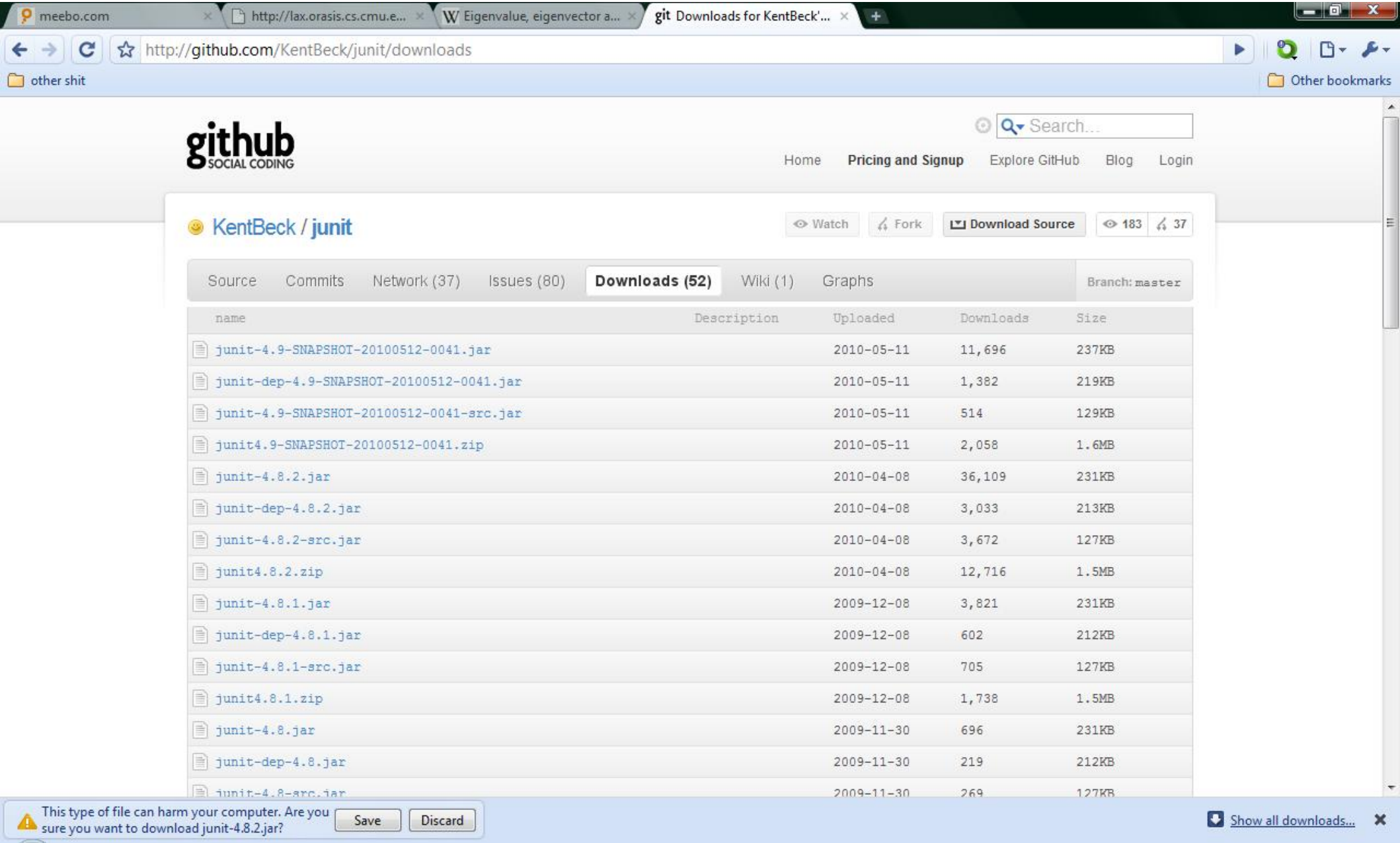
