Section 4:
Graphs and Testing

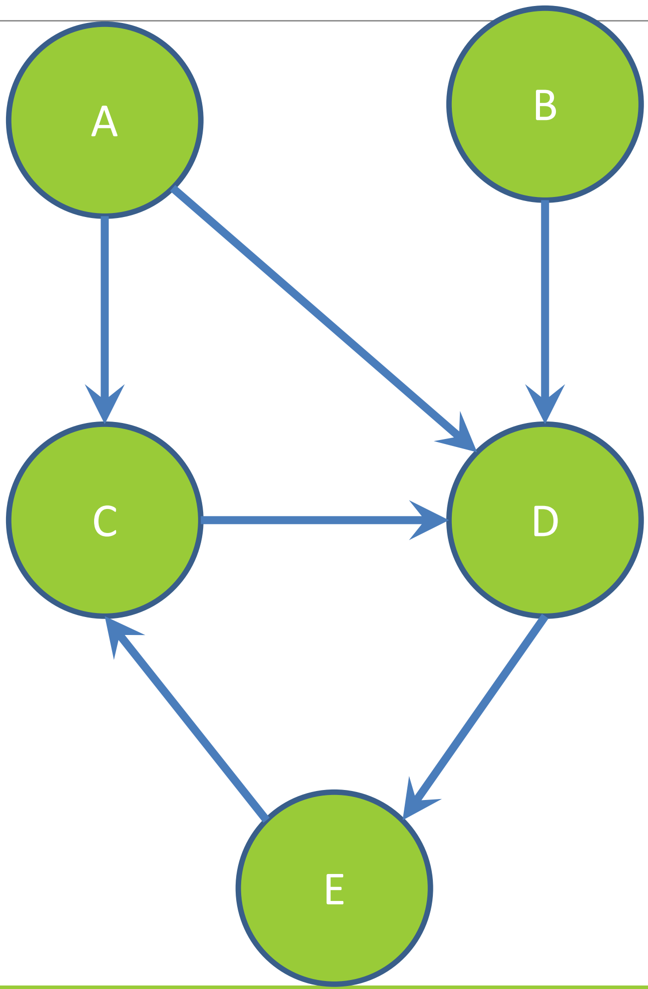
Slides by Erin Peach and Nick Carney

with material from Vinod Rathnam, Alex Mariakakis, Krysta Yousoufian, Mike Ernst, Kellen Donohue

AGENDA

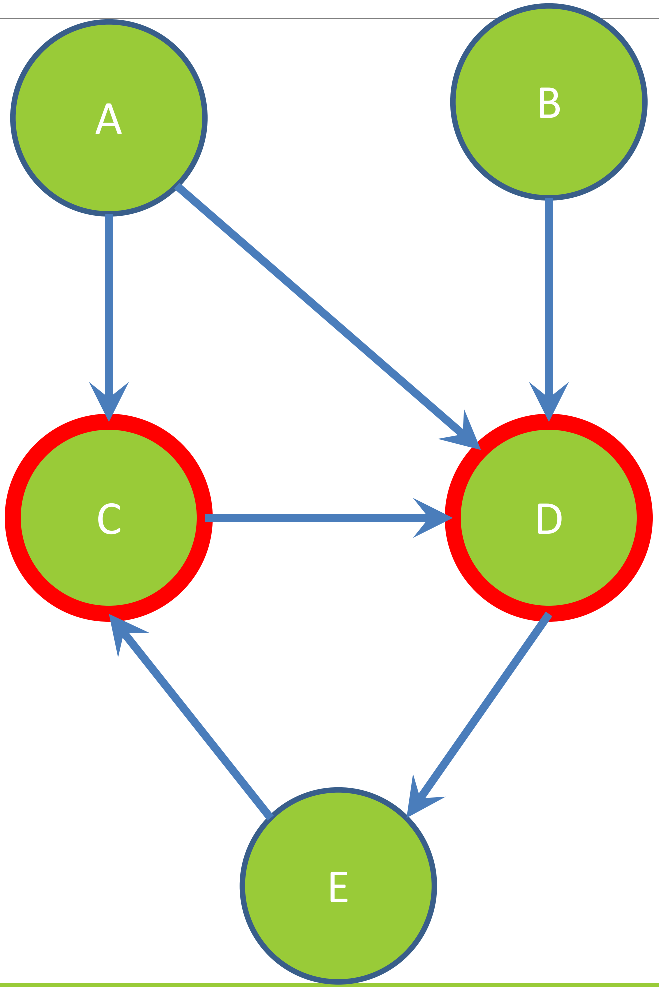
- Graphs
- JUnit Testing
- Test Script Language
- JavaDoc
- Code coverage in eclipse (OPTIONAL)

