# Practical Unit Testing jUnit

Atul Prakash

## Unit Testing

- Software must be reliable for it to be useful
- You would not like a microwave oven or TV that keeps misbehaving. It could be even dangerous
- Same thing with software. Today, software controls your car, iPod, cellphone, your records at the university, etc. It is everywhere even though you don't see it.

## Role of Testing in Development

- Typically, companies spend 50-60% of their software budgets on testing. More than on development.
- If bugs remain in the code after release, it is usually expensive to fix them.
  - Bad publicity; releasing patches on a time cycle that is not under your control; legal liability; fixes may break other things; etc.

## Unit Testing

- As you write each module (e.g., class), write the corresponding tests.
- If you find a bug in your code that is not uncovered by testing, add a test case as well that exhibits the bug first. That way, you will prevent the bug from creeping back again.
- Software changes all the time. Testing helps ensure that changes don't break old tests.

## Testing we have seen so far...

- SpaceshipGameTest: we wrote that to test some of the functions provided by the game.
- Assignment 4:
  - SquareShipTest: test a square-shaped ship.
  - BirdOfPreyTest: test a ship of another shape

## Making testing convenient

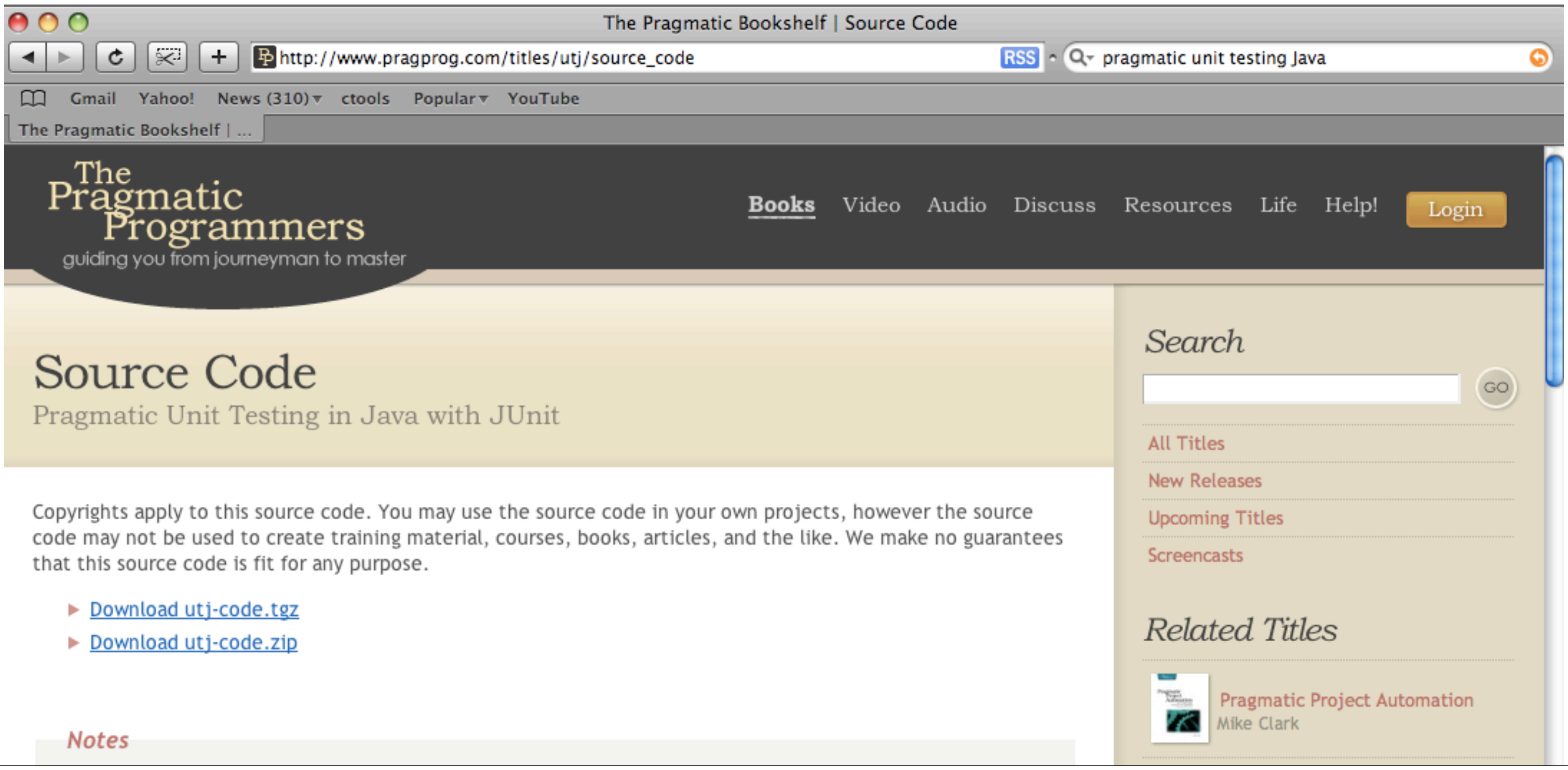
- Break your tests into small, independent tests. That way, you can say that your program passes m out of n tests. Fixes will be easier as well.
- What is it that you would like to see improved in the way we have been testing so far?

## jUnit

- jUnit is a standard unit testing framework for Java.
- pyUnit: similar framework in Python
- cppUnit: a similar framework for C++

# Sample Program and Test

- Download the zip file from Pragmatic Unit Testing book to a directory of your choice.