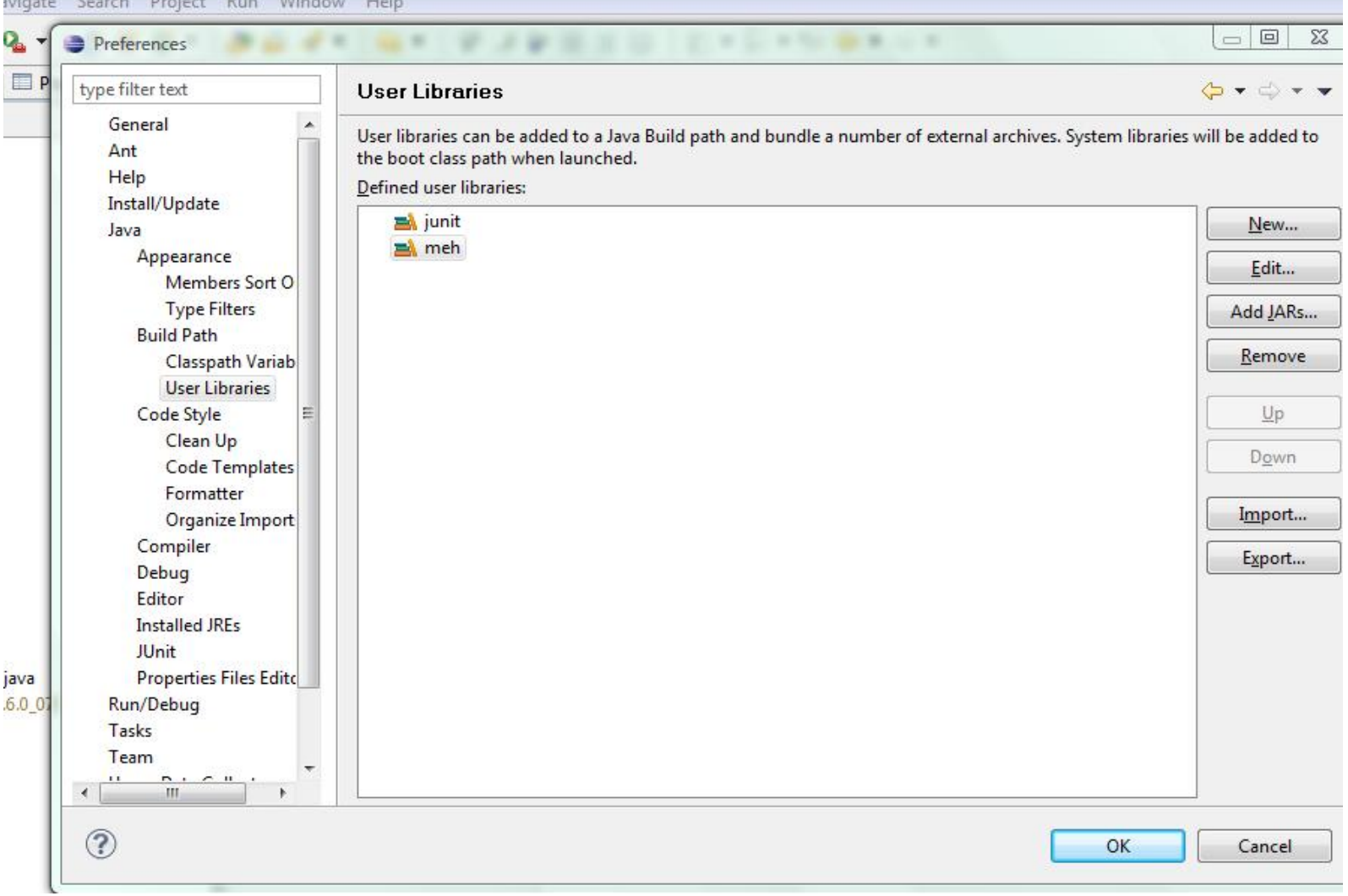
JUnit Tests

