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

Graphs

- Node
- Edge

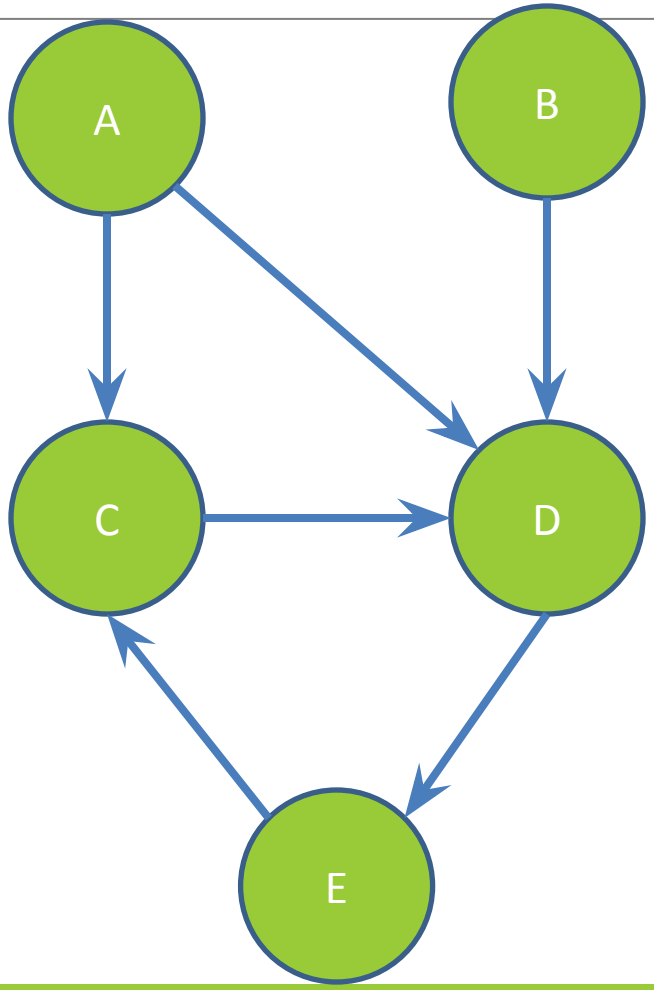
Graphs

- Node
  - data item in a graph
- Edge
  - connection between two nodes

Graphs

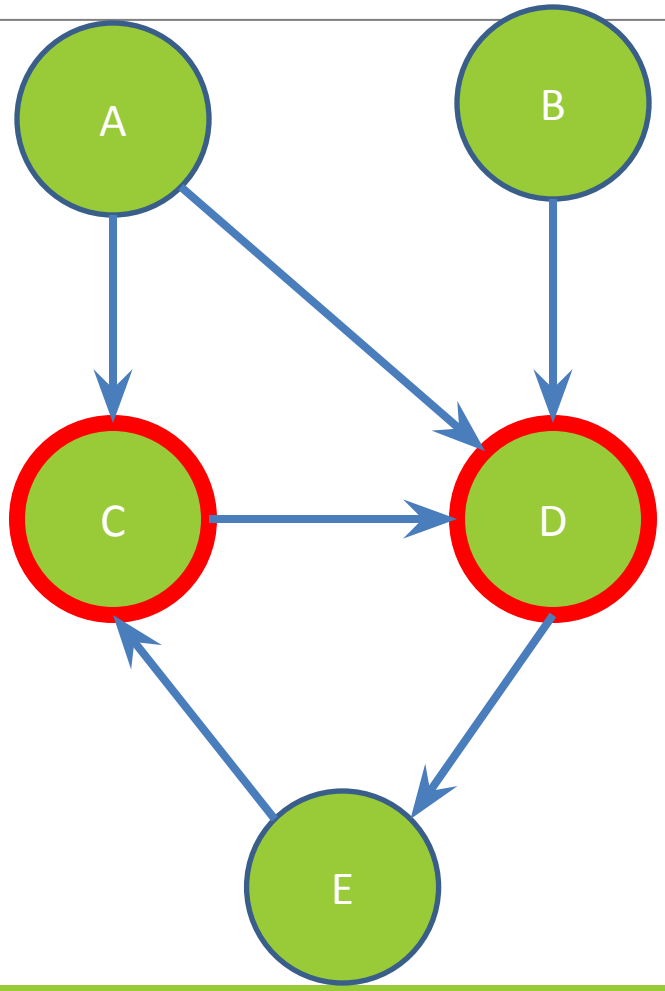
- ***Directed*** graph: edges have a *source* and *destination*
- Edges represented with arrows
- Parent/child nodes: related by an edge

