hw6, BFS, debugging

CSE 331

Section 5 – 10/25/12

Slides by Kellen Donohue

Agenda

- hw5 to graded in time for feedback to be used on hw6
- hw6 due next week

- Today
  - Asserts
  - hw6 data
  - BFS
  - Debugging

hashcode() and equals()

Overriding these important for using classes you write in collections, e.g.

Read Javadoc for requirements

- Transitive, symmetric, etc. we'll discuss later in lecture
- Usually must override hashcode() if you override equals()

Eclipse can generate them for you

- Right click in class source file
- Source -> Generate hashCode() and equals()
- Not always perfect – learn more later & in 332

Asserts

```
assert true;  // nothing happens

assert false;  // program terminates with an
               // assertion failure
```

Asserts

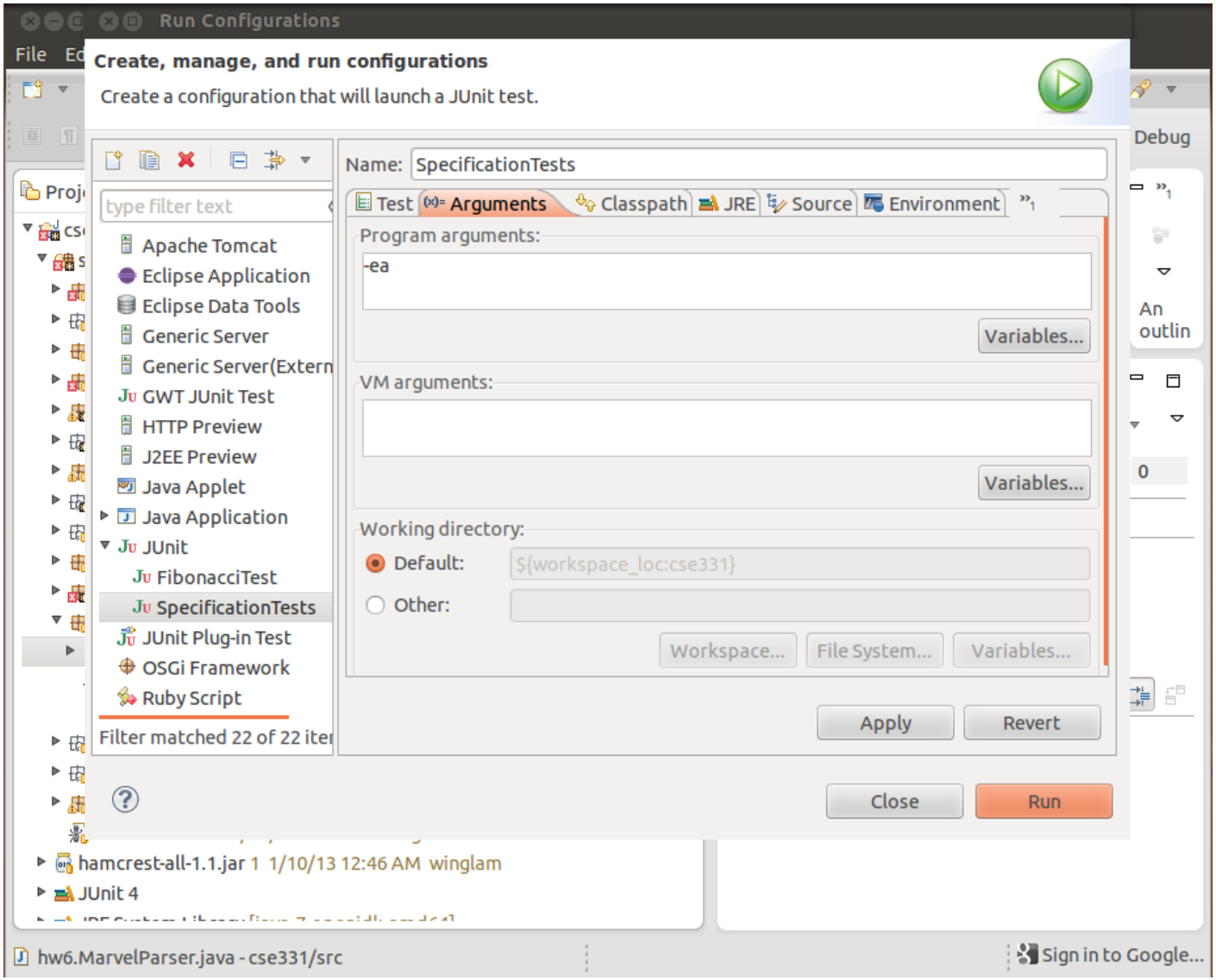
You must manually turn on assert statements for them to be run in your code.

The command line flag is "-ea"

To set command line flags in eclipse:

- The .java file you are running -> Run As -> Run Configurations
  - Arguments tab
    - Enter "-ea" under 'Program arguments'

Asserts


Run Configurations
File
Create, manage, and run configurations
Create a configuration that will launch a JUnit test.
Name: SpecificationTests
type filter text
Apache Tomcat
Eclipse Application
Eclipse Data Tools
Generic Server
Generic Server(Extern
GWT JUnit Test
HTTP Preview
J2EE Preview
Java Applet
Java Application
JUnit
FibonacciTest
SpecificationTests
JUnit Plug-in Test
OSGi Framework
Ruby Script
Filter matched 22 of 22 iter
Test
Arguments
Classpath
JRE
Source
Environment
Program arguments:
-ea
Variables...
VM arguments:
Variables...
Working directory:
Default:
${workspace_loc:cse331}
Other:
Workspace...
File System...
Variables...
Apply
Revert
Close
Run
Debug
hamcrest-all-1.1.jar 1 1/10/13 12:46 AM winglam
JUnit 4
hw6.MarvelParser.java - cse331/src
Sign in to Google...

Homework 6

Use Graph ADT from hw5

Fill with Marvel Data

- Nodes = characters
- Edges = books
  - Labeled with title
  - Connecting characters if both characters appeared in that book

Turns out to model real life social graphs

Homework 6

The Data

Download from HW6 assignment page

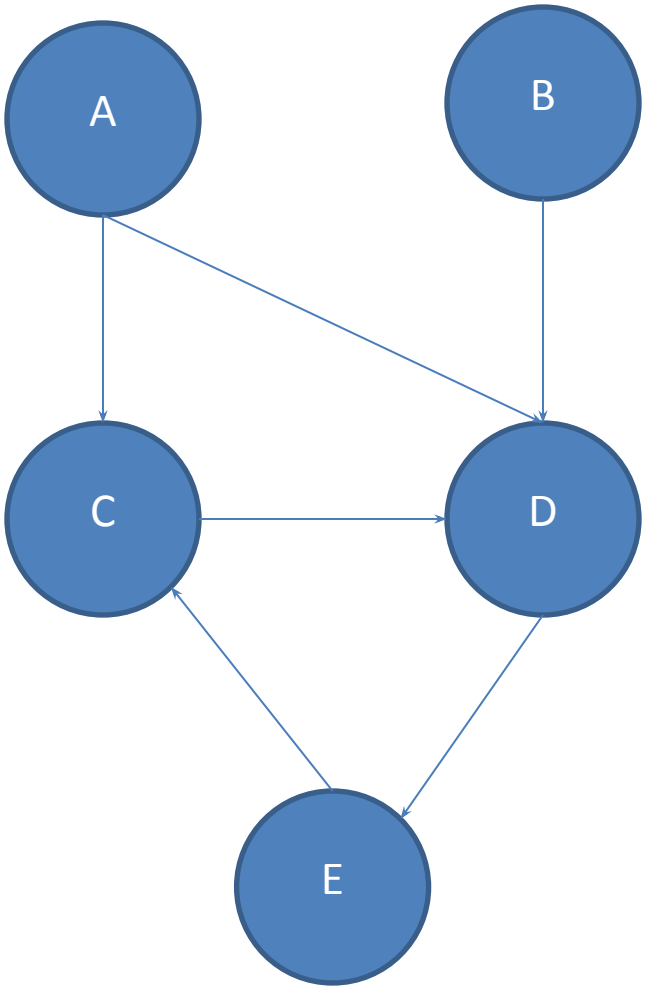
labeled_edges.tsv

```
"FROST, CARMILLA"       "AA2 35"
"KILLRAVEN/JONATHAN R"  "AA2 35"
"M'SHULLA"      "AA2 35"
"24-HOUR MAN/EMMANUEL"  "AA2 35"
"OLD SKULL"     "AA2 35"
"G'RATH"        "AA2 35"
"3-D MAN/CHARLES CHAN"  "M/PRM 35"
"3-D MAN/CHARLES CHAN"  "M/PRM 36"
"3-D MAN/CHARLES CHAN"  "M/PRM 37"
"HUMAN ROBOT"   "WI? 9"
"MARVEL BOY III/ROBER"  "WI? 9"
"GORILLA-MAN"   "WI? 9"
"3-D MAN/CHARLES CHAN"  "WI? 9"
"VENUS II"      "WI? 9"
"HUMAN ROBOT"   "AVF 4"
"GORILLA-MAN"   "AVF 4"
"JONES, RICHARD MILHO"  "AVF 4"
"3-D MAN/CHARLES CHAN"  "AVF 4"
"WASP/JANET VAN DYNE "  "AVF 4"
"LIBRA/GUSTAV BRANDT"   "AVF 4"
"CAPTAIN AMERICA"       "AVF 4"
"VENUS II"      "AVF 4"
```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 INS

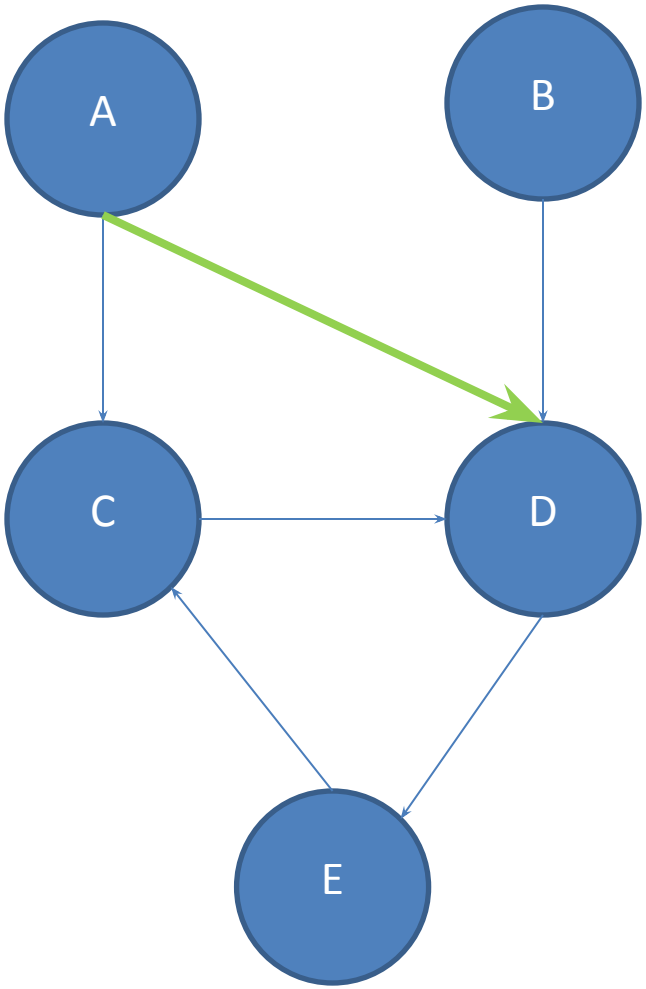
Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction



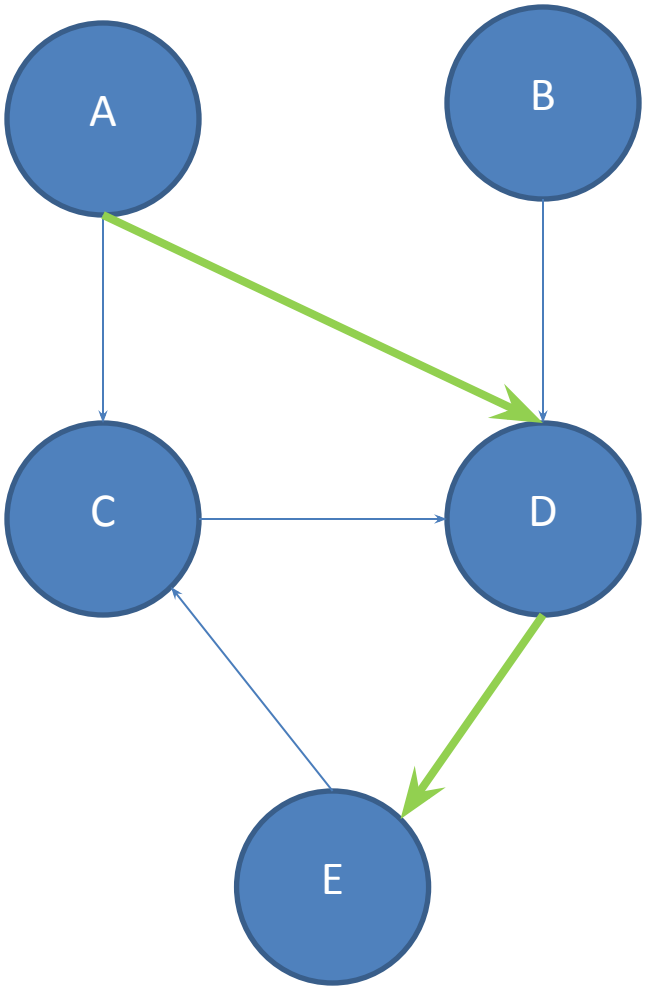
Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction



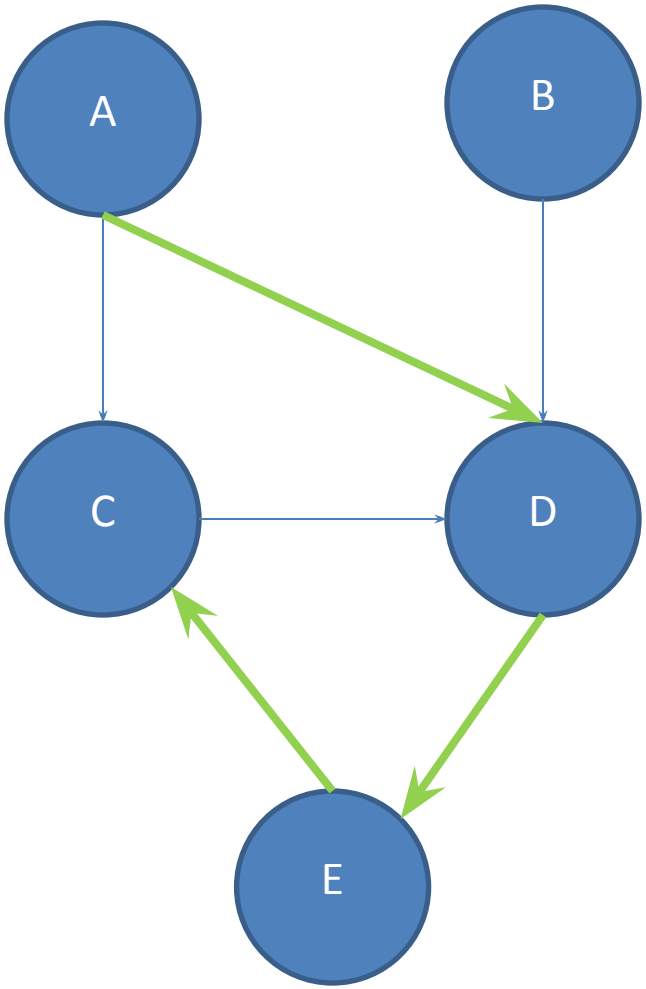
Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction



Graph paths

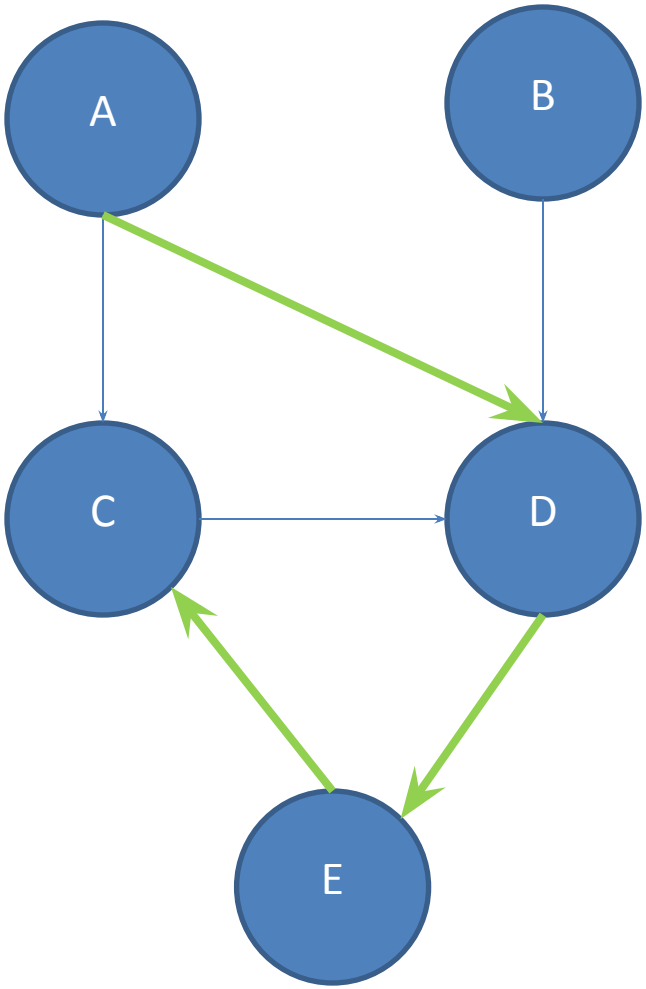
- List of nodes travelled to get from one node to another, moving along edges, respecting direction



Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction

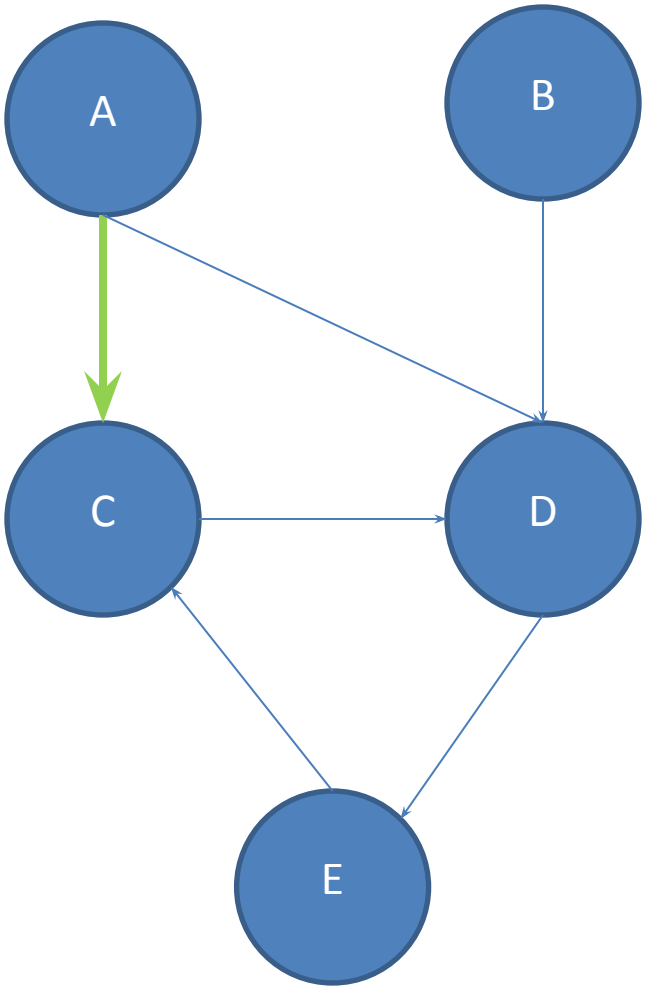
- ADEC is a path A to C



Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction

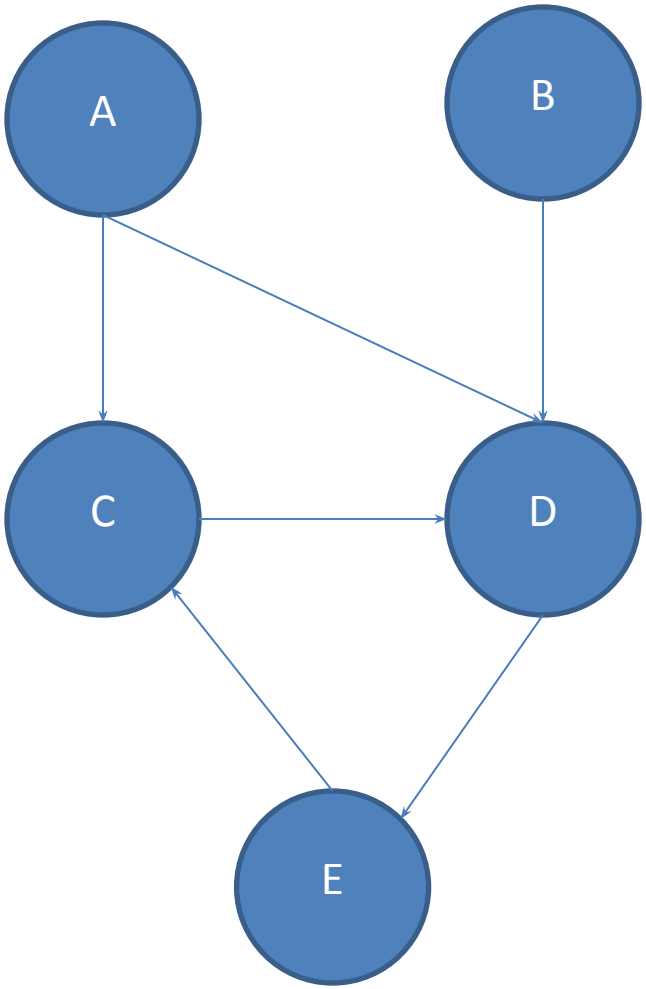
- ADEC is a path A to C
- AC is a path A to C



Graph paths

- List of nodes travelled to get from one node to another, moving along edges, respecting direction

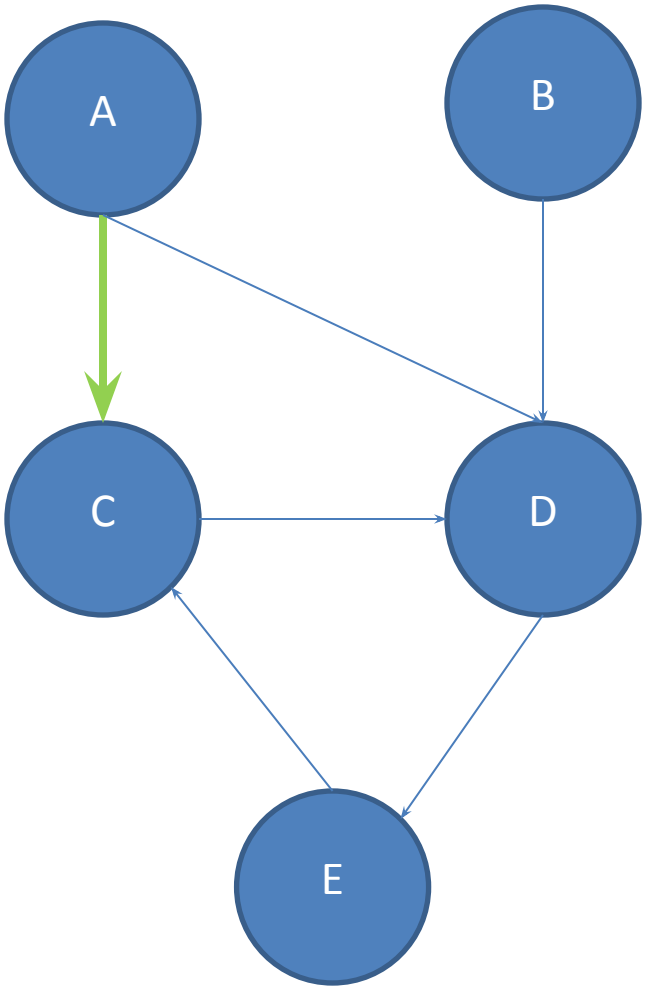
- ADEC is a path A to C
- AC is a path A to C
- There's no path A to B



