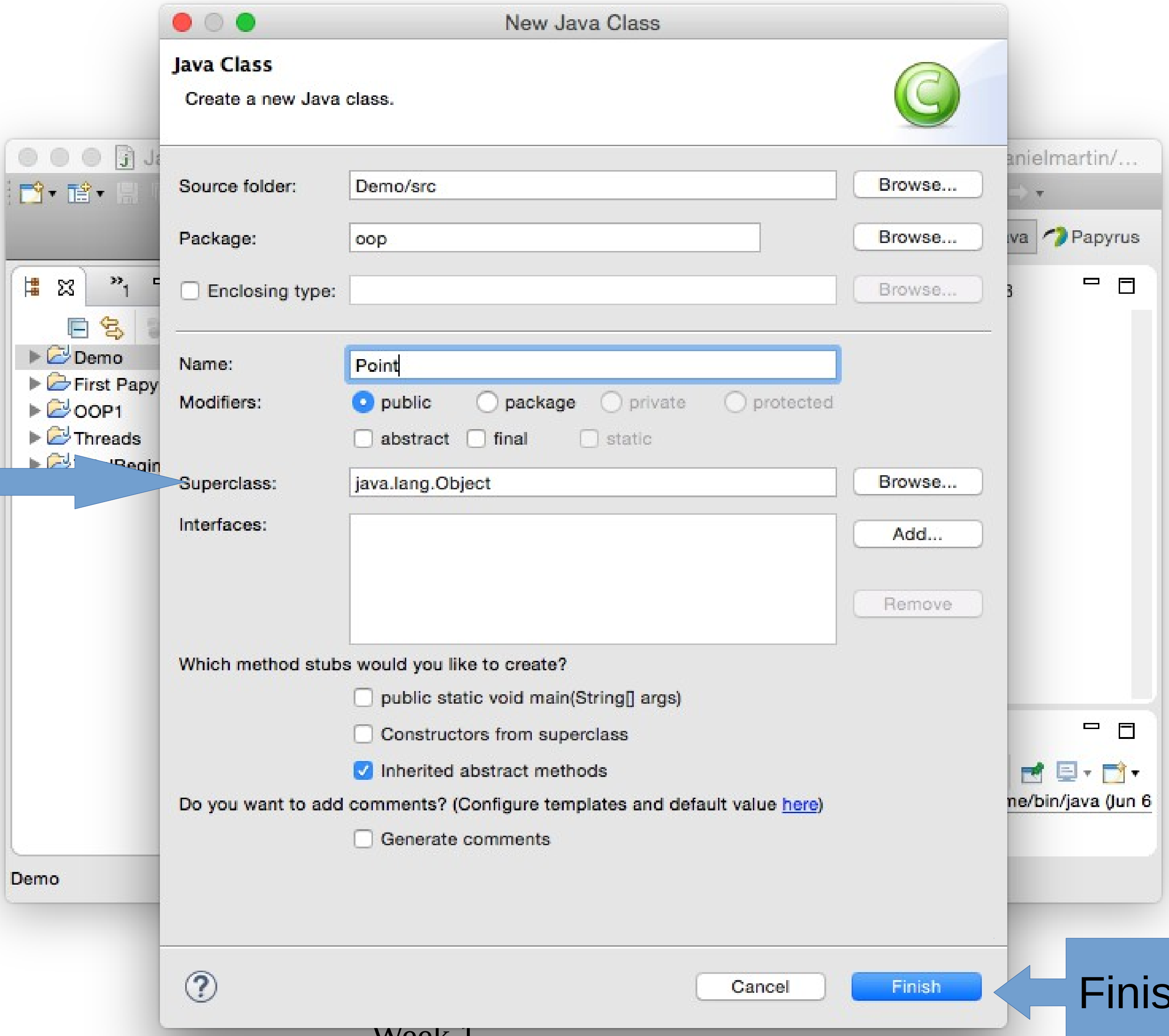
No change to program

New version of toString

Inheritance

- In Java classes inherit all of the methods and instance variables of their super class
- Any class can be a super class
  - For example, we could have a RedPoint that inherits from Point, and is different only in the way it prints.
- We specify inheritance when we define a class

Specify Inheritance when Creating Class



New Java Class

Java Class

Create a new Java class.

Source folder:	Demo/src	Browse...
Package:	oop	Browse...
☐ Enclosing type:		Browse...
Name:	Point	
Modifiers:	[✓ public] package private protected	
	☐ abstract ☐ final ☐ static	
Superclass:	java.lang.Object	Browse...
Interfaces:		Add... Remove

Which method stubs would you like to create?

- ☐ public static void main(String[] args)
- ☐ Constructors from superclass
- ☑ Inherited abstract methods

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

- ☐ Generate comments

Inherits from Java.lang.Object

Finish

Review

- In this lecture we have covered:
  - TDD
    - To add instance variables
    - To add a Constructor
    - To add a method
  - Writing a main() function
  - Redefining inherited function toString