GRAPHS



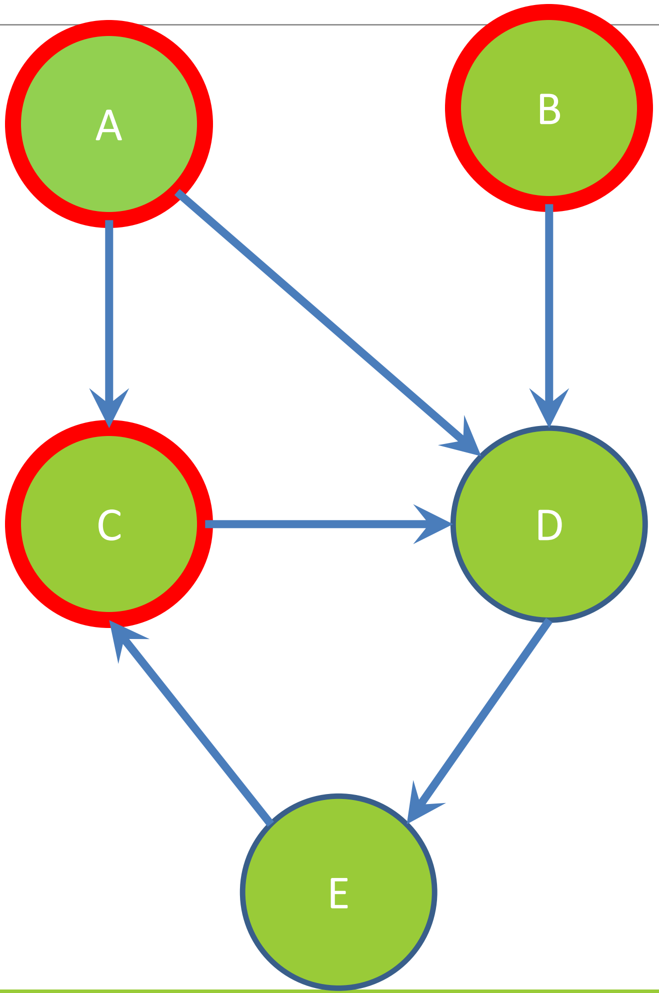
Nodes and Edges

GRAPHS



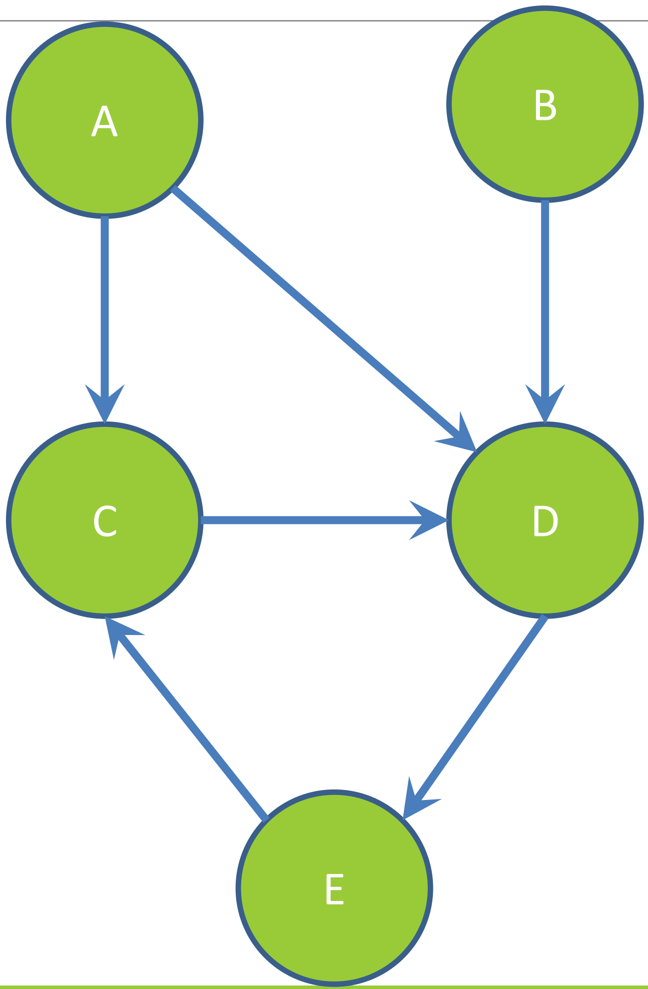
Children of A

GRAPHS



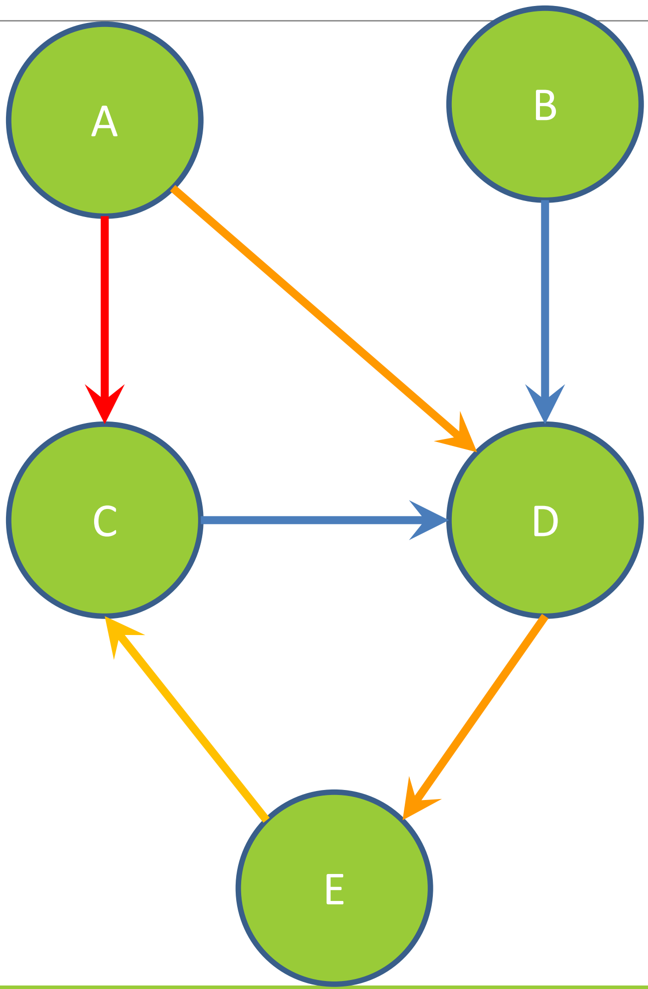
Parents of D

GRAPHS



Paths from A to C:

GRAPHS



Paths from A to C:

A -> C

A -> D -> E -> C

Shortest path from A to C?