Graph paths

- We often want to find the <u>shortest</u> path between two nodes
  - Google Maps
  - Optimal route through a maze

- AC is the shortest path A to C



Breadth-first search

Pseudo code

```
Put start node in a queue
While the queue isn't empty
    Pick a node N off the queue
    If N is the goal then return
    Else, for each node O you can reach from N
        If O isn't marked
            Add O to the queue
            Mark O
// Couldn't find a path from start node to goal node
Return false
```

Breadth-first search

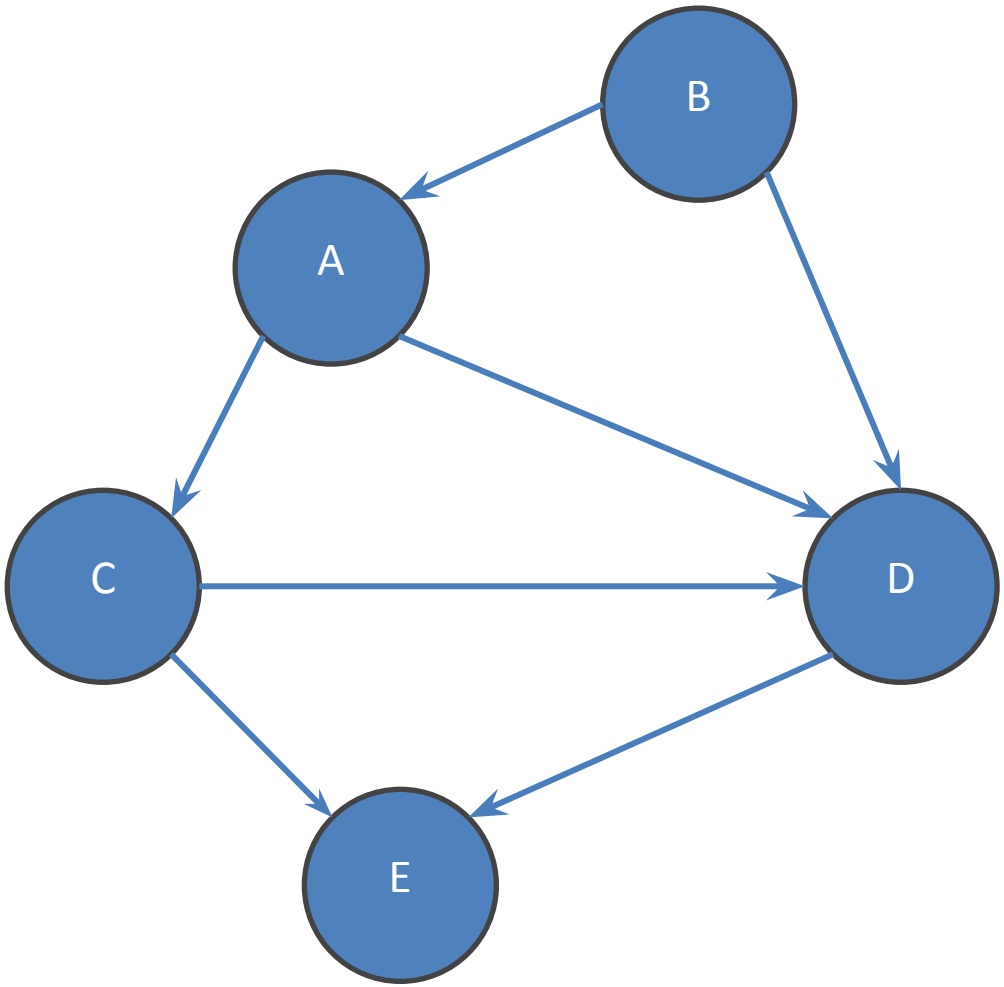
We often want to find the <u>shortest</u> path between two nodes