Graphs



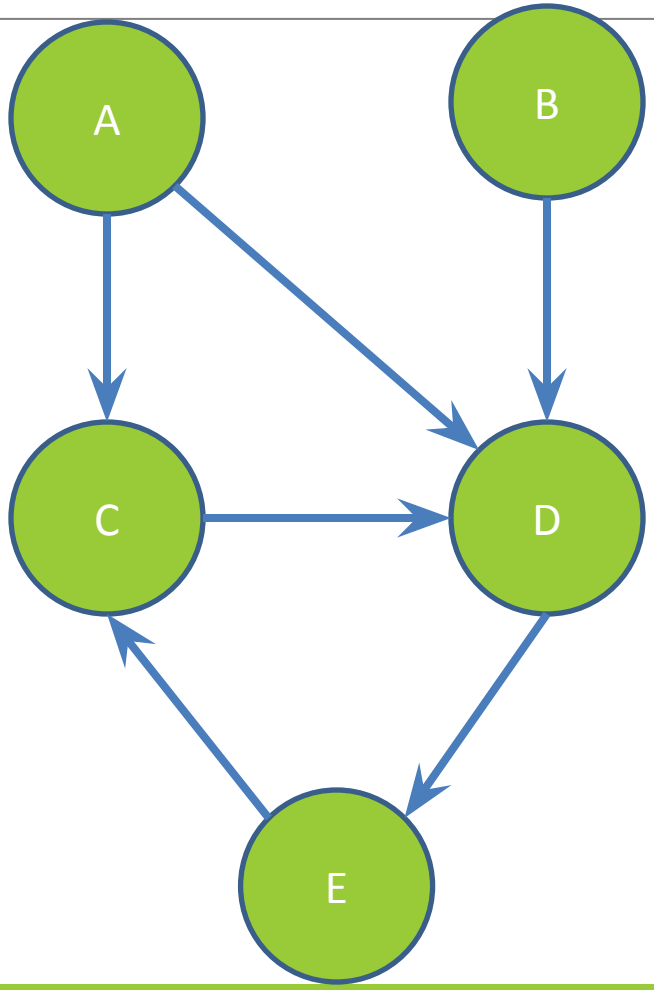
Children of A?