The Pragmatic Bookshelf | Source Code
http://www.pragprog.com/titles/utj/source_code
RSS
pragmatic unit testing Java
Gmail
Yahoo!
News (310)
ctools
Popular
YouTube
The Pragmatic Bookshelf | ...
The Pragmatic Programmers
guiding you from journeyman to master
Books
Video
Audio
Discuss
Resources
Life
Help!
Login
Search
GO
All Titles
New Releases
Upcoming Titles
Screencasts
Related Titles
Pragmatic Project Automation
Mike Clark

Source Code

Pragmatic Unit Testing in Java with JUnit

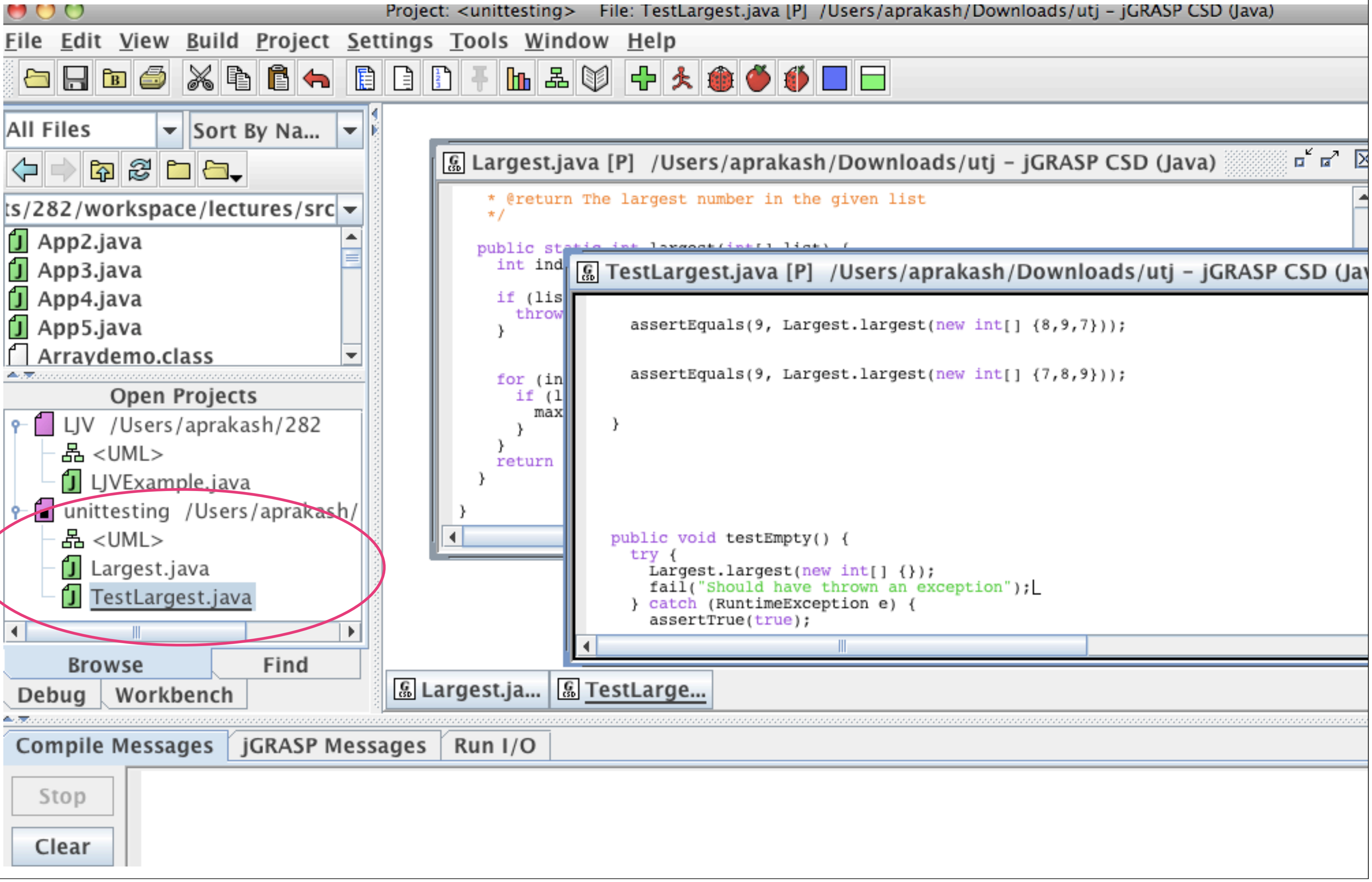
Copyrights apply to this source code. You may use the source code in your own projects, however the source code may not be used to create training material, courses, books, articles, and the like. We make no guarantees that this source code is fit for any purpose.

- Download utj-code.tgz
- Download utj-code.zip

Notes

## Add the files to jGrasp or Eclipse

- unzip the downloaded files to a directory.
- In jGrasp, go to Project -> New… and go to the directory containing the downloaded files. Create a project there called junittesting.
- Add the files "Largest.java" and "TestLargest.java" to the junittesting project.


Project: <unittesting>    File: TestLargest.java [P]  /Users/aprakash/Downloads/utj - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
All Files
Sort By Na...
ts/282/workspace/lectures/src
App2.java
App3.java
App4.java
App5.java
Arraydemo.class
Open Projects
LJV  /Users/aprakash/282
<UML>
LJVExample.java
unittesting  /Users/aprakash/
<UML>
Largest.java
TestLargest.java
Browse
Find
Debug
Workbench
Largest.java [P]  /Users/aprakash/Downloads/utj - jGRASP CSD (Java)
TestLargest.java [P]  /Users/aprakash/Downloads/utj - jGRASP CSD (Ja
Largest.ja...
TestLarge...
Compile Messages
jGRASP Messages
Run I/O
Stop
Clear

```
   * @return The largest number in the given list
   */

  public st
    int ind

    if (lis
      throw
    }


    for (in
      if (l
        max
      }
    }
    return
  }

}
```

```
    assertEquals(9, Largest.largest(new int[] {8,9,7}));


    assertEquals(9, Largest.largest(new int[] {7,8,9}));


  }




  public void testEmpty() {
    try {
      Largest.largest(new int[] {});
      fail("Should have thrown an exception");
    } catch (RuntimeException e) {
      assertTrue(true);
```

## Take a few minutes ...

- Read through the tests that are there in the TestLargest.java. Understand what they are doing.
- Largest takes an int array and finds the largest value in that array

# Test structure

```java
import junit.framework.*;

public class TestLargest extends TestCase {

  public TestLargest(String name) {
    super(name);
  }

  public void testOrder() {

    assertEquals(9, Largest.largest(new int[] {9,8,7}));

    assertEquals(9, Largest.largest(new int[] {8,9,7}));


    assertEquals(9, Largest.largest(new int[] {7,8,9}));


  }

  public void testEmpty() {
    try {
      Largest.largest(new int[] {});
      fail("Should have thrown an exception");
    } catch (RuntimeException e) {
      assertTrue(true);
    }
  }


  public void testOrder2() {
    int[] arr = new int[3];
    arr[0] = 8;
    arr[1] = 9;
    arr[2] = 7;
    assertEquals(9, Largest.largest(arr));
  }

}
```

## Junit Tests

- Class inherits from TestCase
- Constructor needed as given.
- It provides a few new methods:
  - assertEquals(value, expression)
  - assertTrue(expr)
  - assertFalse(expr)
  - fail(msg)
- Each test run till it either completes or an assertion fails

## Assertions

- Assertions are simply a condition that you expect to be true at a given point in the program, based on its specs:
- We saw:
  - assert expression
- If the expression is false, the program quits.