- Google Maps
- Optimal route through a maze

Breadth-first search

Queue

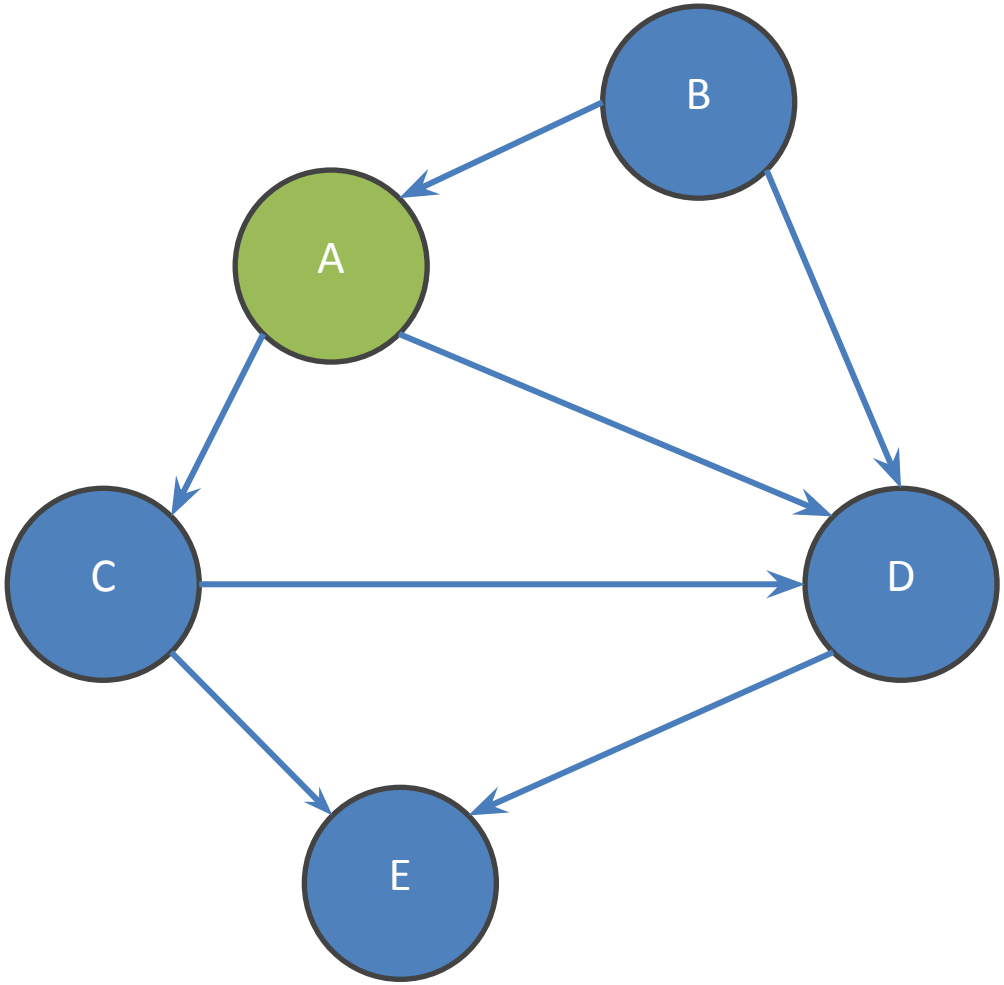
< <



Breadth-first search

Queue

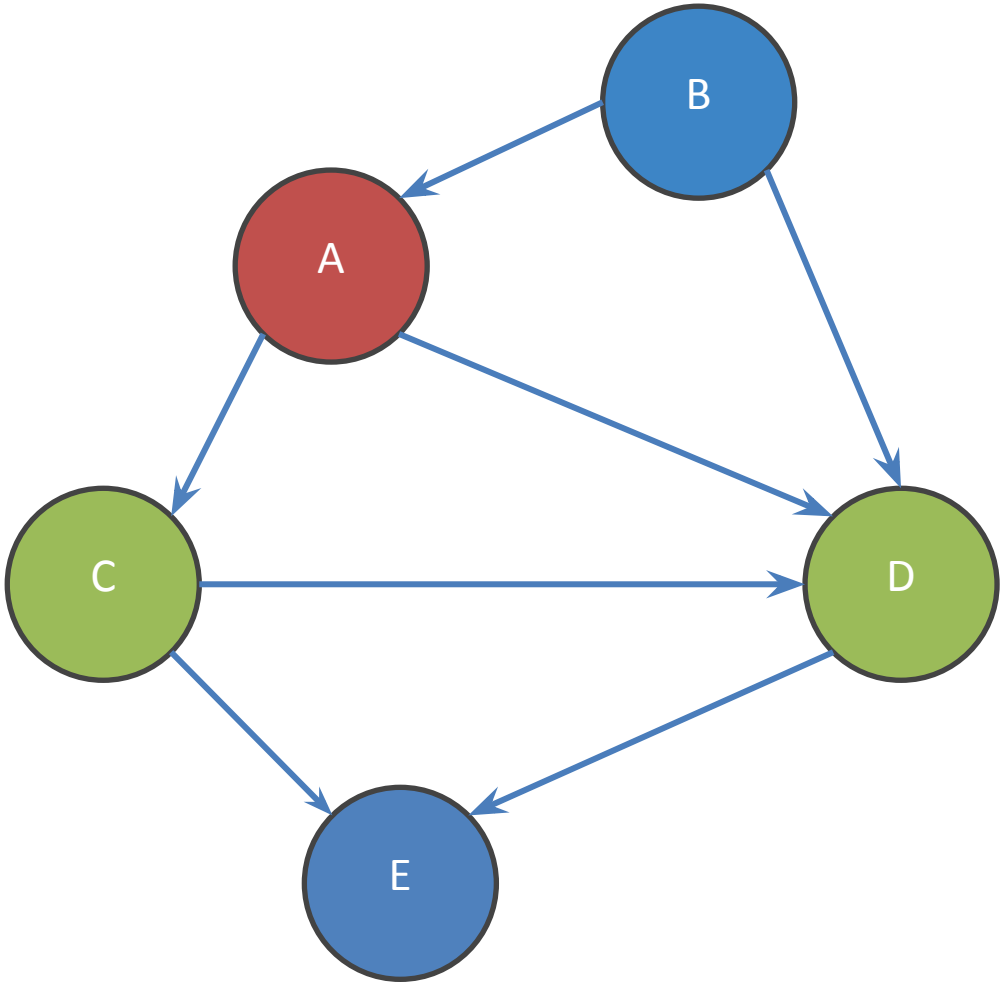
< A <



Breadth-first search

Queue

< C D <



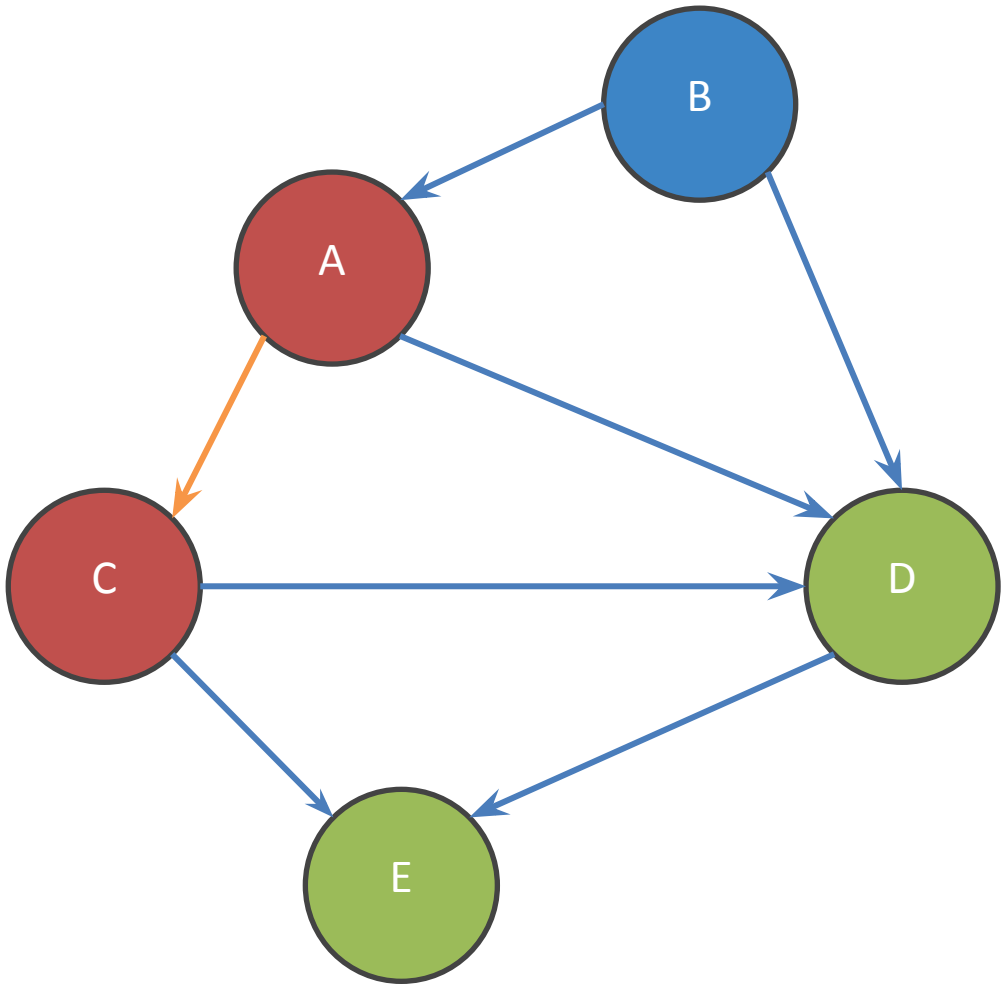
Breadth-first search

Queue

< D E <

Reach C by path

A -> C



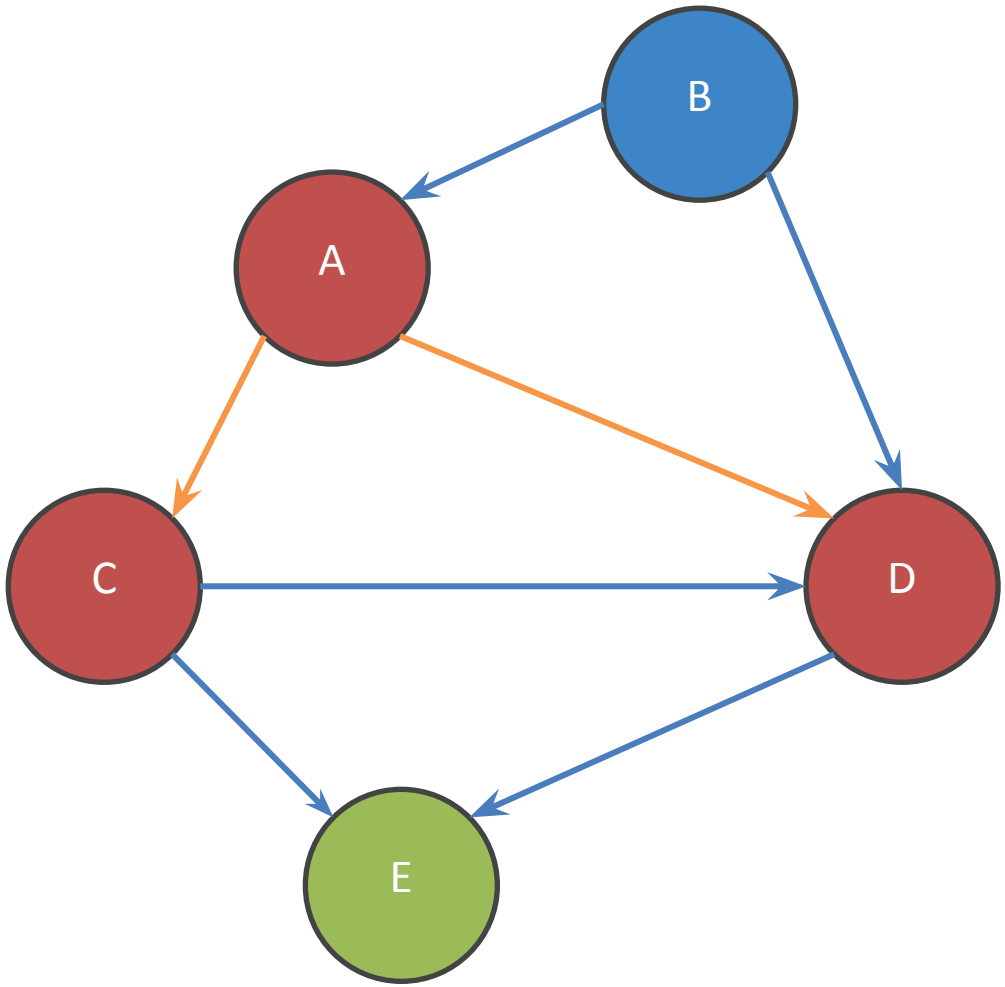
Breadth-first search

Queue

< E <

Reach D by path

A -> D



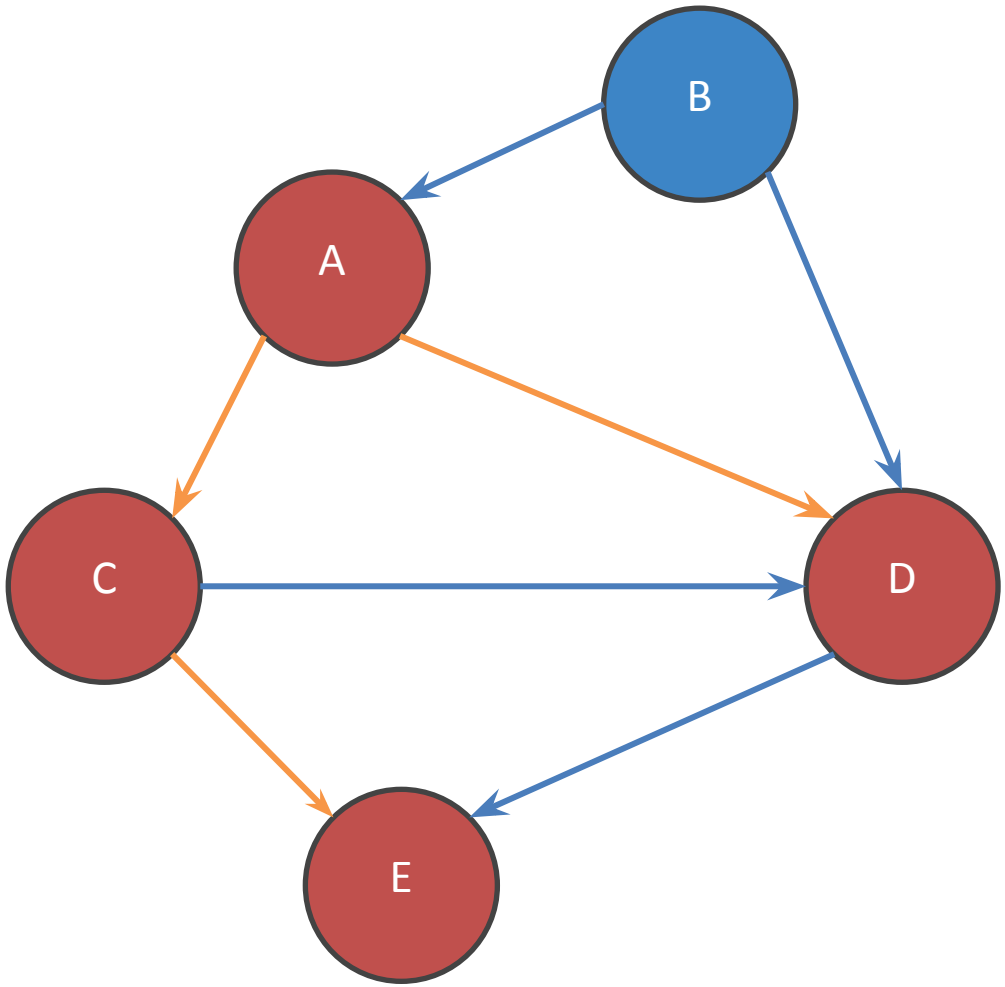
Breadth-first search

Queue

< <

Reach E by path

A -> C -> E

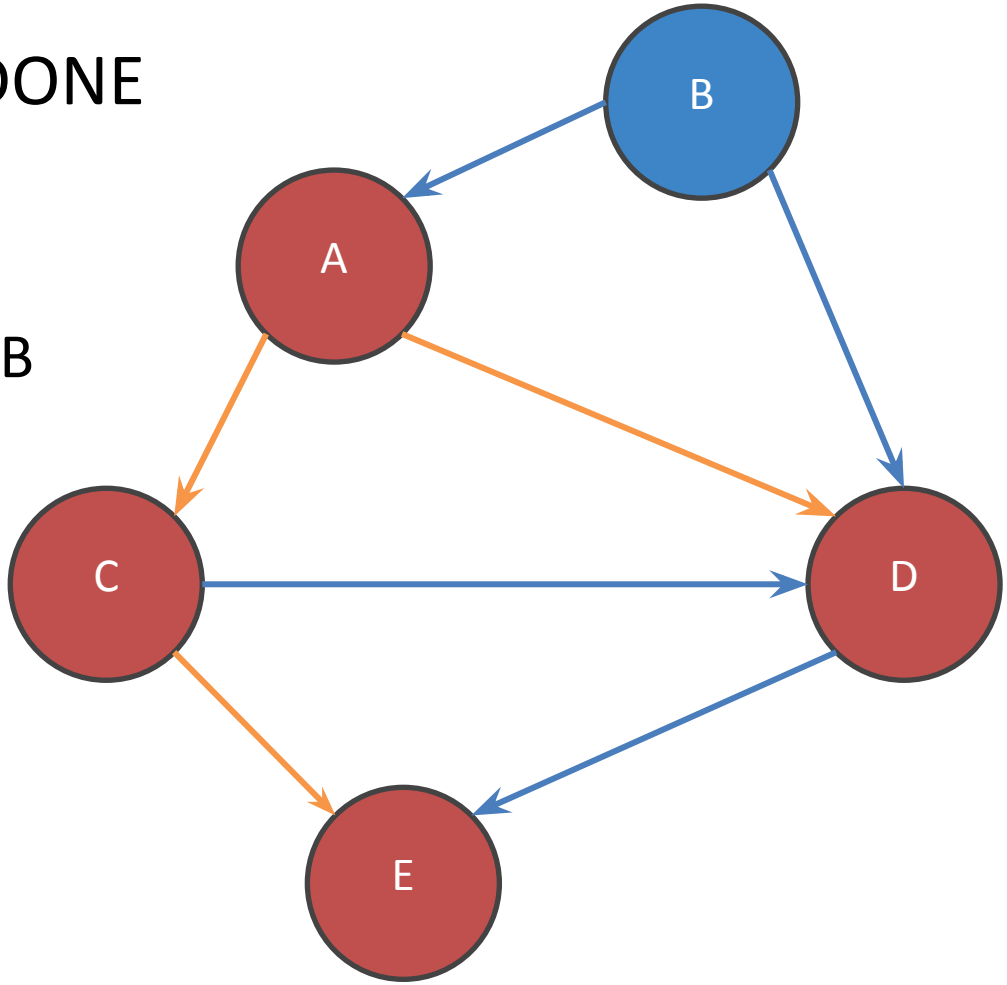


Breadth-first search

Queue empty: DONE

< <

No path from A to B



Breadth-first search

Guaranteed to find shortest-path

- In number of nodes
- Not lowest cost path if edges have cost

Breadth First Search uses a Queue. Change to a Stack to change it to a Depth First Search

Very memory intensive for large graphs -- O(b^d)

Will use in HW6 to find shortest paths between two characters

Eclipse Debugging

Eclipse has a great debugger!