Graphs



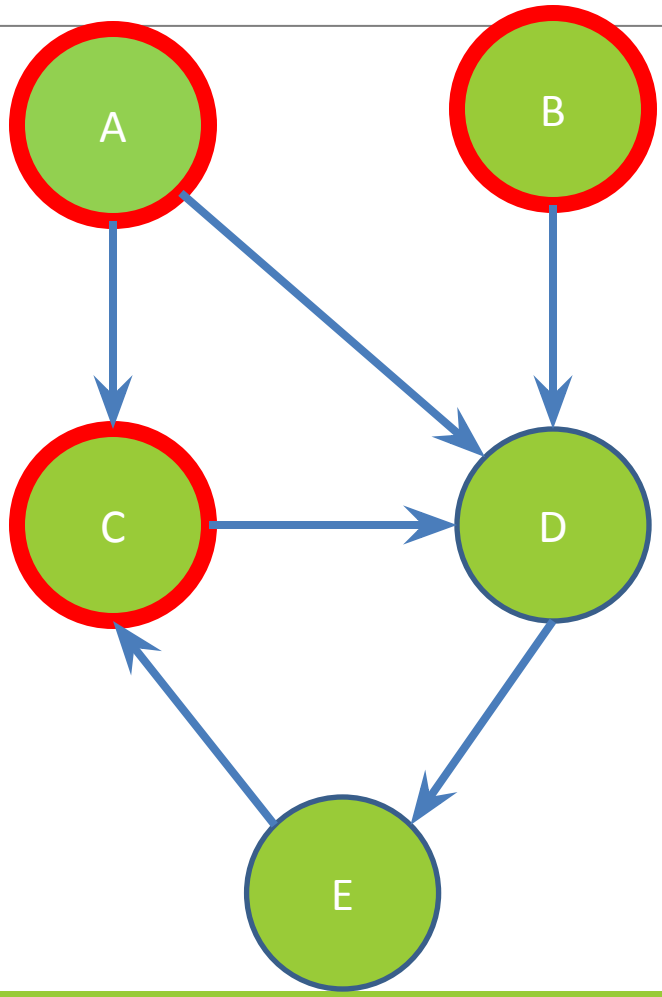
Children of A?

Graphs



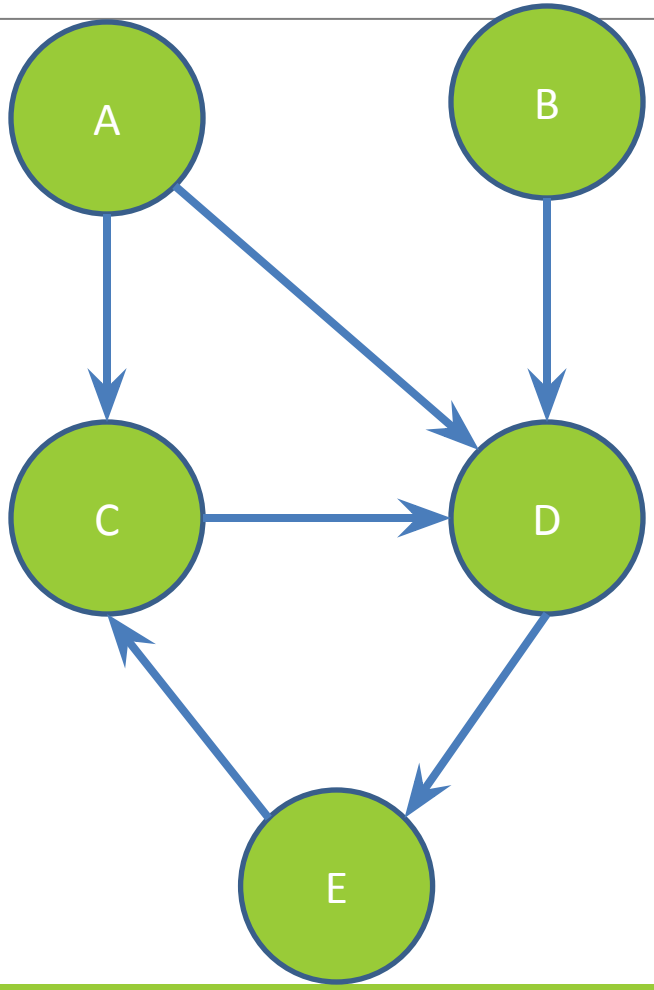
Parents of D?