# jUnit Assertions

- jUnit provides a few additional ways of expressing assertions.
- assertTrue(expr) is similar to
  - assert expr
- assertFalse(expr) is similar to:
  - assert !expr
- assertEquals(val, expr) is similar to:
  - assert val ==expr
- fail(msg) is similar to
  - assert false
  - But it also prints a msg.

# Example Test

```
public void testOrder2() {
  int[] arr = new int[3];
  arr[0] = 8;
  arr[1] = 9;
  arr[2] = 7;
  assertEquals(9, Largest.largest(arr));
}
```

## Equivalent test

```
public void testOrder2() {
  int[] arr = new int[3];
  arr[0] = 8;
  arr[1] = 9;
  arr[2] = 7;
  assertTrue(Largest.largest(arr) == 9);
}
```

## jUnit: Pros

- Unlike our tests so far, jUnit is more flexible. If one test fails (because an assertion failed), it continues to the next test.
- No main required. jUnit provides one in its junit.textui.TestRunner class.

## Digression: Java Code Organization

- Java provides a way to organize code
  - Not use a package: Default package
  - Using packages

## Packages

- Naming packages
  - The package names matches the directory name and can be hierarchical
    - java.util has classes stored in
      - [path-to-java-packages]/java/util

## Declaring your file to be within a package

- At the top of your source file, you add

```
package edu.um.eecs282.hello;

class HelloWorld {

   // …

}
```

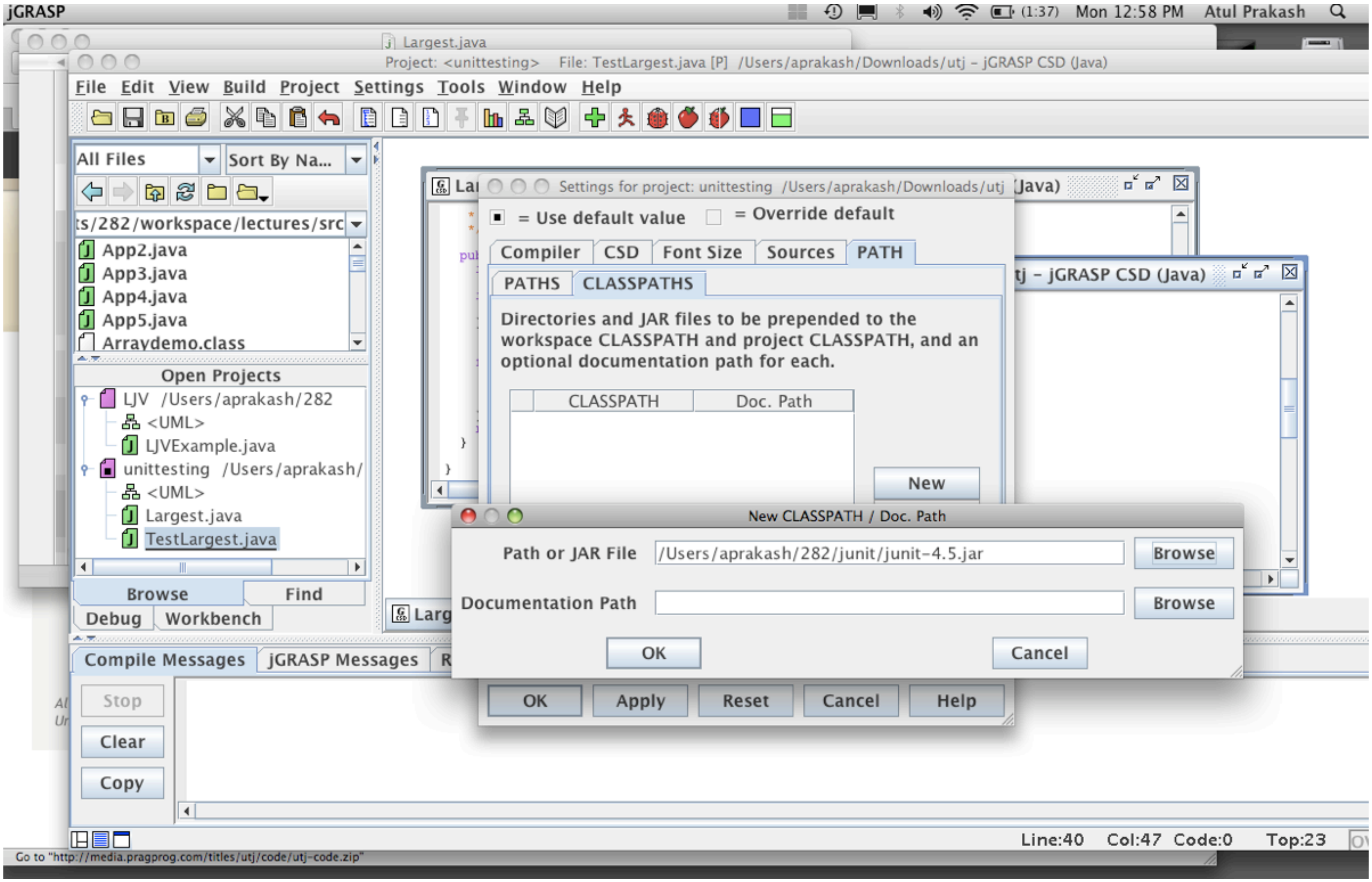
- For this example
  - File name = HelloWorld.java
  - Class name = HelloWorld.class
  - Location = [path-to]edu/um/eecs285/hello

# CLASSPATH

- jar files: Packages in Java are often compressed into a jar file. E.g., junit-4.5.jar.
  - To extract (you do not need to though), you can use the jar command:
    - jar xf junit-4.5.jar
- Java uses an environment variable called CLASSPATH to find jar files or other packages.
  - Format: dir1;dir2…;dirN on Windows
  - dir1:dir2:…:dirN on Unix/Mac OS.