- Complicated, hidden features
- I'll demo, but don't feel try to remember how to do everything – slides will be posted

Eclipse Debugging

Quick Access | Java | Debug | SVN Repository Exploring | PyDev

Debug

- DelegatingMethodAccessorImpl.invoke(Object, Object[]) lir
- Method.invoke(Object, Object...) line: not available
- FrameworkMethod$1.runReflectiveCall() line: 45
- FrameworkMethod$1(ReflectiveCallable).run() line: 15
- FrameworkMethod.invokeExplosively(Object, Object...) line:
- InvokeMethod.evaluate() line: 20
- BlockJUnit4ClassRunner(ParentRunner<T>).runLeaf(Statem
- BlockJUnit4ClassRunner.runChild(FrameworkMethod, RunN
- BlockJUnit4ClassRunner.runChild(Object, RunNotifier) line:
- ParentRunner$3.run() line: 231
- ParentRunner$1.schedule(Runnable) line: 60
- BlockJUnit4ClassRunner(ParentRunner<T>).runChildren(Ru
- ParentRunner<T>.access$000(ParentRunner, RunNotifier) li

Variables | Breakpoints | Expressions

Name	Value
this	RatPolyStackTest (id=33)

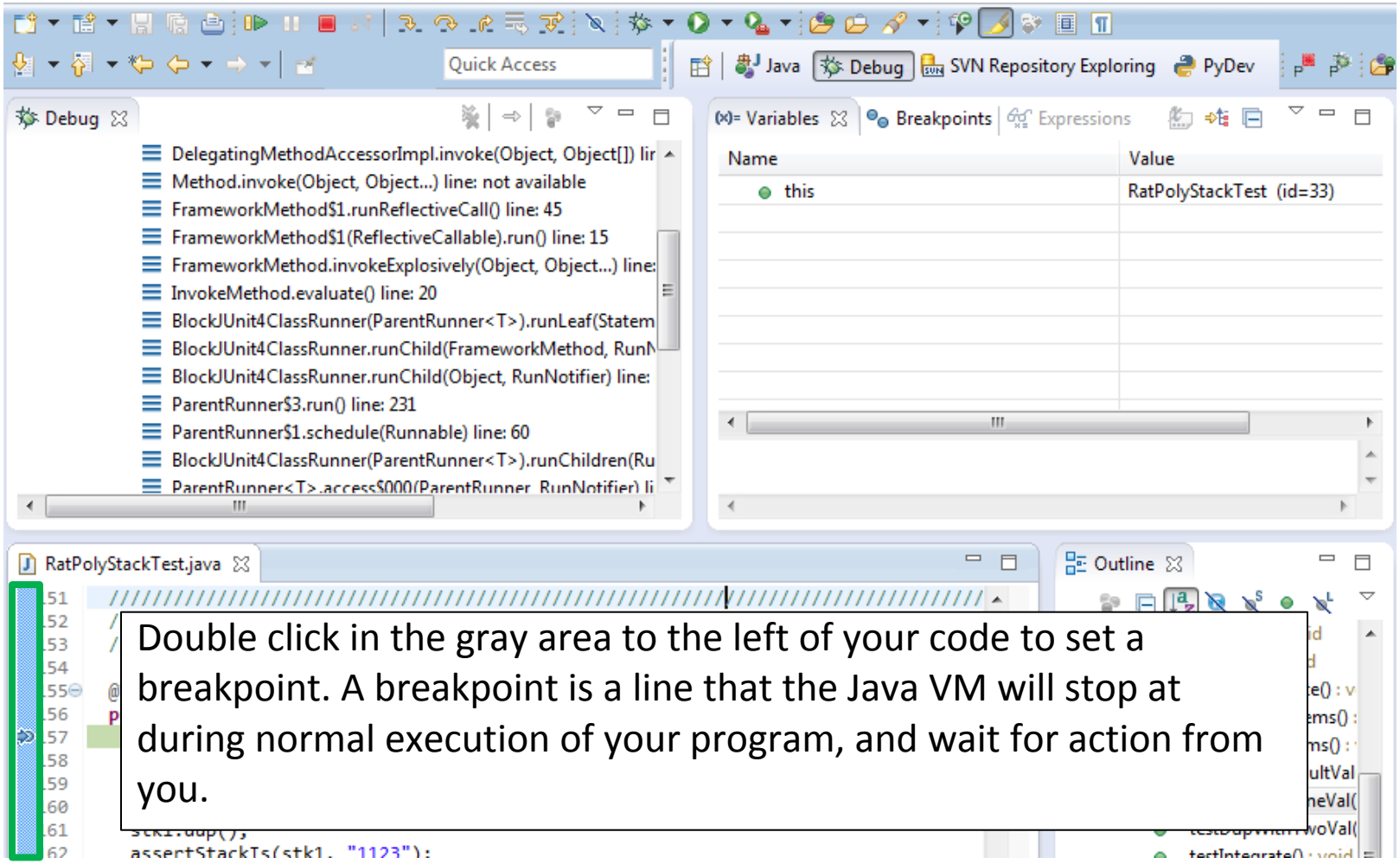
RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

Outline

- testClear() : void
- testCtor() : void
- testDifferentiate() : v
- testDivMultiElems() :
- testDivTwoElems() :
- testDupWithMultVal
- testDupWithOneVal(
- testDupWithTwoVal(
- testIntegrate() : void

Eclipse Debugging



Double click in the gray area to the left of your code to set a breakpoint. A breakpoint is a line that the Java VM will stop at during normal execution of your program, and wait for action from you.

Eclipse Debugging

Click the Bug icon to run in Debug mode. Otherwise your program won't stop at your breakpoints.

Debug

DelegatingMethodAc
Method.invoke(Obje
FrameworkMethod$1
FrameworkMethod$1(ReflectiveCallable).run() line: 15
FrameworkMethod.invokeExplosively(Object, Object...) line:
InvokeMethod.evaluate() line: 20
BlockJUnit4ClassRunner(ParentRunner<T>).runLeaf(Statem
BlockJUnit4ClassRunner.runChild(FrameworkMethod, RunN
BlockJUnit4ClassRunner.runChild(Object, RunNotifier) line:
ParentRunner$3.run() line: 231
ParentRunner$1.schedule(Runnable) line: 60
BlockJUnit4ClassRunner(ParentRunner<T>).runChildren(Ru
ParentRunner<T>.access$000(ParentRunner, RunNotifier) li

Repository Exploring PyDev

Expressions

Value
RatPolyStackTest (id=33)

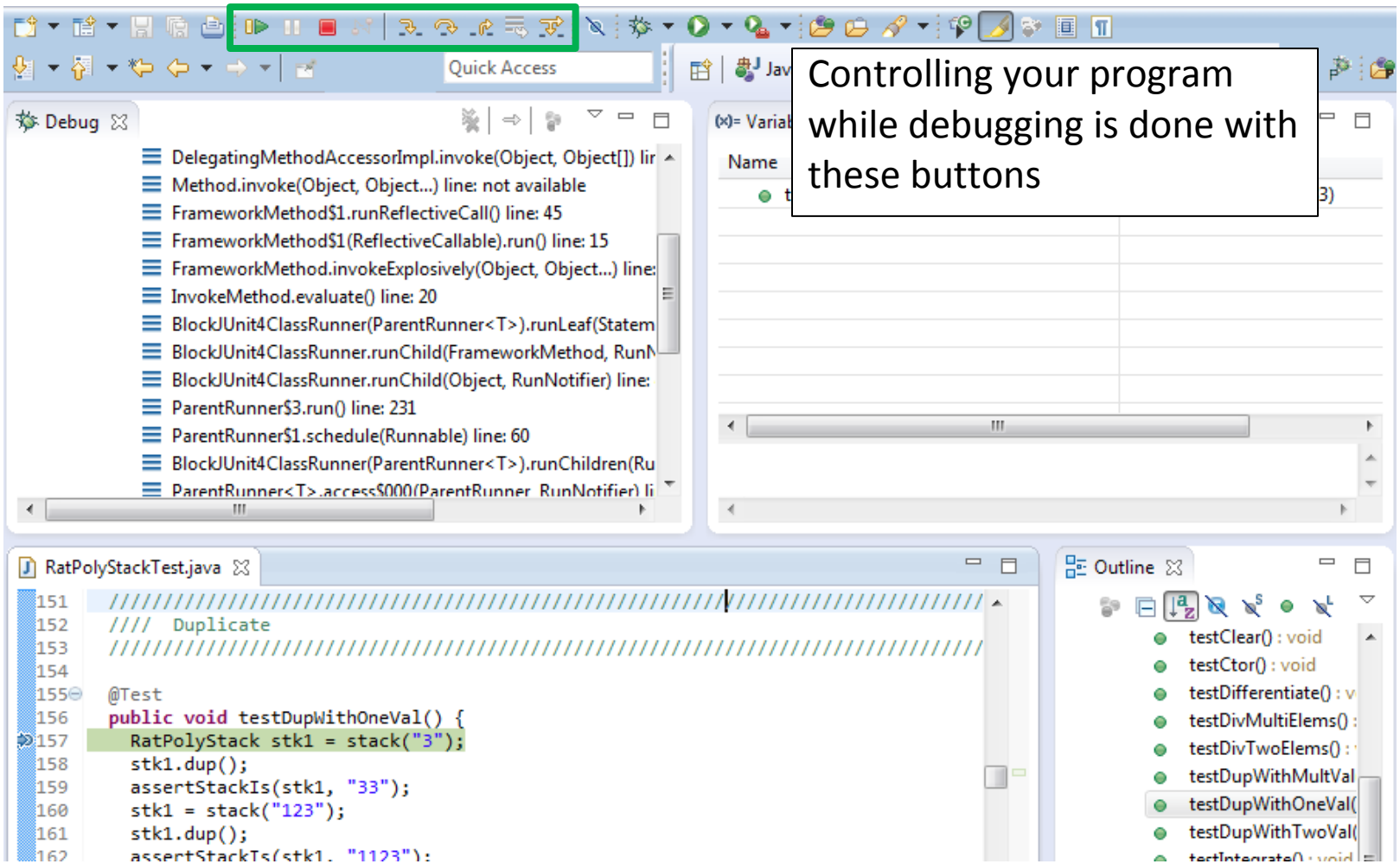
RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