LOOK AT ALL
THE BUGS WE WILL
FIX
memegenerator.net

FIX
ALL THE BUGS
DIYLOL.COM

Testing

INTERNAL VS. EXTERNAL TESTING

- Internal : JUnit
  - How you decide to implement the object
  - Checked with implementation tests
- External: test script
  - Your API and specifications
  - Testing against the specification
  - Checked with specification tests

A JUNIT TEST CLASS

- A method with `@Test` is flagged as a JUnit test
- All `@Test` methods run when JUnit runs

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

public class TestSuite {
   ...

    @Test
    public void TestName1() {
        ...
    }
}
```

USING JUNIT ASSERTIONS

- Verifies that a value matches expectations
  - `assertEquals(42, meaningOfLife());`
  - `assertTrue(list.isEmpty());`
- If the assert fails:
  - Test immediately terminates
  - Other tests in the test class are still run as normal
  - Results show “details” of failed tests (We’ll get to this later)

USING JUNIT ASSERTIONS

Assertion	Case for failure
`assertTrue(test)`	the boolean test is false
`assertFalse(test)`	the boolean test is true
`assertEquals(expected, actual)`	the values are not equal
`assertSame(expected, actual)`	the values are not the same (by ==)
`assertNotSame(expected, actual)`	the values are the same (by ==)
`assertNull(value)`	the given value is not null
`assertNotNull(value)`	the given value is null

- And others: http://www.junit.org/apidocs/org/junit/Assert.html
- Each method can also be passed a string to display if it fails:
  - `assertEquals("message", expected, actual)`

CHECKING FOR EXCEPTIONS

- Verify that a method throws an exception when it should:
  - Passes if specified exception is thrown, fails otherwise
- Only time it’s OK to write a test without a form of `asserts`

```
@Test(expected=IndexOutOfBoundsException.class)
public void testGetEmptyList() {
   List<String> list = new ArrayList<String>();
   list.get(0);
}
```

“But don’t I need to create a list before checking if I’ve successfully added to it?”

SETUP AND TEARDOWN

- Methods to run before/after each test case method is called:

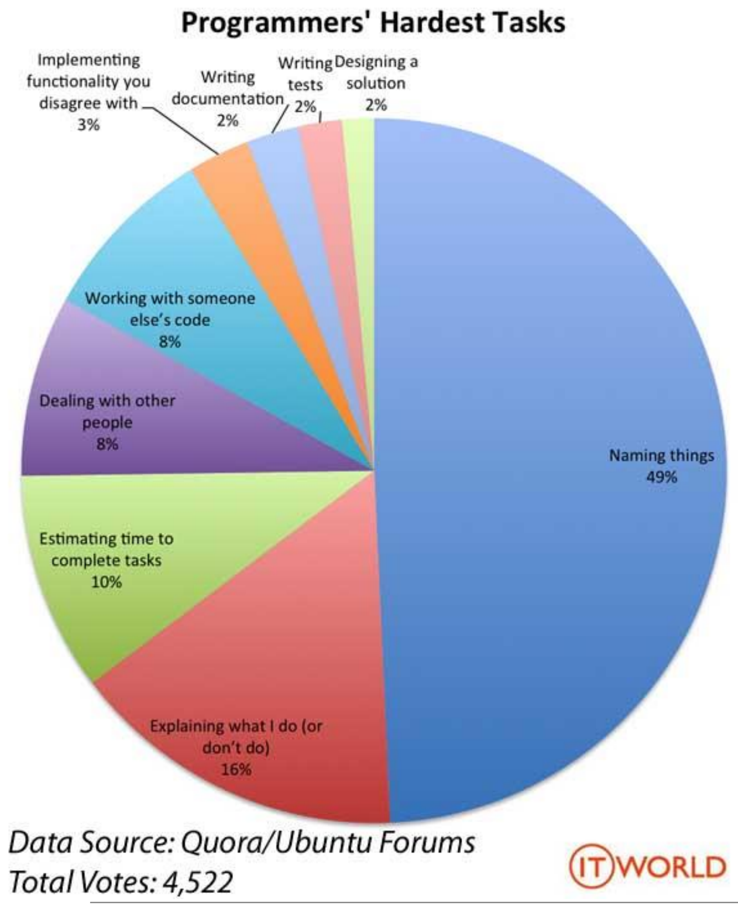
```
@Before
public void name() { ... }
@After
public void name() { ... }
```

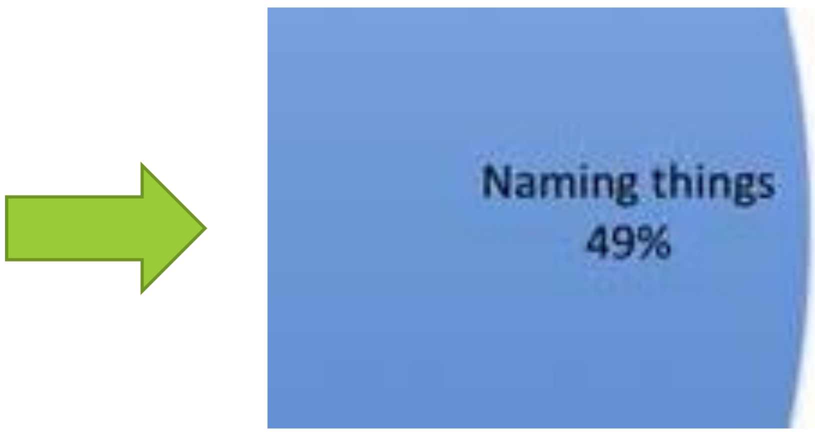
- Methods to run once before/after the entire test class runs:

```
@BeforeClass
public static void name() { ... }
@AfterClass
public static void name() { ... }
```

SETUP AND TEARDOWN

```
public class Example {
      List empty;

      @Before
      public void initialize() {
            empty = new ArrayList();
      }
      @Test
      public void size() {
            ...
      }
      @Test
      public void remove() {
            ...
      }
}
```





Test Writing Etiquette

The Rules

1. Don't Repeat Yourself
   - Use constants and helper methods
2. Be Descriptive
   - Take advantage of message, expected, and actual values
3. Keep Tests Small
   - Isolate bugs one at a time – Test halts after failed assertion
4. Be Thorough
   - Test big, small, boundaries, exceptions, errors

LET’S PUT IT ALL TOGETHER!

```
public class DateTest {

    ...
    // Test addDays when it causes a rollover between months
    @Test
    public void testAddDaysWrapToNextMonth() {
        Date actual = new Date(2050, 2, 15);
        actual.addDays(14);
        Date expected = new Date(2050, 3, 1);
        assertEquals("date after +14 days", expected,
                actual);
    }
```

How To Create JUnit Test Classes

- Right-click hw5.test -> New -> JUnit Test Case
- **Important**: Follow naming guidelines we provide
- Demo

JUNIT ASSERTS VS. JAVA ASSERTS

- We’ve just been discussing JUnit assertions so far
- Java itself has assertions

```
public class LitterBox {
  ArrayList<Kitten> kittens;

  public Kitten getKitten(int n) {
      assert(n >= 0);
      return kittens(n);
  }
}
```

ASSERTIONS VS. EXCEPTIONS

```
public class LitterBox {
   ArrayList<Kitten> kittens;

   public Kitten getKitten(int n) {
      assert(n >= 0);
      return kittens(n);
   }
}
```

```
public class LitterBox {
   ArrayList<Kitten> kittens;

   public Kitten getKitten(int n) {
      try {
         return kittens(n);
      } catch(Exception e) {
      }
   }
}
```

- Assertions should check for things that should never happen
- Exceptions should check for things that might happen
- “Exceptions address the robustness of your code, while assertions address its correctness”

REMINDER: ENABLING ASSERTS IN ECLIPSE

To enable asserts:

Go to Run -> Run Configurations… -> Arguments tab -> input **-ea** in VM arguments section

Do this for every test file

Expensive CheckReps

- Ant Validate and Staff Grading will have assertions enabled
- But sometimes a checkRep can be expensive
  - For example, looking at each node in a Graph with a large number of nodes
- This could cause the grading scripts to timeout

Expensive CheckReps

- Before your final commit, remove the checking of expensive parts of your checkRep or the checking of your checkRep entirely
- Example: boolean flag and structure your checkRep as so:

```
private void checkRep() {
    cheap-stuff
    if(DEBUG_FLAG) { // or can have this for entire checkRep
      expensive-stuff
    }
    cheap-stuff
    ...
```

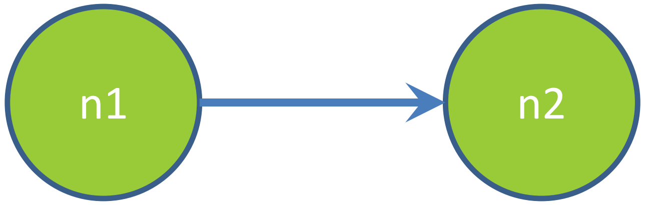
EXTERNAL TESTS:
TEST SCRIPT LANGUAGE

TEST SCRIPT LANGUAGE

- Text file with one command listed per line
- First word is always the command name
- Remaining words are arguments
- Commands will correspond to methods in your code

TEST SCRIPT LANGUAGE (ex .test file)