# jGrasp and jUnit

- You need to tell jGrasp where the jUnit package is.
- Go to Settings -> Path/ClassPath -> Project …
- Add the directory containing junit's jar file to the path.



Settings for project: unittesting /Users/aprakash/Downloads/utj

■ = Use default value   ☐ = Override default

Compiler | CSD | Font Size | Sources | PATH

PATHS | CLASSPATHS

Directories and JAR files to be prepended to the workspace CLASSPATH and project CLASSPATH, and an optional documentation path for each.

| | CLASSPATH | Doc. Path |
|---|---|---|

New CLASSPATH / Doc. Path

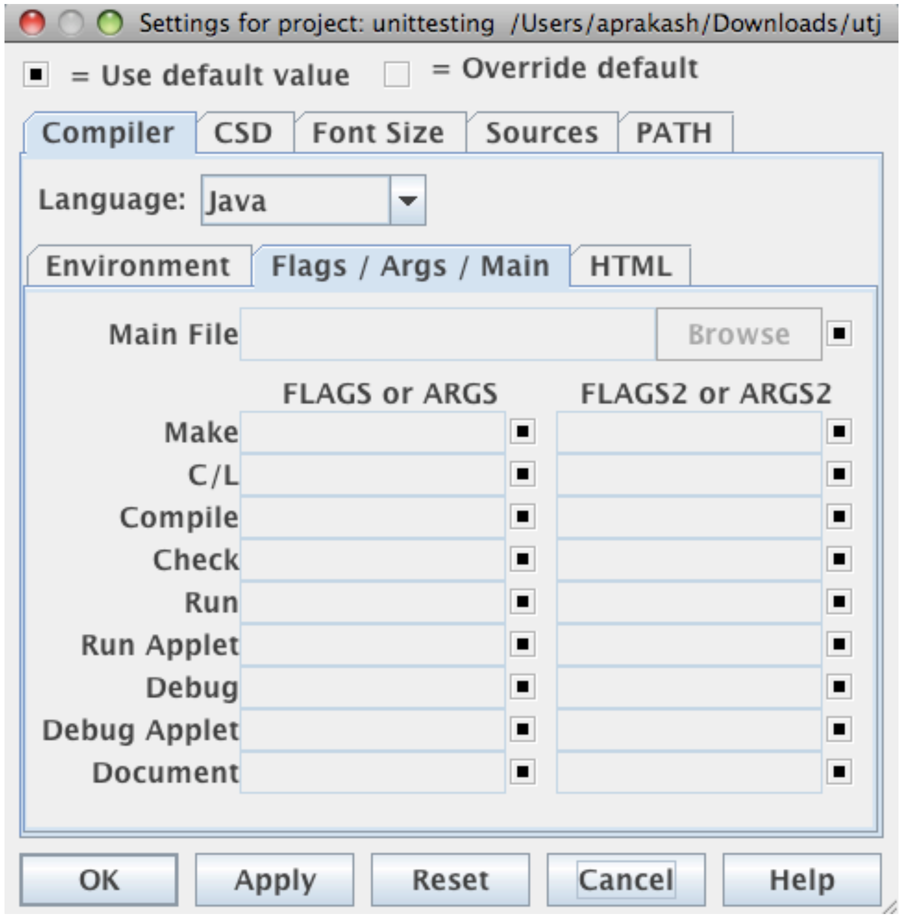
Path or JAR File: /Users/aprakash/282/junit/junit-4.5.jar

Documentation Path:

# jGrasp and jUnit

- jUnit provides its own main method for running tests. From command-line (provided your CLASSPATH includes junit's jar file):
  - java junit.textui.TestRunner TestLargest
- Tell jGrasp to use that:
  - Settings->Compiler Settings -> Project <unittesting>