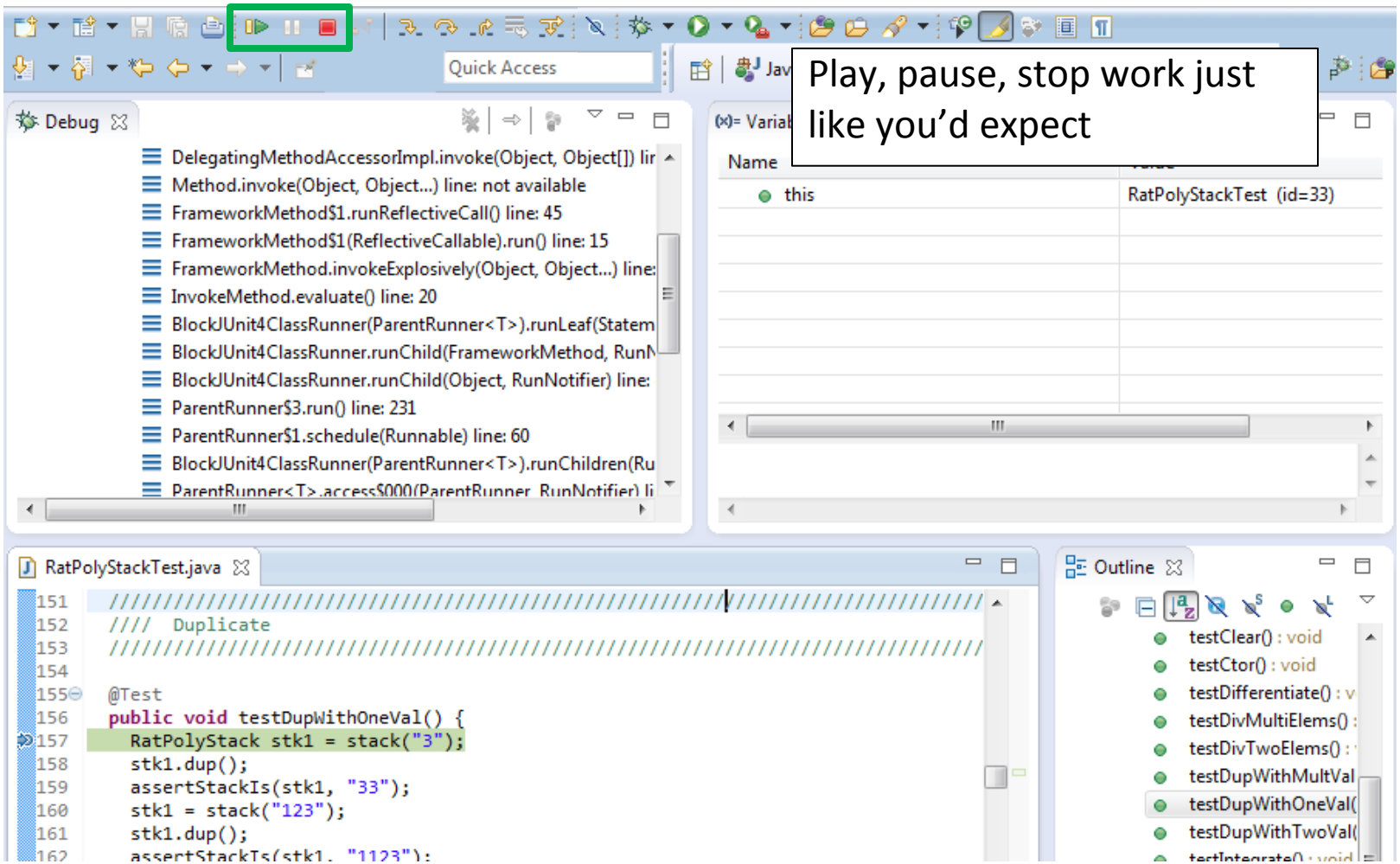
Outline

- testClear() : void
- testCtor() : void
- testDifferentiate() : v
- testDivMultiElems() :
- testDivTwoElems() :
- testDupWithMultVal
- testDupWithOneVal(
- testDupWithTwoVal(
- testIntegrate() : void

Eclipse Debugging

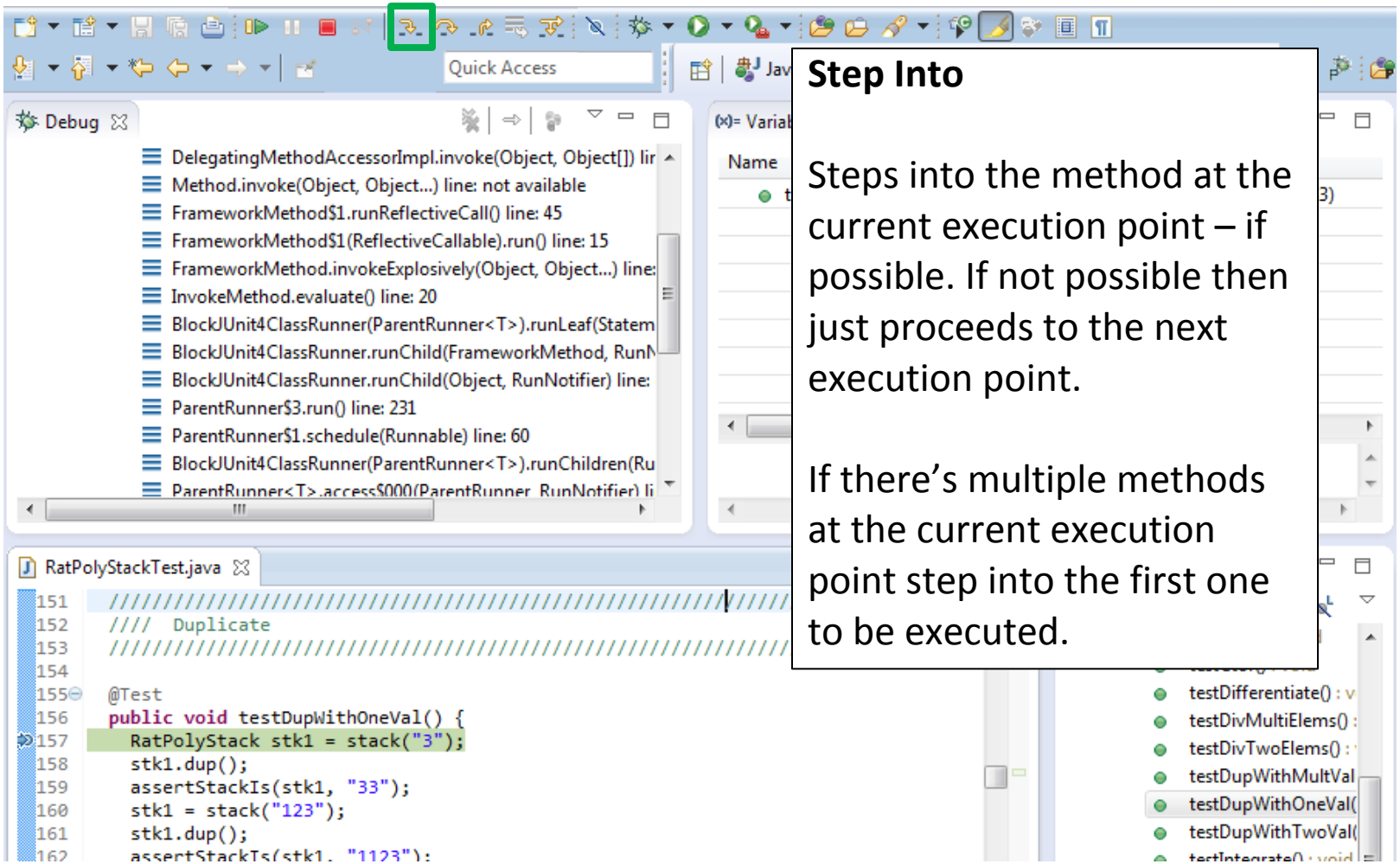


Eclipse Debugging



Play, pause, stop work just like you'd expect

Eclipse Debugging

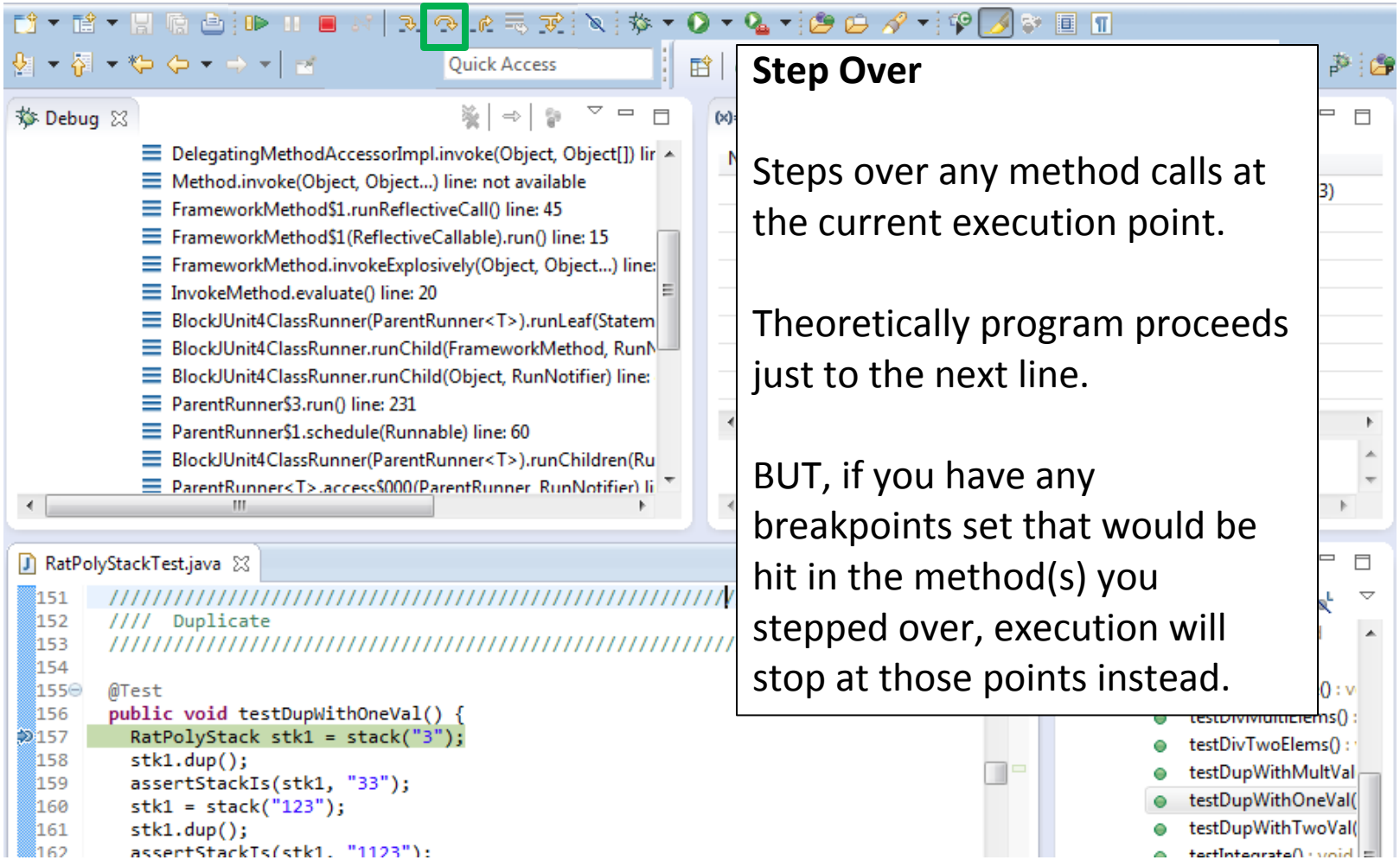


Step Into

Steps into the method at the current execution point – if possible. If not possible then just proceeds to the next execution point.

If there's multiple methods at the current execution point step into the first one to be executed.