Graphs



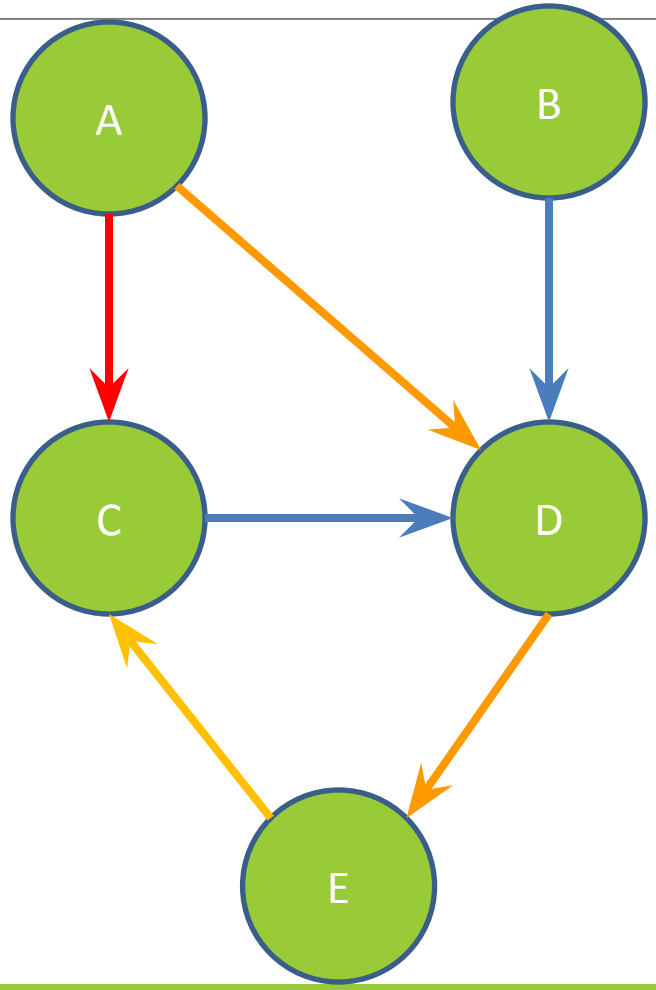
Parents of D?

Graphs



Paths from A to C:

Graphs



Paths from A to C:

A -> C

A -> D -> E -> C

Shortest path from A to C?

Before we move on...

Read the wikipedia article in the spec!

(It has implementation hints!)


LOOK AT ALL
THE BUGS WE WILL
FIX
memegenerator.net


FIX
ALL THE BUGS
DIYLOL.COM

Testing

Internal vs. external

- 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 Test1() { … }
```

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 only if specified exception is thrown
- 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);
}
```

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:

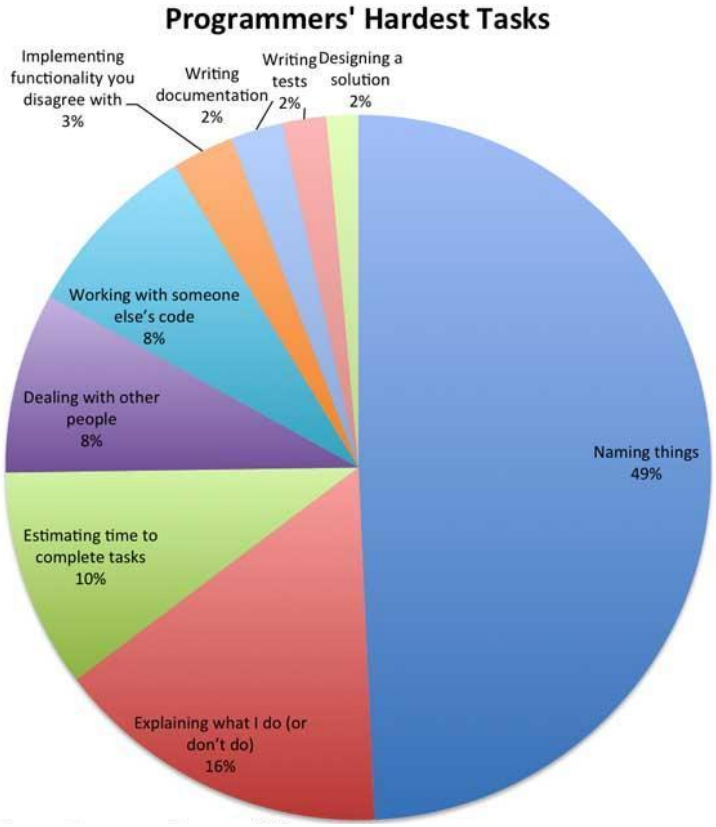
```
@BeforeClass
public static void name() { ... }

@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

Ground 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; failing assertion halts test
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

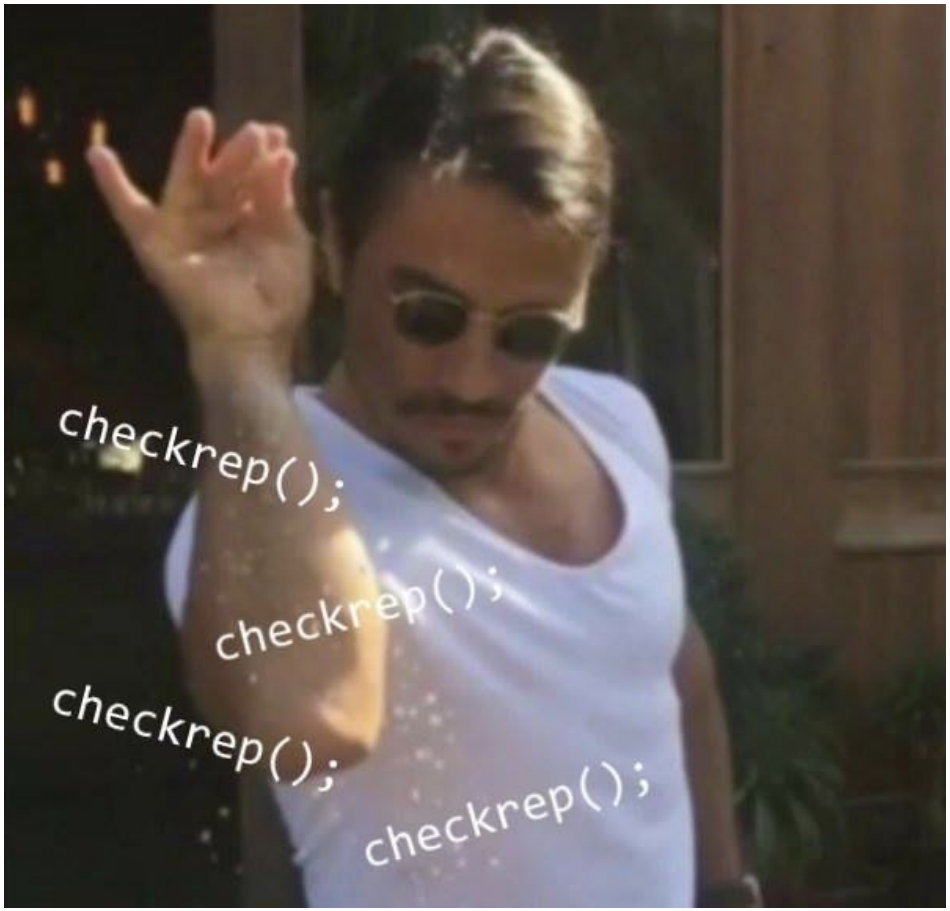
- We've just been discussing JUnit assertions so far
- Java itself has assertions

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

Reminder: Enabling asserts in Eclipse

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

Don’t forgot your CheckReps!



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

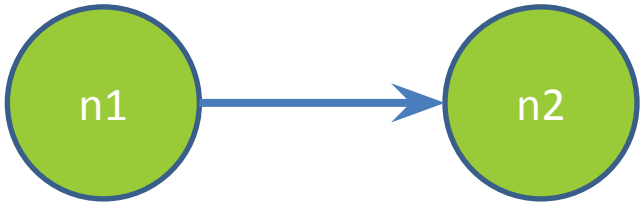
Test script language

```
# 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
```



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
- Demo steps are in spec