```
# Create a graph
CreateGraph graph1

# Add a pair of nodes
AddNode graph1 n1
AddNode graph1 n2

# Add an edge
AddEdge graph1 n1 n2 e1

# Print the nodes in the graph
and the outgoing edges from n1
ListNodes graph1
ListChildren graph1 n1
```



$$\begin{tikzcd} n1 \arrow[r] & n2 \end{tikzcd}$$

How To Create Specification Tests

- Create .test and .expected file pairs under hw5.test
- Implement parts of HW5TestDriver
  - driver connects commands from .test file to your Graph implementation to the output which is matched with .expected file
- Run all tests by running SpecificationTests.java
  - Note: staff will have our own .test and .expected pairs to run with your code
  - **Do not** hardcode .test/.expected pairs to pass, but instead make sure the format in hw5 instructions is correctly followed

DEMO: TEST SCRIPT LANGUAGE

JAVADOC API

- Now you can generate the JavaDoc API for your code
- Instructions in the Editing/Compiling Handout
- Demo: Generate JavaDocs

CODE COVERAGE TOOL (OPTIONAL)

Code coverage

- One measure of how well you've tested your code
- Different kinds:
  - Statements
  - Branches
  - Paths
  - (see lecture slides on testing for more detail)

When is coverage knowledge useful?

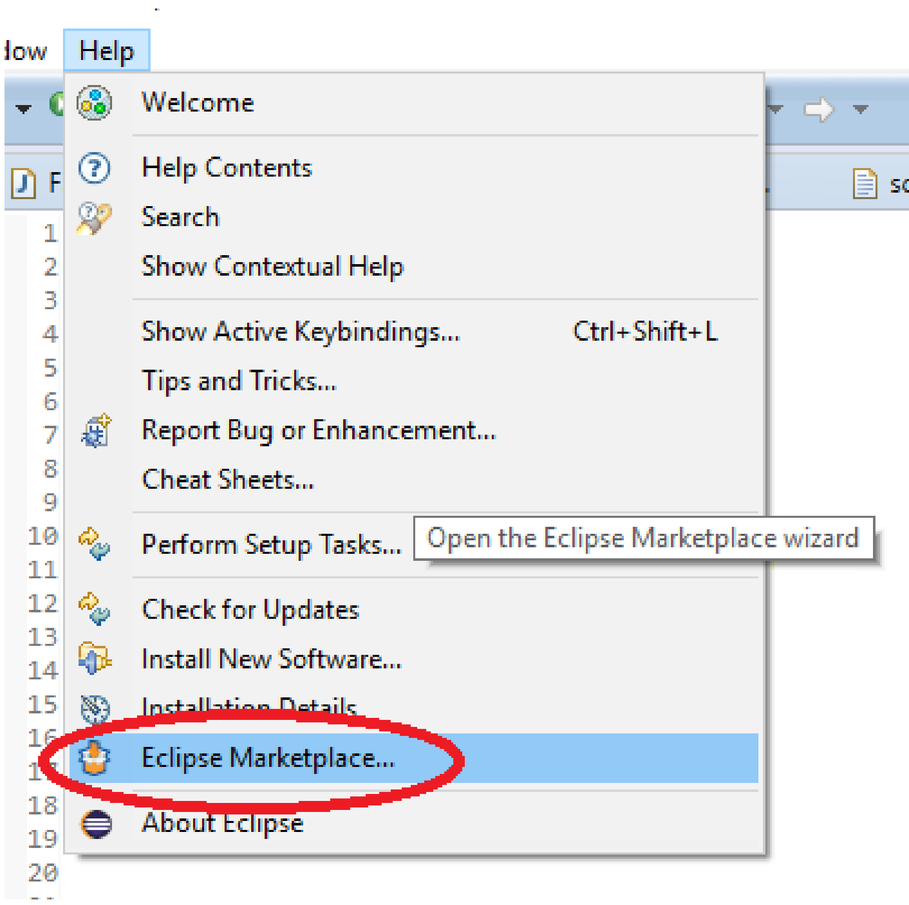
- What if testInductiveCase were missing from FibonacciTest.java and getFibTerm(int n) in Fibonacci.java were still returning the difference instead of the sum of previous terms?
  - All tests pass, but code isn't correct!

Code Coverage in Eclipse