- Select the Flags/Args tab under Compiler

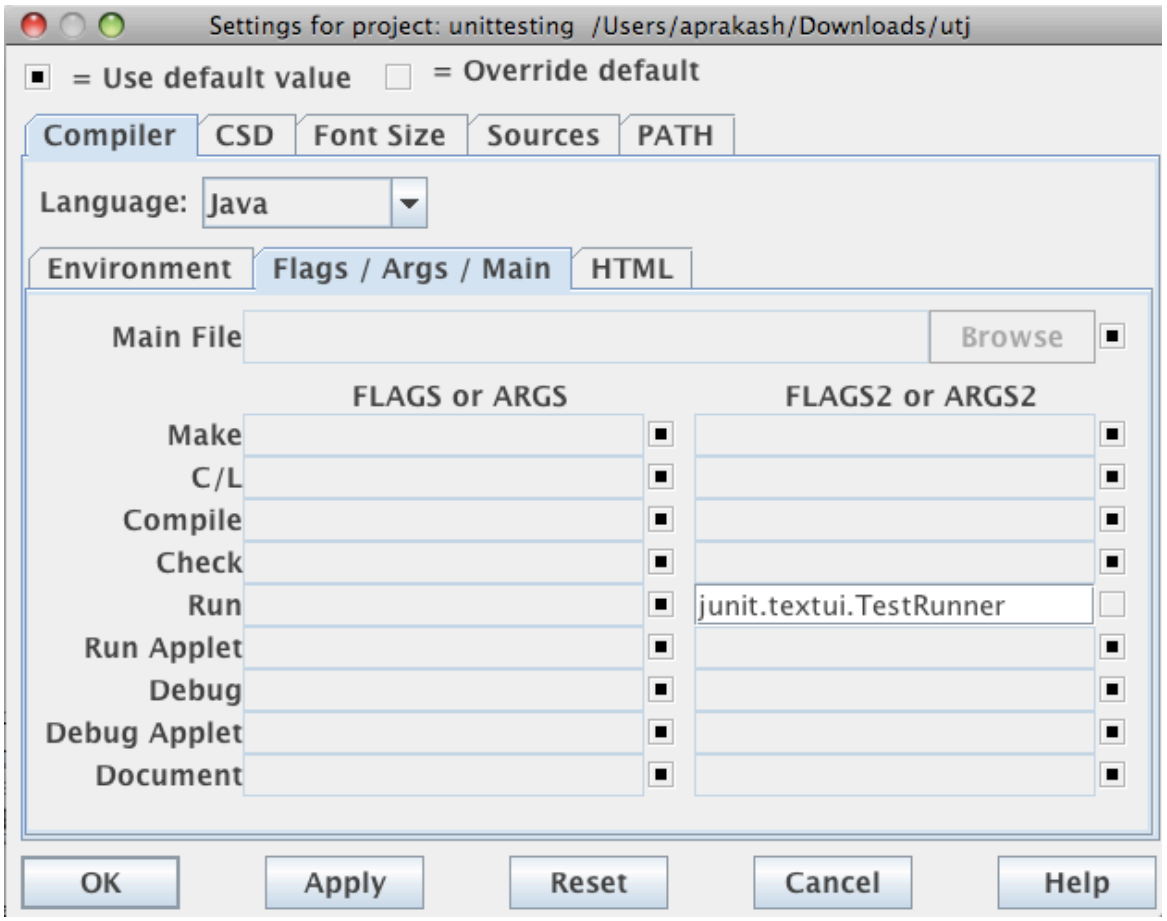


Language: Java

Main File ■

| | FLAGS or ARGS | | FLAGS2 or ARGS2 | |
|---|---|---|---|---|
| Make | | ■ | | ■ |
| C/L | | ■ | | ■ |
| Compile | | ■ | | ■ |
| Check | | ■ | | ■ |
| Run | | ■ | | ■ |
| Run Applet | | ■ | | ■ |
| Debug | | ■ | | ■ |
| Debug Applet | | ■ | | ■ |
| Document | | ■ | | ■ |

# Modify FLAG2 for Run



Settings for project: unittesting /Users/aprakash/Downloads/utj

■ = Use default value ☐ = Override default

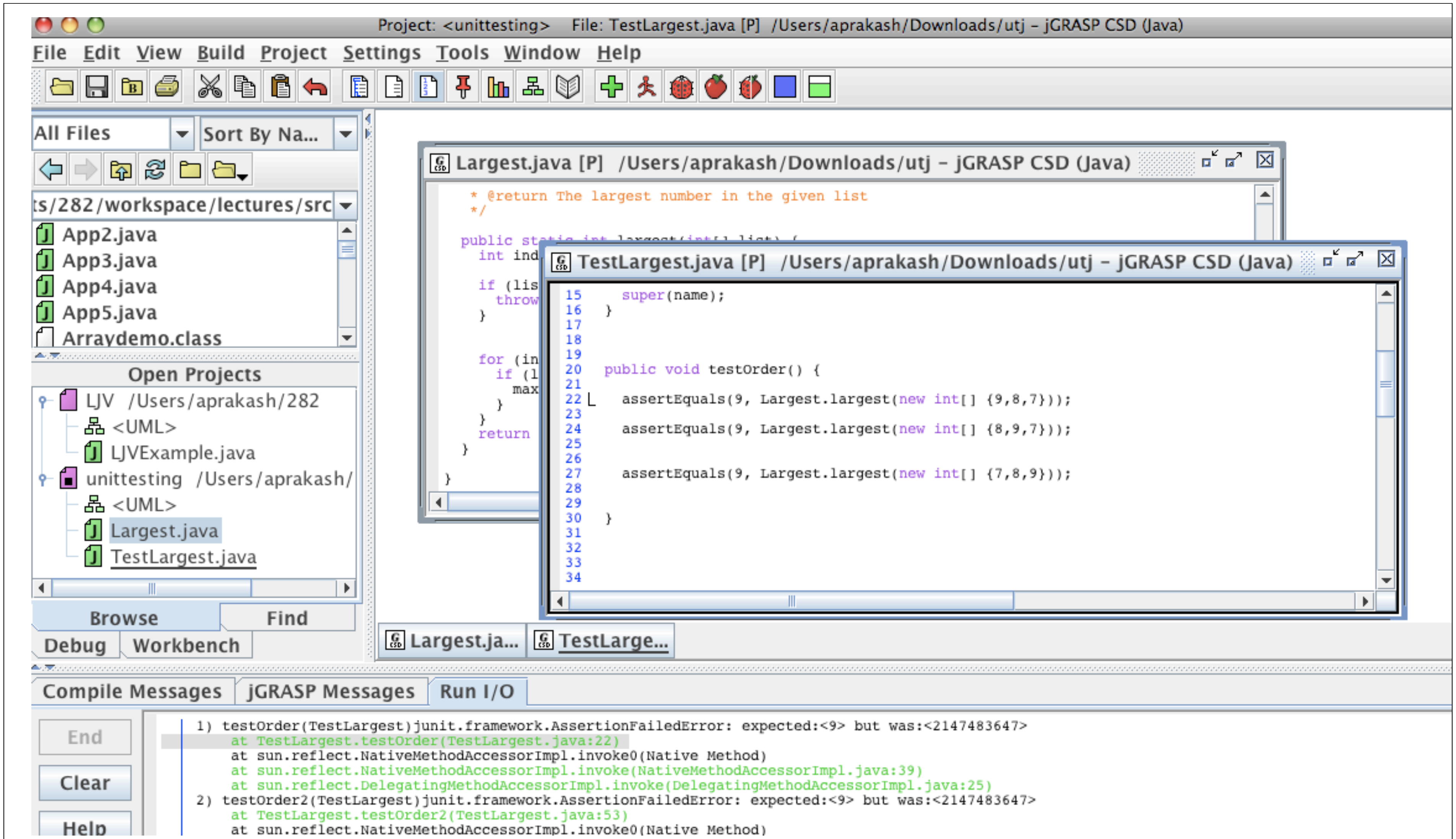
Language: Java

Main File ■

| | FLAGS or ARGS | | FLAGS2 or ARGS2 | |
|---|---|---|---|---|
| Make | | ■ | | ■ |
| C/L | | ■ | | ■ |
| Compile | | ■ | | ■ |
| Check | | ■ | | ■ |
| Run | | ■ | junit.textui.TestRunner | ☐ |
| Run Applet | | ■ | | ■ |
| Debug | | ■ | | ■ |
| Debug Applet | | ■ | | ■ |
| Document | | ■ | | ■ |