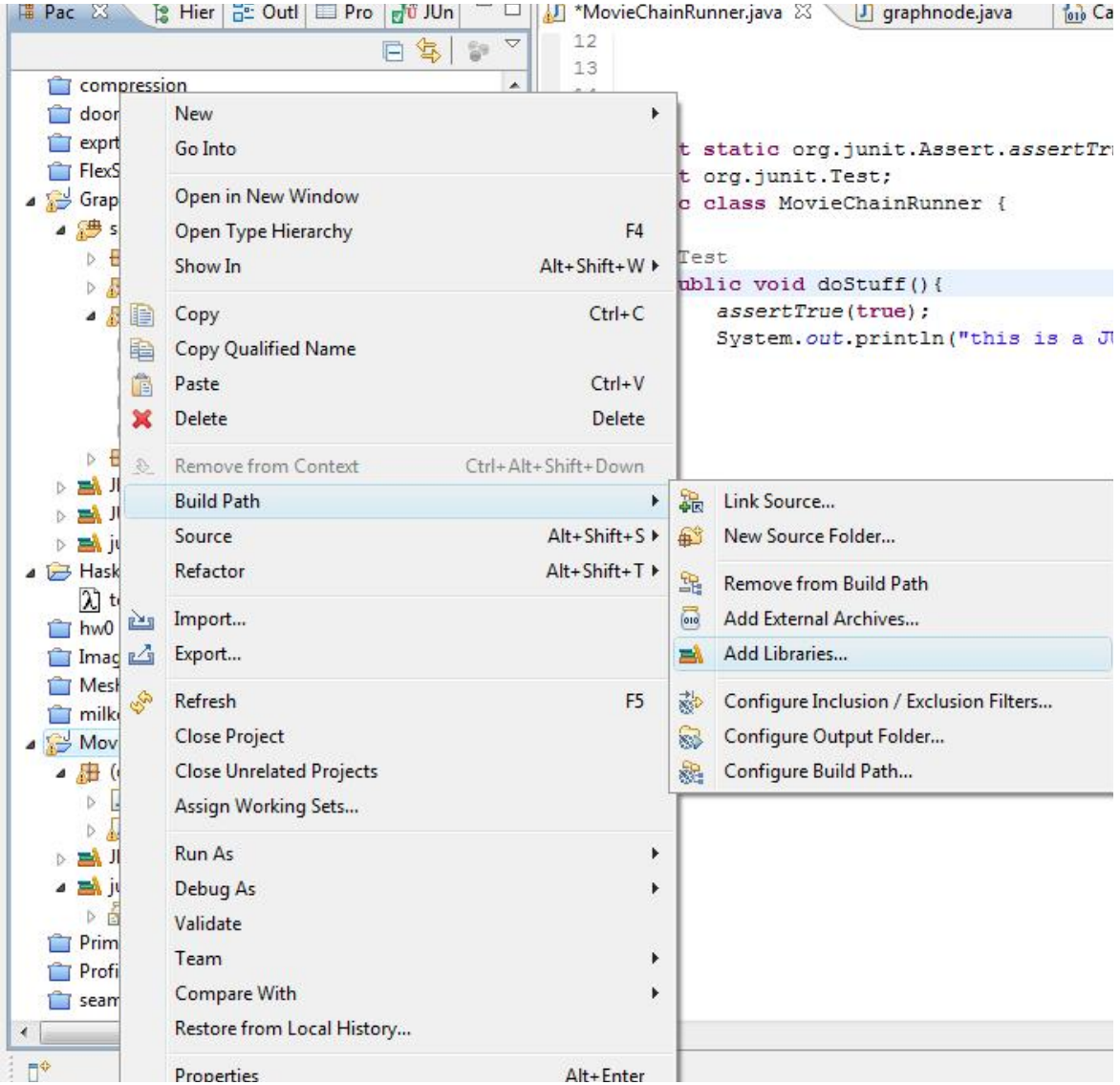
15-211 Recitation #2
02/07/10
TA: Matthew Mirman