- EclEmma (Ecl like Eclipse) lets you visualize statement and branch code coverage
  - http://www.eclemma.org/installation.html
  - The next couple slides will go over installation option 1

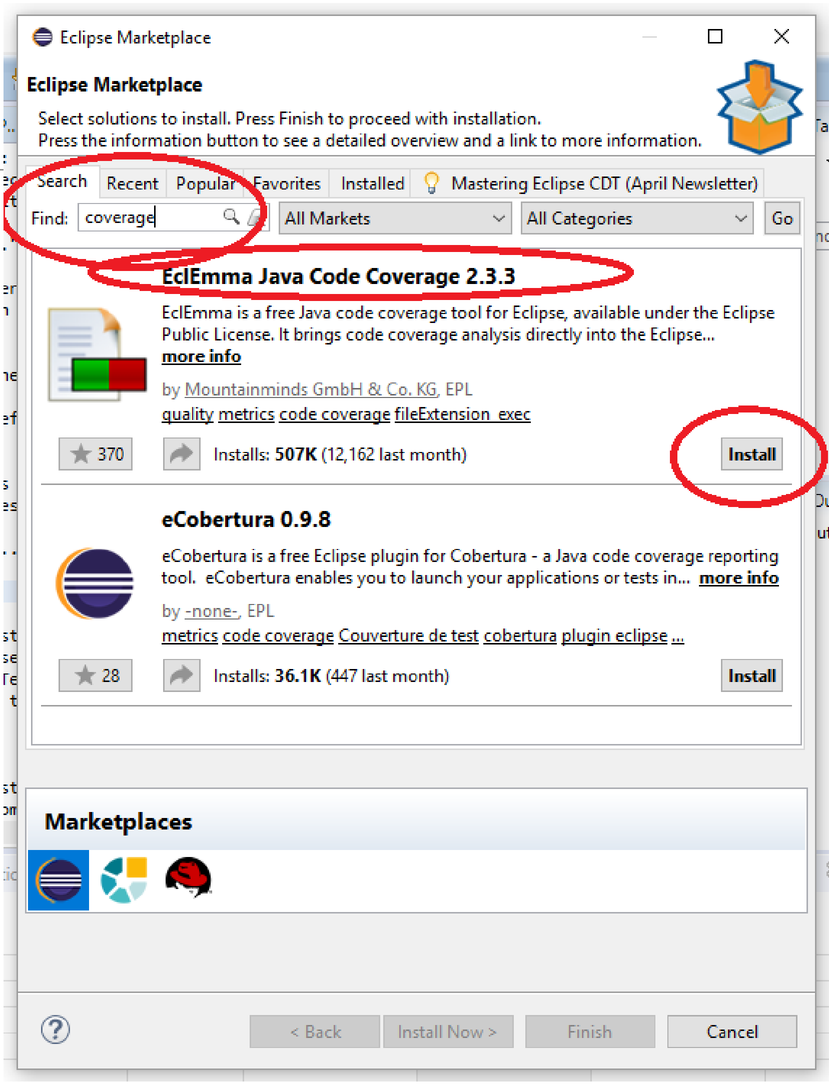
Installation Step 1

- From eclipse, go to the "Help" menu, and then choose "Eclipse Marketplace…"



Installation Step 2

- Search for "coverage," then when "EclEmma Java Code Coverage" shows up, click "Install"
- Then accept the license agreement, hit Finish, and restart Eclipse


Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.

EclEmma Java Code Coverage 2.3.3

EclEmma is a free Java code coverage tool for Eclipse, available under the Eclipse Public License. It brings code coverage analysis directly into the Eclipse... more info

by Mountainminds GmbH & Co. KG, EPL

quality metrics code coverage fileExtension_exec

370 Installs: **507K** (12,162 last month) Install

eCobertura 0.9.8

eCobertura is a free Eclipse plugin for Cobertura - a Java code coverage reporting tool. eCobertura enables you to launch your applications or tests in... more info

by -none-, EPL

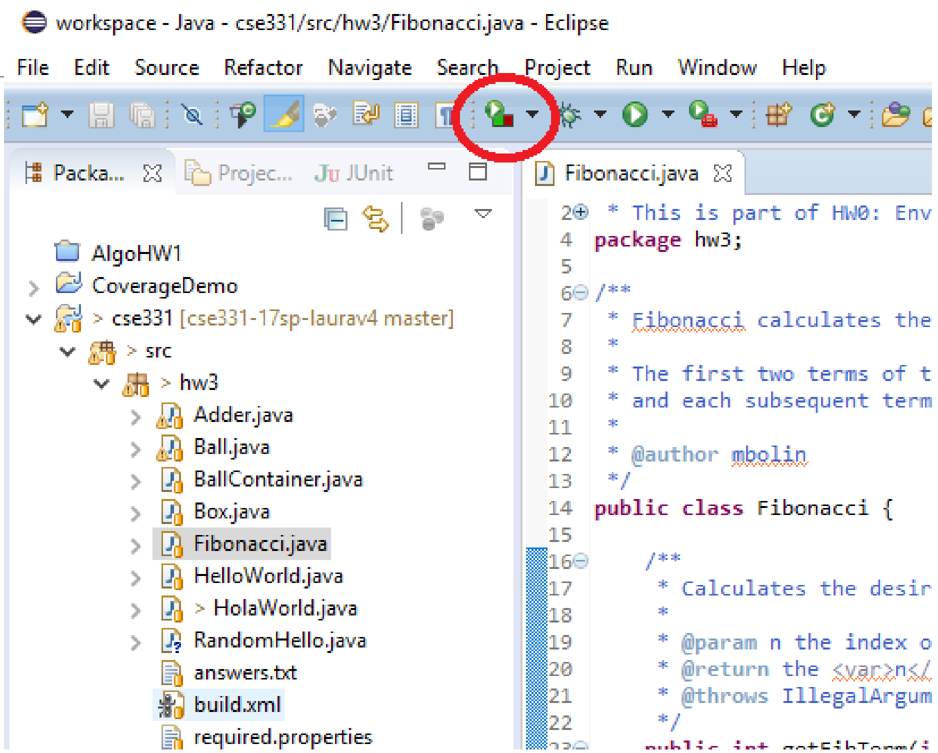
metrics code coverage Couverture de test cobertura plugin eclipse ...

28 Installs: **36.1K** (447 last month) Install

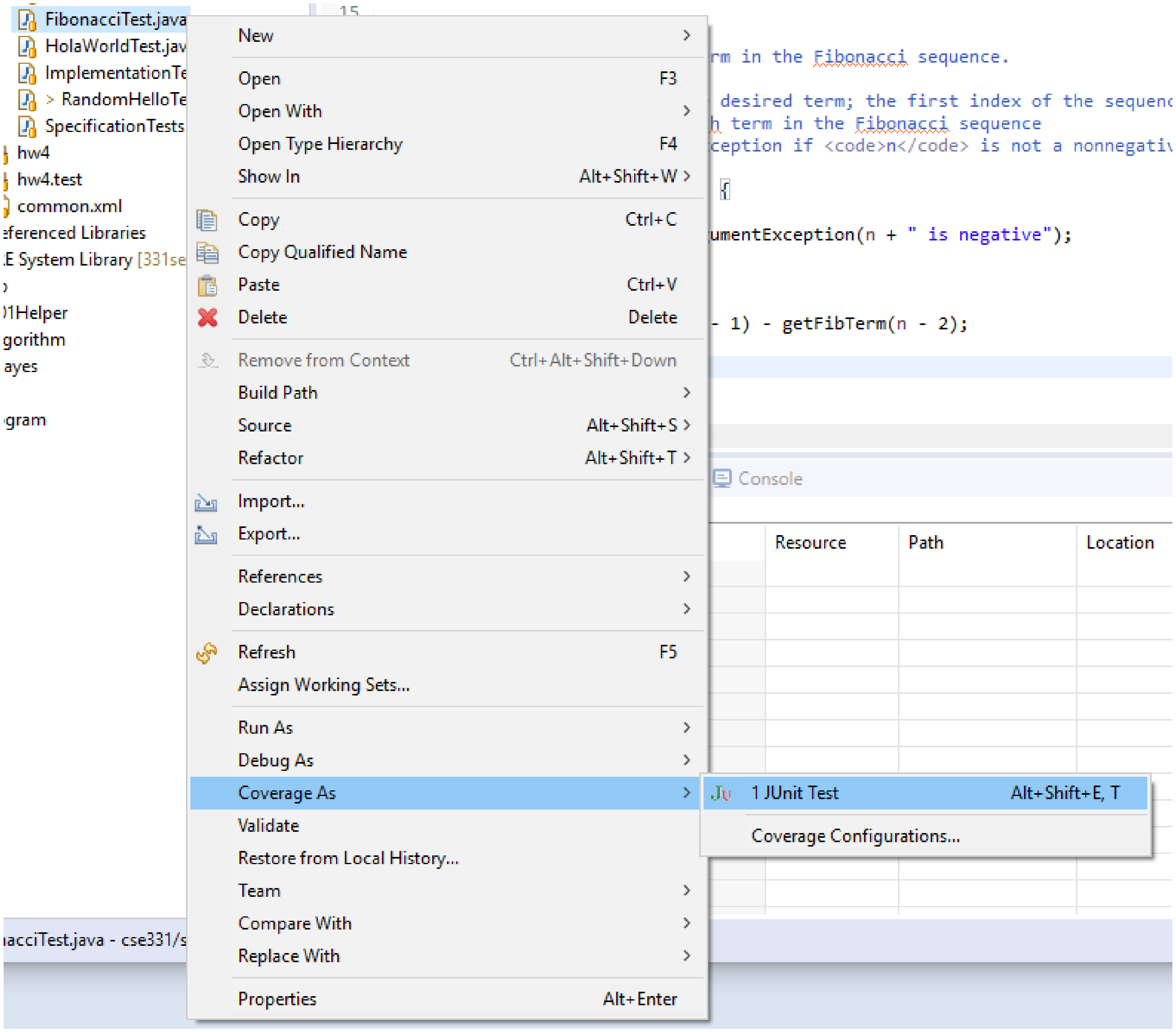
Marketplaces

Using it

- From the top bar, click the coverage arrow instead of the run arrow
- Or, right-click on a .java file and chose “Coverage as” instead of “Run as”
  - (see next slide for screenshot)