## Running the tests

- Compile the project and run TestLargest
- You should see the list of failed tests and the line number


Project: <unittesting>    File: TestLargest.java [P]  /Users/aprakash/Downloads/utj – jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
All Files
Sort By Na...
ts/282/workspace/lectures/src
App2.java
App3.java
App4.java
App5.java
Arraydemo.class
Open Projects
LJV  /Users/aprakash/282
<UML>
LJVExample.java
unittesting  /Users/aprakash/
<UML>
Largest.java
TestLargest.java
Browse
Find
Debug
Workbench
Largest.java [P]  /Users/aprakash/Downloads/utj – jGRASP CSD (Java)
TestLargest.java [P]  /Users/aprakash/Downloads/utj – jGRASP CSD (Java)
Largest.ja...
TestLarge...
Compile Messages
jGRASP Messages
Run I/O
End
Clear
Help

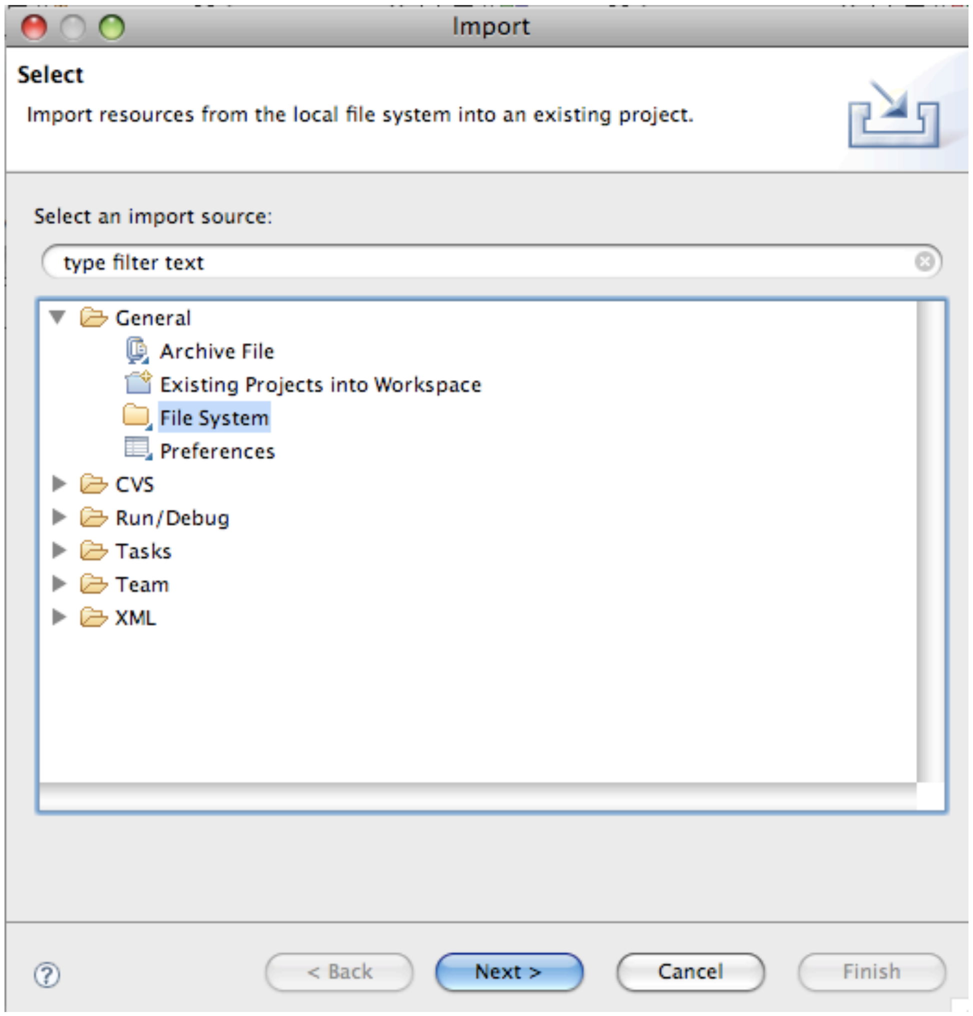
```
     * @return The largest number in the given list
     */

    public static int largest(int[] list) {
      int ind

      if (lis
        throw
      }

      for (in
        if (l
          max
      }
      return
    }

}
```

```
15      super(name);
16    }
17
18
19
20    public void testOrder() {
21
22      assertEquals(9, Largest.largest(new int[] {9,8,7}));
23
24      assertEquals(9, Largest.largest(new int[] {8,9,7}));
25
26
27      assertEquals(9, Largest.largest(new int[] {7,8,9}));
28
29
30    }
31
32
33
34
```

```
1) testOrder(TestLargest)junit.framework.AssertionFailedError: expected:<9> but was:<2147483647>
     at TestLargest.testOrder(TestLargest.java:22)
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
     at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