Eclipse Debugging



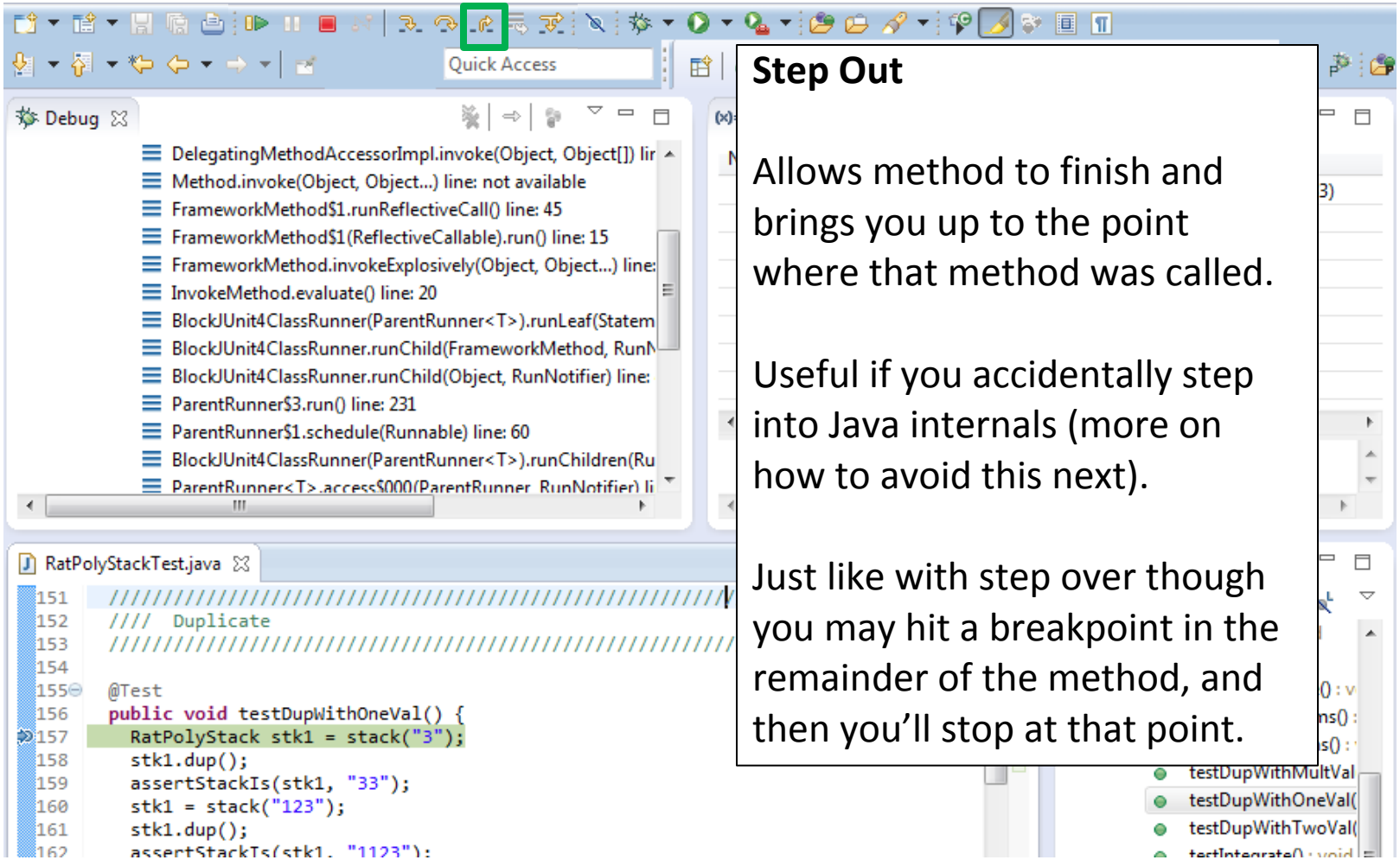
Step Over

Steps over any method calls at the current execution point.

Theoretically program proceeds just to the next line.

BUT, if you have any breakpoints set that would be hit in the method(s) you stepped over, execution will stop at those points instead.

Eclipse Debugging



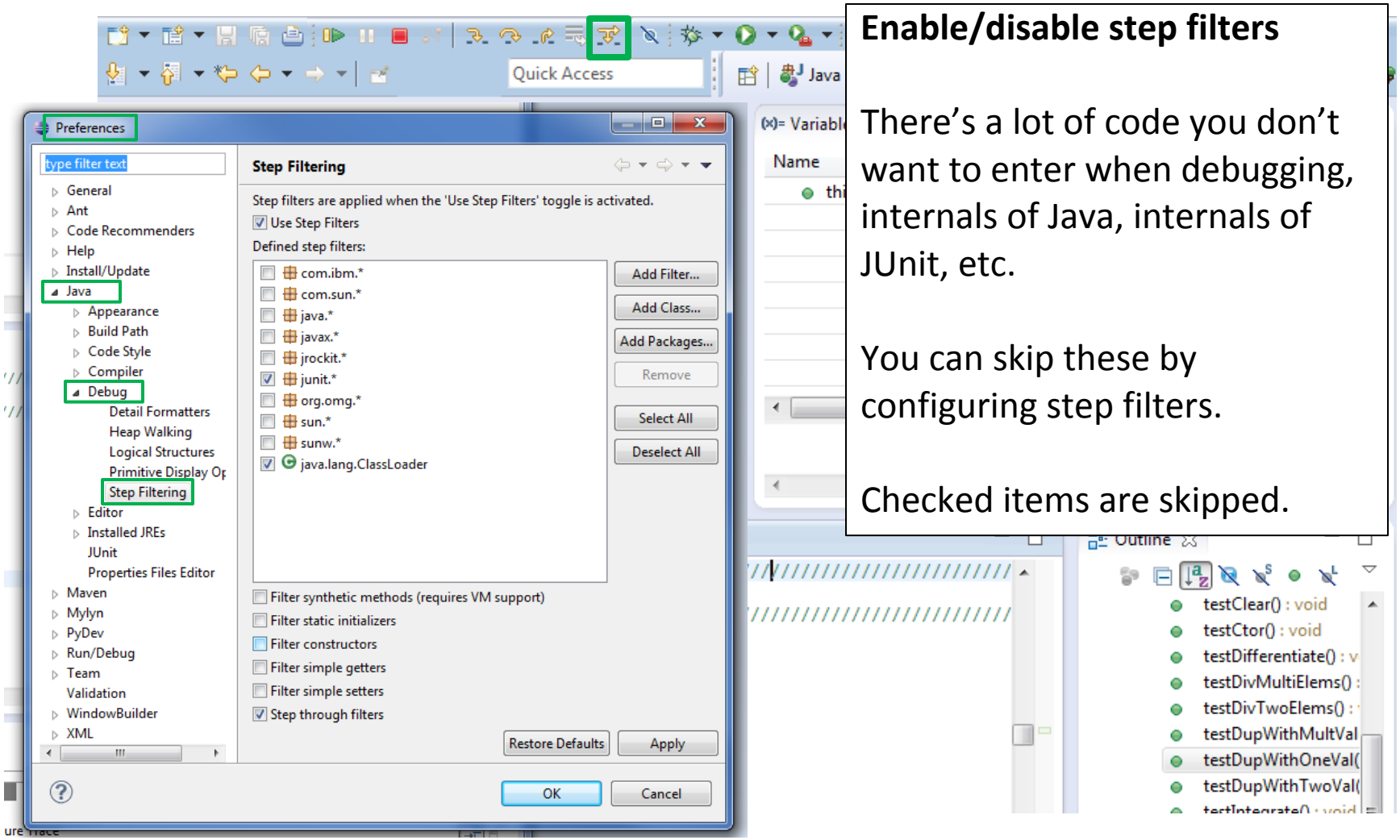
Step Out

Allows method to finish and brings you up to the point where that method was called.

Useful if you accidentally step into Java internals (more on how to avoid this next).

Just like with step over though you may hit a breakpoint in the remainder of the method, and then you'll stop at that point.

Eclipse Debugging



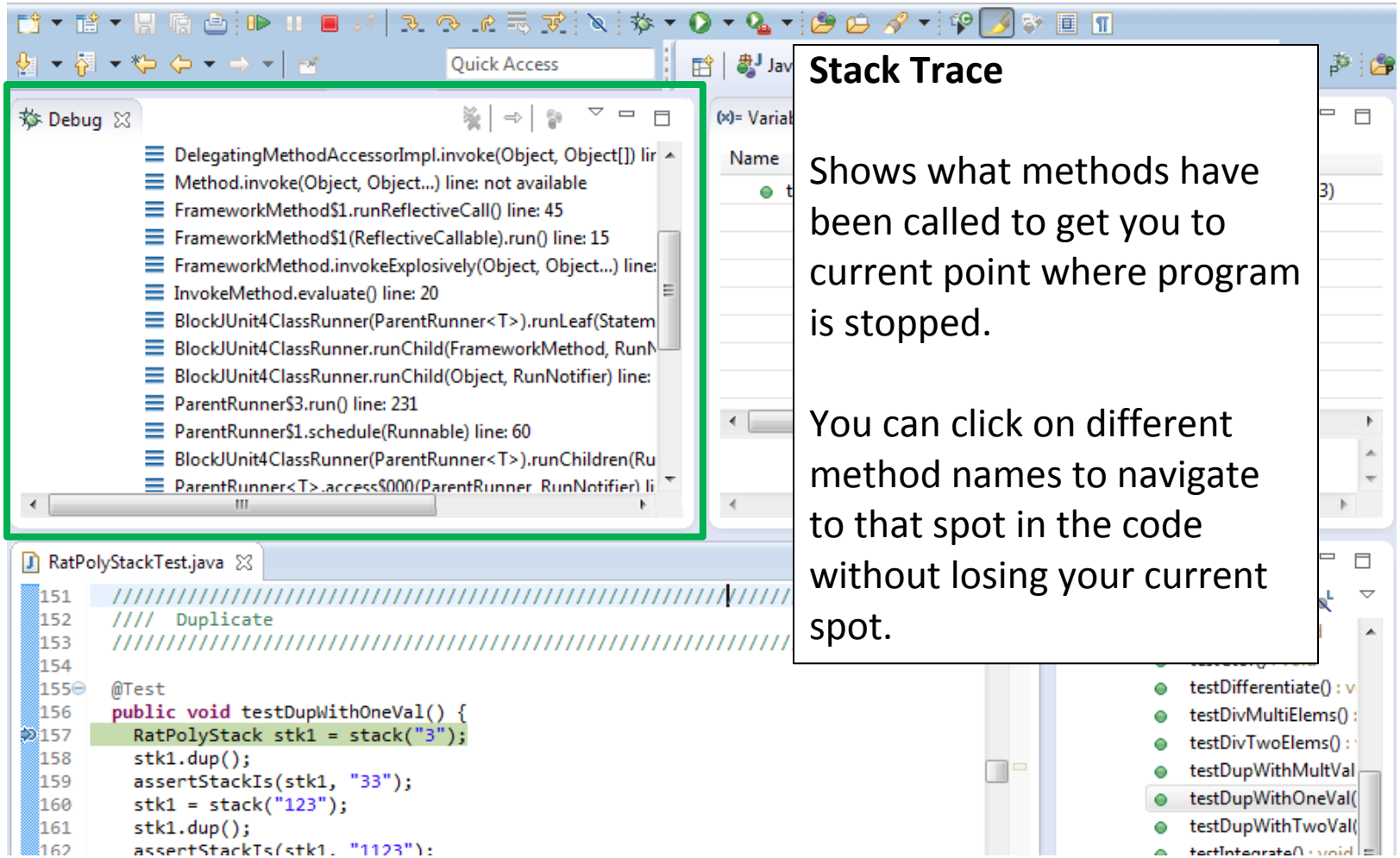
Enable/disable step filters

There's a lot of code you don't want to enter when debugging, internals of Java, internals of JUnit, etc.

You can skip these by configuring step filters.

Checked items are skipped.

Eclipse Debugging



Stack Trace

Shows what methods have been called to get you to current point where program is stopped.

You can click on different method names to navigate to that spot in the code without losing your current spot.

Eclipse Debugging

Variables Window

Shows all variables, including method parameters, local variables, and class variables, that are in scope at the current execution spot. Updates when you change positions in the stackframe. You can expand objects to see child member values. There's a simple value printed, but clicking on an item will fill the box below the list with a pretty format.

Some values are in the form of ObjectName (id=x), this can be used to tell if two variables are reffering to the same object.

Eclipse Debugging

Variables that have changed since the last break point are highlighted in yellow.

You can change variables right from this window by double clicking the row entry in the Value tab.