```
 * This is part of HW0: Env
package hw3;

/**
 * Fibonacci calculates the
 *
 * The first two terms of t
 * and each subsequent term
 *
 * @author mbolin
 */
public class Fibonacci {

    /**
     * Calculates the desir
     *
     * @param n the index o
     * @return the <var>n</
     * @throws IllegalArgum
     */
```



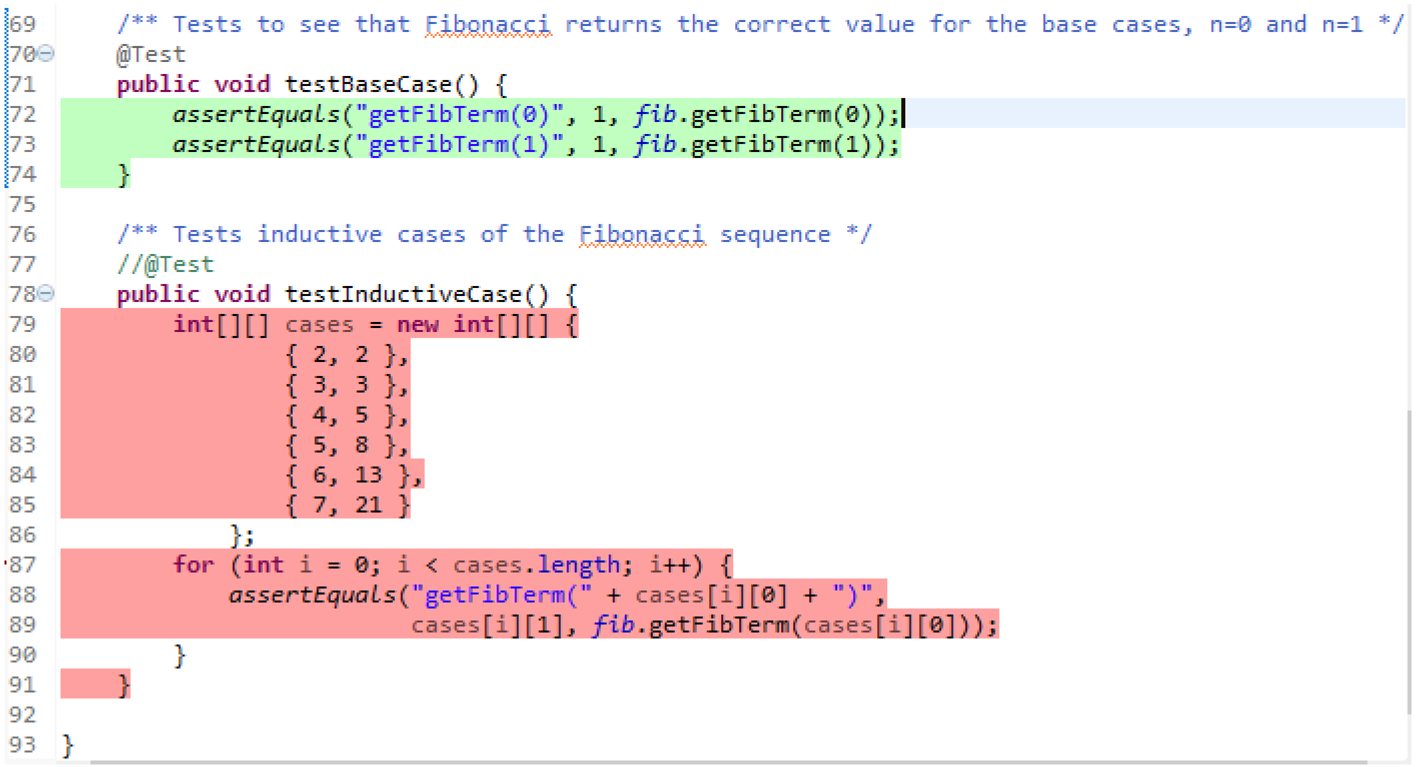
```
rm in the Fibonacci sequence.