     at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
2) testOrder2(TestLargest)junit.framework.AssertionFailedError: expected:<9> but was:<2147483647>
     at TestLargest.testOrder2(TestLargest.java:53)
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

## Eclipse

- Eclipse is much easier. Simply select junit version 4 if you are given the choice (or the latest version) anywhere.
- File->Import -> General -> FileSystem
-



**Select**

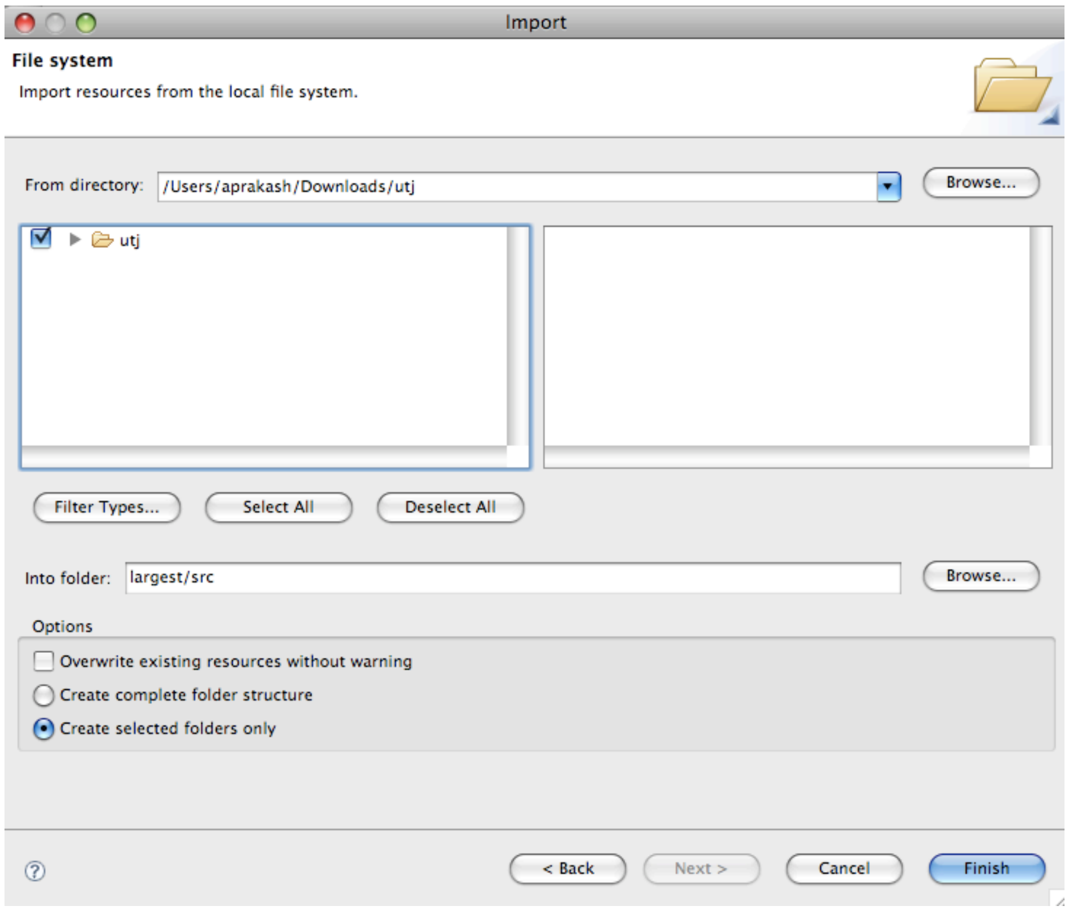
Import resources from the local file system into an existing project.

Select an import source:

type filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences
- CVS
- Run/Debug
- Tasks
- Team
- XML

# Select From directory



Import

File system

Import resources from the local file system.

From directory: /Users/aprakash/Downloads/utj

[✓] utj

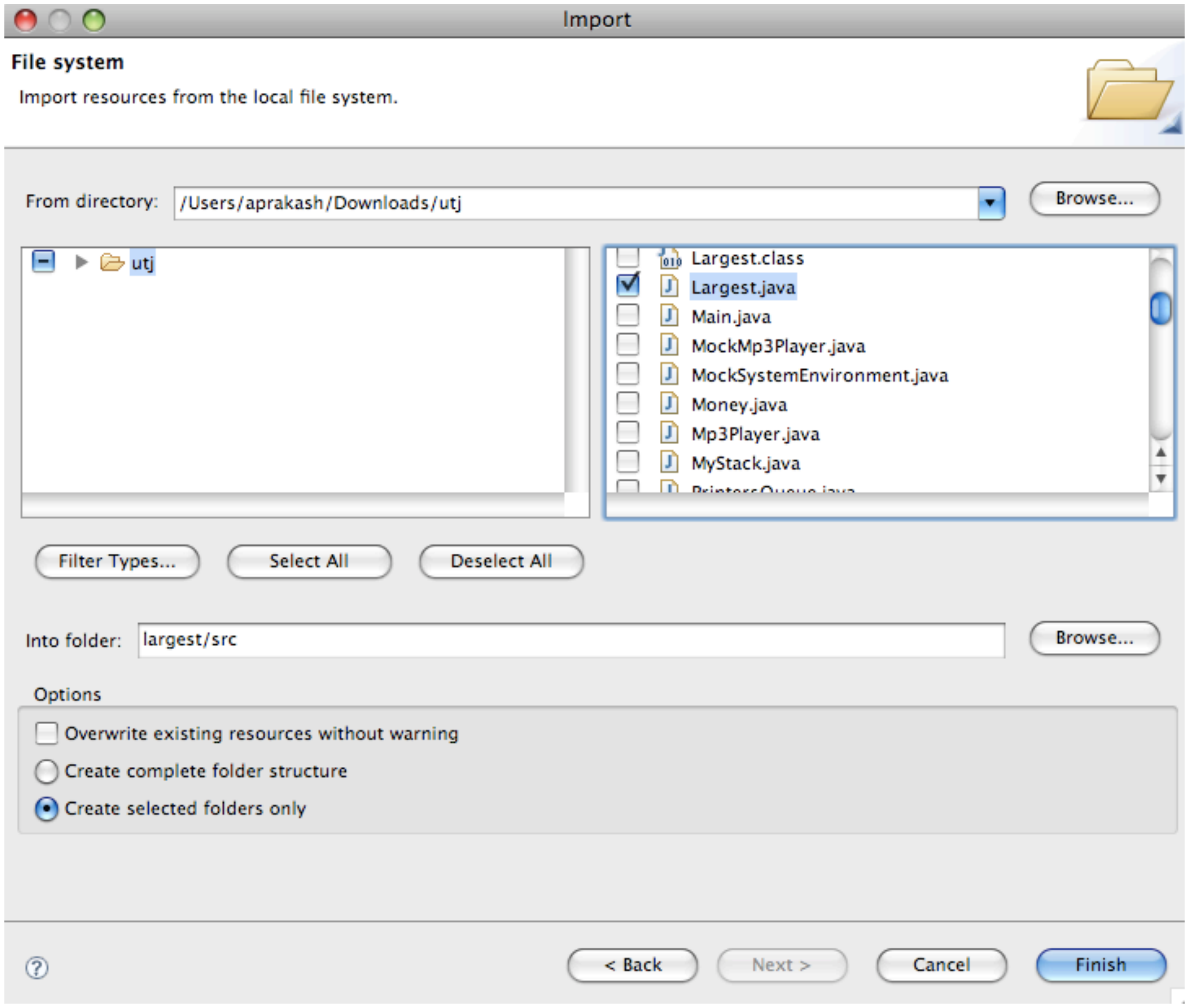
Into folder: largest/src

Options

- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

## Next steps

- Click on the directory "utj". You should see the files on the right side. Deselect all and select Largest.java and TestLargest.java.


Import

**File system**

Import resources from the local file system.

From directory: /Users/aprakash/Downloads/utj

utj

- [ ] Largest.class
- [✓] Largest.java
- [ ] Main.java
- [ ] MockMp3Player.java
- [ ] MockSystemEnvironment.java
- [ ] Money.java
- [ ] Mp3Player.java
- [ ] MyStack.java

Into folder: largest/src

Options

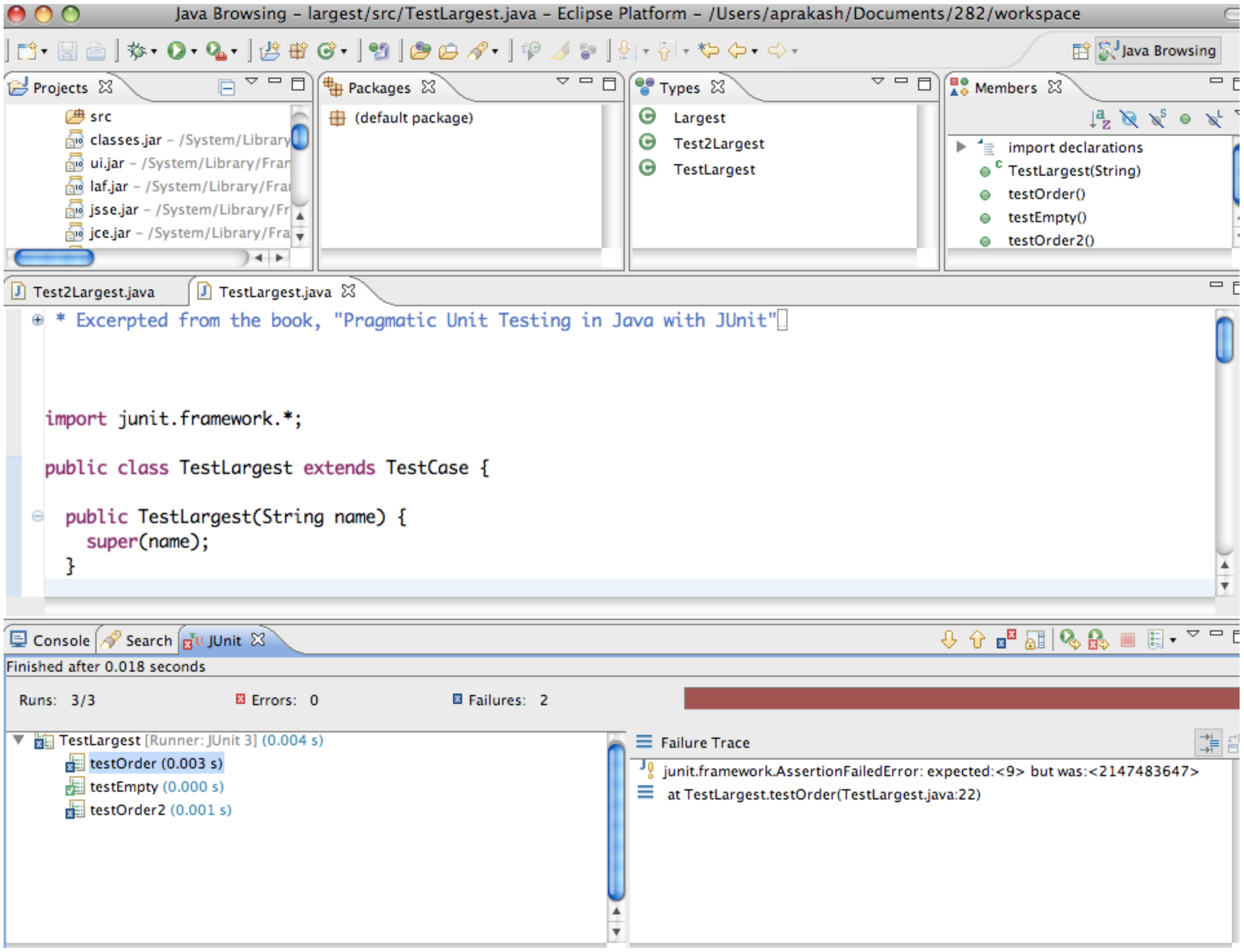
- [ ] Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