Download JUnit 4 & Eclipse

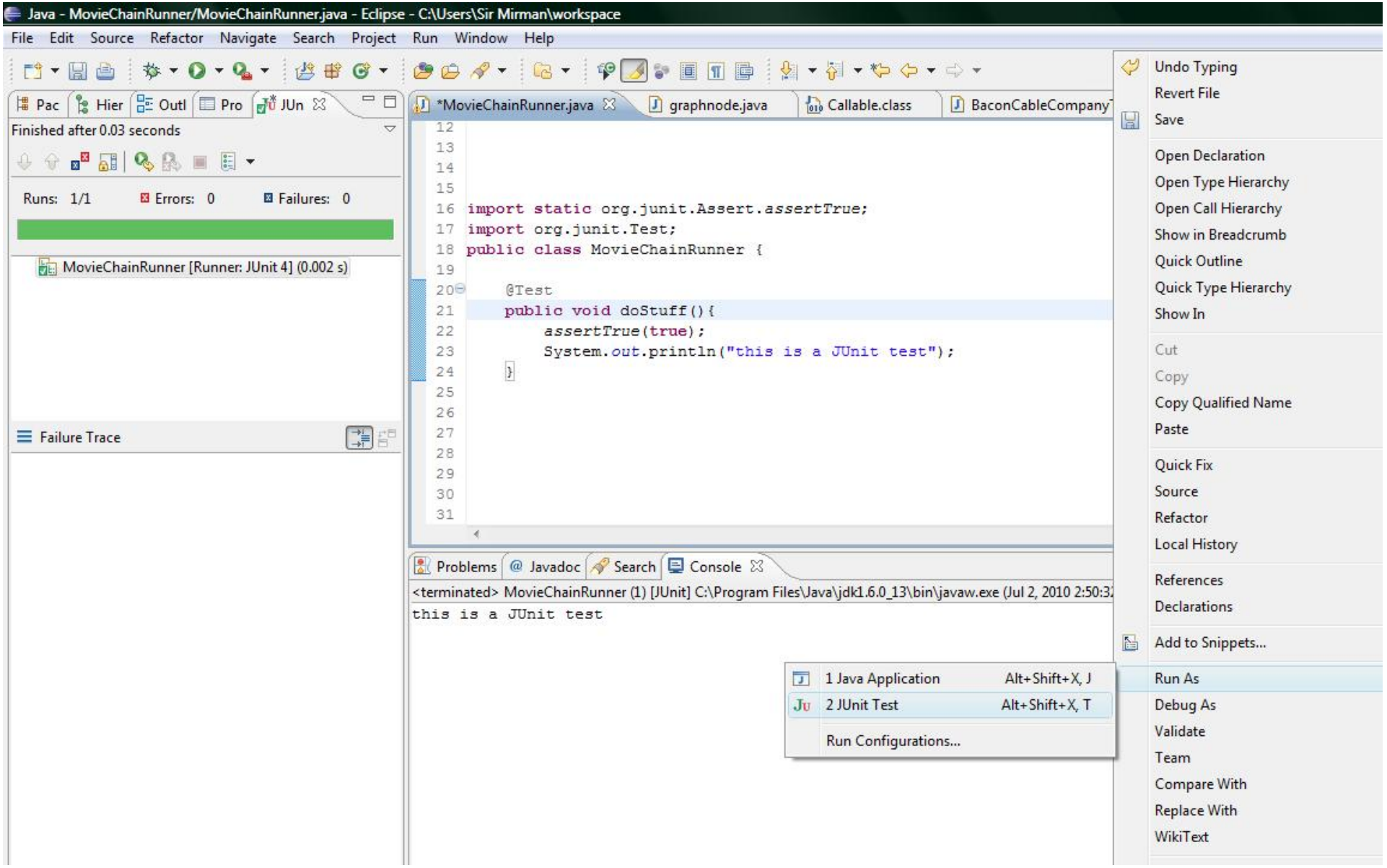




Window->Preferences->Java->
BuildPath->User Libraries->New
Call it JUnit.
Select it and Add Jar.