 desired term; the first index of the sequenc
h term in the Fibonacci sequence
ception if <code>n</code> is not a nonnegativ

 {

umentException(n + " is negative");



- 1) - getFibTerm(n - 2);
```

What it looks like

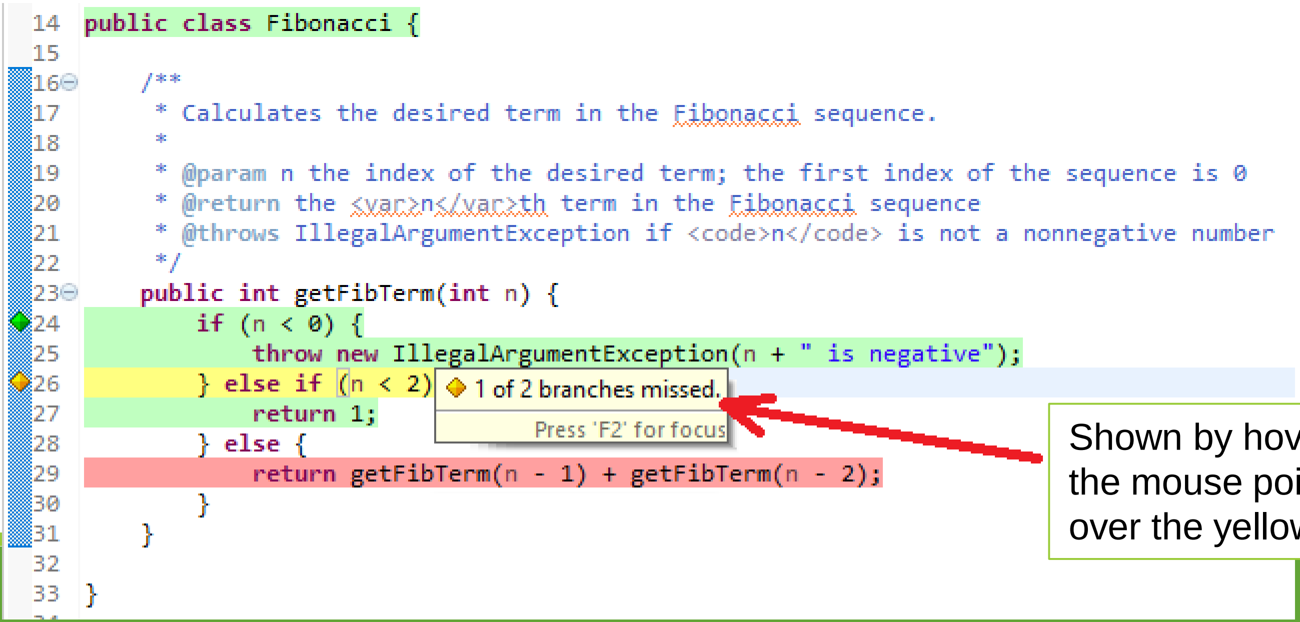
- Basic idea:
  - Highlights lines of code green (covered), yellow (partially covered—missing some branch(es)), or red (no coverage)
  - Also has a view at the bottom with percent of covered code, and you can expand folders and/or packages down to the individual file level
- Demo with hw3 Fibonacci.java and FibonacciTest.java

Questions to help explore the tool

- What happens if you run the coverage view after you comment out the @Test before testInductiveCase in FibonacciTest.java?
  - What color(s) do the lines of that method turn?
  - What color(s) do the lines of the method getFibTerm(int n) in FibonacciTest.java turn?



```java
    /** Tests to see that Fibonacci returns the correct value for the base cases, n=0 and n=1 */
    @Test
    public void testBaseCase() {
        assertEquals("getFibTerm(0)", 1, fib.getFibTerm(0));
        assertEquals("getFibTerm(1)", 1, fib.getFibTerm(1));
    }

    /** Tests inductive cases of the Fibonacci sequence */
    //@Test
    public void testInductiveCase() {
        int[][] cases = new int[][] {
                { 2, 2 },
                { 3, 3 },
                { 4, 5 },
                { 5, 8 },
                { 6, 13 },
                { 7, 21 }
            };
        for (int i = 0; i < cases.length; i++) {
            assertEquals("getFibTerm(" + cases[i][0] + ")",
                         cases[i][1], fib.getFibTerm(cases[i][0]));
        }
    }

}
```



```java
public class Fibonacci {

    /**
     * Calculates the desired term in the Fibonacci sequence.
     *
     * @param n the index of the desired term; the first index of the sequence is 0
     * @return the <var>n</var>th term in the Fibonacci sequence
     * @throws IllegalArgumentException if <code>n</code> is not a nonnegative number
     */
    public int getFibTerm(int n) {
        if (n < 0) {
            throw new IllegalArgumentException(n + " is negative");
        } else if (n < 2) {
            return 1;
        } else {
            return getFibTerm(n - 1) + getFibTerm(n - 2);
        }
    }

}
```

◆ 1 of 2 branches missed.
Press 'F2' for focus

Shown by hovering the mouse pointer over the yellow line

So, coverage is…

- Good for catching things like
  - Missing @Test before a test method
  - Finding branches/statements you’re forgetting to test
- Bad for things like
  - Making sure you test edge cases
    - If original FibonacciTest had only tested n=-1, n=1, and n=3, would have caught difference instead of sum bug, but might not have caught the edge/base case issues
  - Making sure your tests make sense
    - Good style
    - Good choice of things to test
    - Etc.

Final note

- This plugin is just a tool
  - It can’t test for you
  - It is only one way of visualizing the tests you’ve written
  - It can be misleading
- It is optional
  - If it doesn’t make your life easier, don’t use it