## Fix errors

- TestLargest.java may show errors. Hover your mouse over the word "junit" and select a fix to reorganize the project. You may have to add junit3 to the project in Eclipse.

## Run the test

- Run the TestLargest as a jUnit test. You should see a window showing the failed tests. Clicking on the failed line should take you to that line in TestLargest.
- Select Project -> Run, when you have selected TestLargest.


Java Browsing - largest/src/TestLargest.java - Eclipse Platform - /Users/aprakash/Documents/282/workspace
Java Browsing
Projects
src
classes.jar - /System/Library
ui.jar - /System/Library/Fran
laf.jar - /System/Library/Fra
jsse.jar - /System/Library/Fr
jce.jar - /System/Library/Fra
Packages
(default package)
Types
Largest
Test2Largest
TestLargest
Members
import declarations
TestLargest(String)
testOrder()
testEmpty()
testOrder2()
Test2Largest.java
TestLargest.java
Console
Search
JUnit
Failure Trace

```
* Excerpted from the book, "Pragmatic Unit Testing in Java with JUnit"


import junit.framework.*;

public class TestLargest extends TestCase {

  public TestLargest(String name) {
    super(name);
  }
```

```
Finished after 0.018 seconds

Runs:  3/3          Errors:  0          Failures:  2

TestLargest [Runner: JUnit 3] (0.004 s)
    testOrder (0.003 s)
    testEmpty (0.000 s)
    testOrder2 (0.001 s)

junit.framework.AssertionFailedError: expected:<9> but was:<2147483647>
at TestLargest.testOrder(TestLargest.java:22)
```

## Common Next Steps

- Read through your test and the code to see why the test failed.
- Set breakpoints and use a debugger if necessary to help identify the problems.
  - For jGrasp, you will have to change the FLAG2 settings for the Debugger.