- Make a new project (or use an old one)
- Right click the project (or mac-click)
- Build Path->Add Libraries->User Library
- Select JUnit


Java - MovieChainRunner/MovieChainRunner.java - Eclipse - C:\Users\Sir Mirman\workspace
File Edit Source Refactor Navigate Search Project Run Window Help
Pac Hier Outl Pro JUn
Finished after 0.03 seconds
Runs: 1/1 Errors: 0 Failures: 0
MovieChainRunner [Runner: JUnit 4] (0.002 s)
Failure Trace
*MovieChainRunner.java graphnode.java Callable.class BaconCableCompany
12
13
14
15
16 import static org.junit.Assert.assertTrue;
17 import org.junit.Test;
18 public class MovieChainRunner {
19
20 @Test
21 public void doStuff(){
22 assertTrue(true);
23 System.out.println("this is a JUnit test");
24 }
25
26
27
28
29
30
31
Problems Javadoc Search Console
<terminated> MovieChainRunner (1) [JUnit] C:\Program Files\Java\jdk1.6.0_13\bin\javaw.exe (Jul 2, 2010 2:50:3
this is a JUnit test
1 Java Application Alt+Shift+X, J
2 JUnit Test Alt+Shift+X, T
Run Configurations...
Undo Typing
Revert File
Save
Open Declaration
Open Type Hierarchy
Open Call Hierarchy
Show in Breadcrumb
Quick Outline
Quick Type Hierarchy
Show In
Cut
Copy
Copy Qualified Name
Paste
Quick Fix
Source
Refactor
Local History
References
Declarations
Add to Snippets...
Run As
Debug As
Validate
Team
Compare With
Replace With
WikiText

Why Unit Test?

- Prevents subtle bugs in large software
- Makes writing lots of tests easier
- Makes running lots of tests easier
- Convince others your code is bug free
- Convince yourself that somebody else's code is bug free

Types of unit tests

1) Basic cases
   I) Test cases that should come out positive
   II) Test cases that should come out negative
   III) If doc specifies something should happen, write a test for it.
2) Bad Input
   I) Null Cases
   II) NaN/ +infinity/-infinity
   III) Check that exceptions are thrown in right places.
3) Edge Cases
   I) Test maximum and minimum inputs
   II) Zero sized arrays

Types of unit tests

1) Stress Test
   I) Test random inputs for large cases that you can't do yourself.
   II) Always use a static seed (or a few)! Otherwise You might not be able to reproduce the error.
      a. Use: Random m=new Random(5);
      b. Not: Math.random()
2) Timed Tests
   I) Make sure that programs halt when they should.
   II) Impossible to write a program to tell if another program stops (There is no oracle for the halting set)
   III) Instead, you can time how long things take

JUnit functionality

Operation	Functionality
assertTrue(boolean arg)	Ensure arg is true
assertFalse(boolean arg)	Ensure arg is false
assertEqual(A, B)	Ensure object A is equal to object B (using .equals())
assertNotNull(A)	...
assertSame(A, B)	Ensure A==B (checks if they point to the same object)
assertArrayEquals(A, B)	Ensure each element in A equals the same element in B (using .equals())
fail()	Always fails.

Normal Test

```
@Test
public void doStuff(){
   System.out.println("this is a JUnit test");
   assertTrue(true);
}
```

Expecting an exception

```
@Test(expected = NullPointerException.class)
public void nullTest() throws Exception {
    KevinBaconGameSolver.createUnweightedGameSolver(null);
}
```

Timed

```
@Test(timeout = 1000) //timeout is in ms
public void createQueueTest ()
{
    Queue<Object> q = new ArrayQueue<Object> ();
    assertEquals("Create new queue, call size()",0,q.size ());
    assertEquals("Create new queue, call peek()",null,q.peek ());
    assertEquals("Create new queue, call poll()",null,q.poll ());
}
```