Java | Debug | SVN Repository Exploring | PyDev

(x)= Variables | Breakpoints | Expressions

Name	Value
this	RatTermTest (
t	RatTerm (id=4
coeff	RatNum (id=4
expt	5

-2*x^5

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

Outline

- testClear() : void
- testCtor() : void
- testDifferentiate() : v
- testDivMultiElems() :
- testDivTwoElems() :
- testDupWithMultVal
- testDupWithOneVal(
- testDupWithTwoVal(
- testIntegrate() : void

Eclipse Debugging

Variables that have changed since the last break point are highlighted in yellow.

You can change variables right from this window by double clicking the row entry in the Value tab.

Java | Debug | SVN Repository Exploring | PyDev

(x)= Variables | Breakpoints | Expressions

Name	Value
this	RatTermTest (
t	RatTerm (id=4
coeff	RatNum (id=4
expt	5

-2*x^5

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

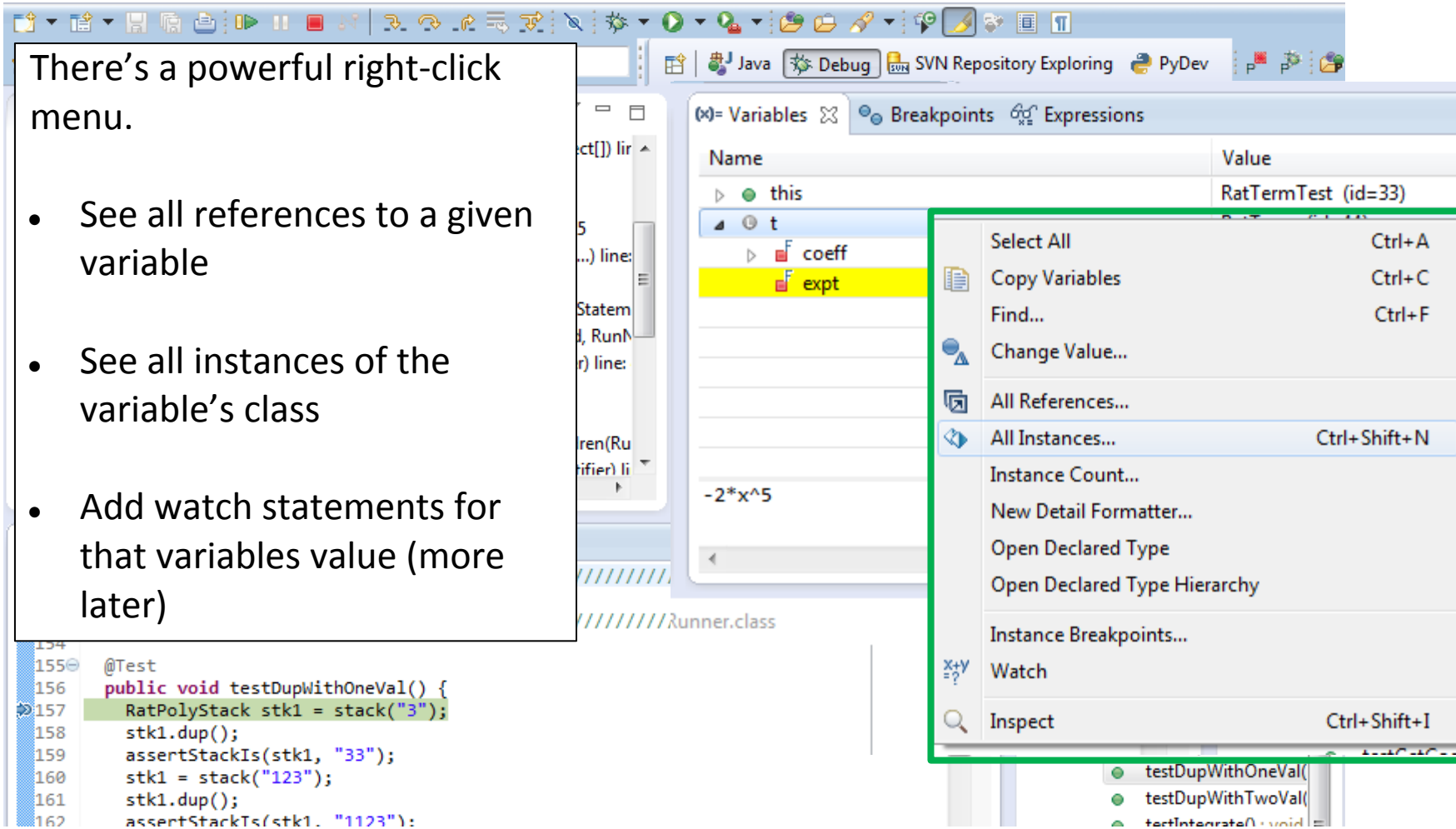
Outline

- testClear() : void
- testCtor() : void
- testDifferentiate() : v
- testDivMultiElems() :
- testDivTwoElems() :
- testDupWithMultVal
- testDupWithOneVal(
- testDupWithTwoVal(
- testIntegrate() : void

Eclipse Debugging

There's a powerful right-click menu.

- See all references to a given variable
- See all instances of the variable's class
- Add watch statements for that variables value (more later)



Eclipse Debugging

Show Logical Structure

Expands out list items so it's as if each list item were a field (and continues down for any children list items)

Java | Debug | SVN Repository Exploring | PyDev

Variables | Breakpoints | Expressions

Name	Value
this	RatPolyStackTest (id=33)
stk1	RatPolyStack (id=44)
polys	Stack<E> (id=49)
[0]	RatPoly (id=719)
terms	ArrayList<E> (id=728)
[0]	RatTerm (id=731)
coeff	RatNum (id=733)
expt	0

BlockJUnit4ClassRunner.runChild(Object, RunNotifier) line:
ParentRunner$3.run() line: 231
ParentRunner$1.schedule(Runnable) line: 60
BlockJUnit4ClassRunner(ParentRunner<T>).runChildren(Ru

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

testClear() : void
testCtor() : void
testDifferentiate() : v
testDivMultiElems() :
testDivTwoElems() :
testDupWithMultVal
testDupWithOneVal(
testDupWithTwoVal(

Eclipse Debugging

Breakpoints Window

Shows all existing breakpoints in the code, along with their conditions and a variety of options.

Double clicking a breakpoint will take you to its spot in the code.

Java | Debug | SVN Repository Exploring | PyDev

Variables | Breakpoints | Expressions

- Ones [line: 33] - main(String[])
- ProjectEuler26 [line: 25] - main(String[])
- RatPolyStackTest [line: 157] - testDupWithOneVal()
- RatPolyStackTest [line: 159] [conditional] - testDupWithOneVal()
- RatPolyStackTest [line: 162] - testDupWithOneVal()

Hit count: | Suspend thread | Suspend VM

Conditional | Suspend when 'true' | Suspend when value changes

<Choose a previously entered condition>

x == 6

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

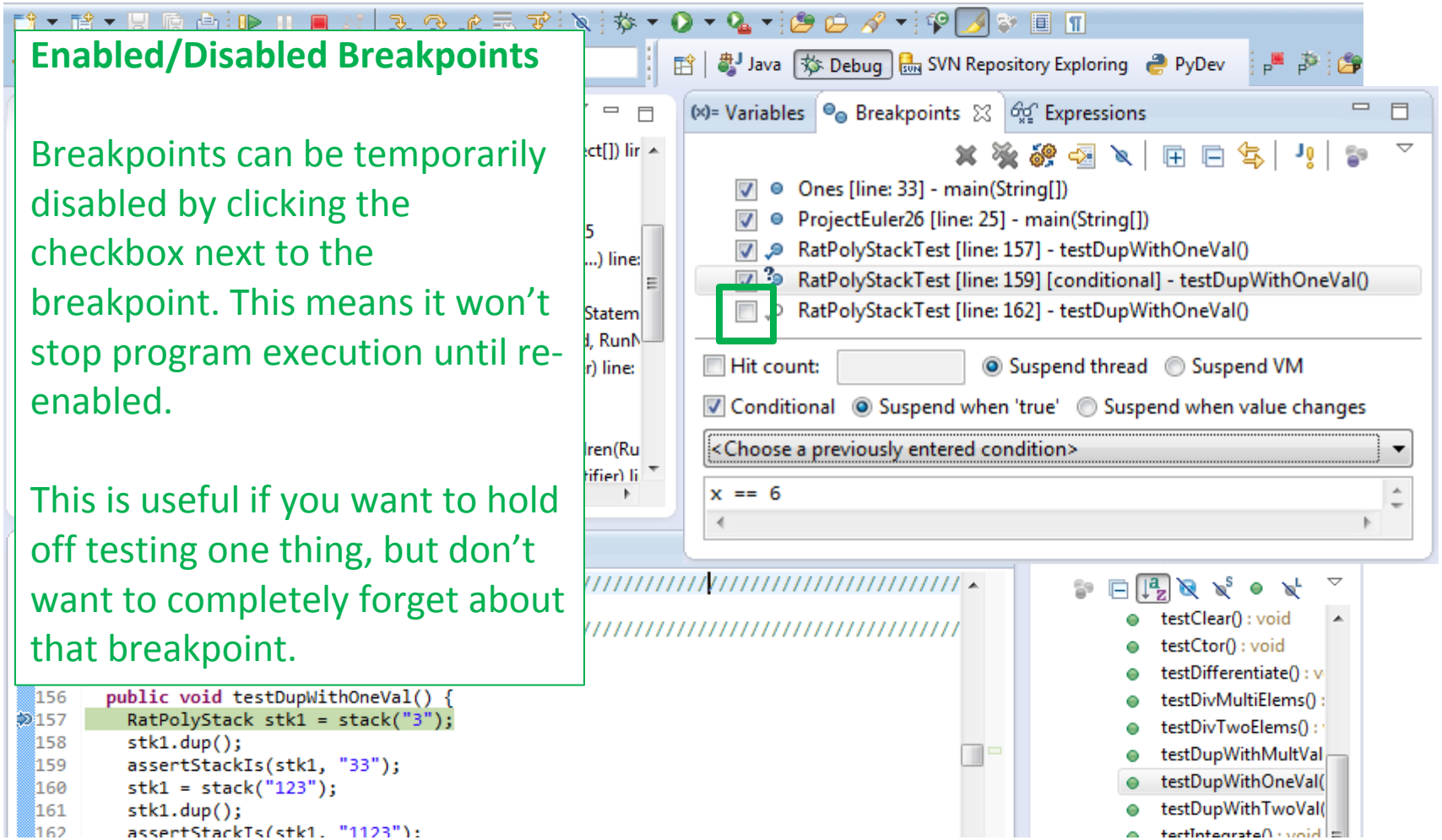
testClear() : void
testCtor() : void
testDifferentiate() : v
testDivMultiElems() :
testDivTwoElems() :
testDupWithMultVal
testDupWithOneVal(
testDupWithTwoVal(
testIntegrate() : void

Eclipse Debugging

Enabled/Disabled Breakpoints

Breakpoints can be temporarily disabled by clicking the checkbox next to the breakpoint. This means it won't stop program execution until re-enabled.

This is useful if you want to hold off testing one thing, but don't want to completely forget about that breakpoint.



Eclipse Debugging

Hit count

Breakpoints can be set to occur less-frequently by supplying a hit count of *n*.

When this is specified, only each *n*-th time that breakpoint is hit will code execution stop.

Quick Access | Java | Debug | SVN Repository Exploring | PyDev

Debug

DelegatingMethodAccessorImpl.invoke(Object, Object[]) lir

Variables | Breakpoints | Expressions

- Ones [line: 33] - main(String[])
- ProjectEuler26 [line: 25] - main(String[])
- RatPolyStackTest [line: 157] - testDupWithOneVal()
- RatPolyStackTest [line: 159] [conditional] - testDupWithOneVal()
- RatPolyStackTest [line: 162] - testDupWithOneVal()

Hit count: | Suspend thread | Suspend VM

Conditional | Suspend when 'true' | Suspend when value changes

<Choose a previously entered condition>

x == 6

```
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

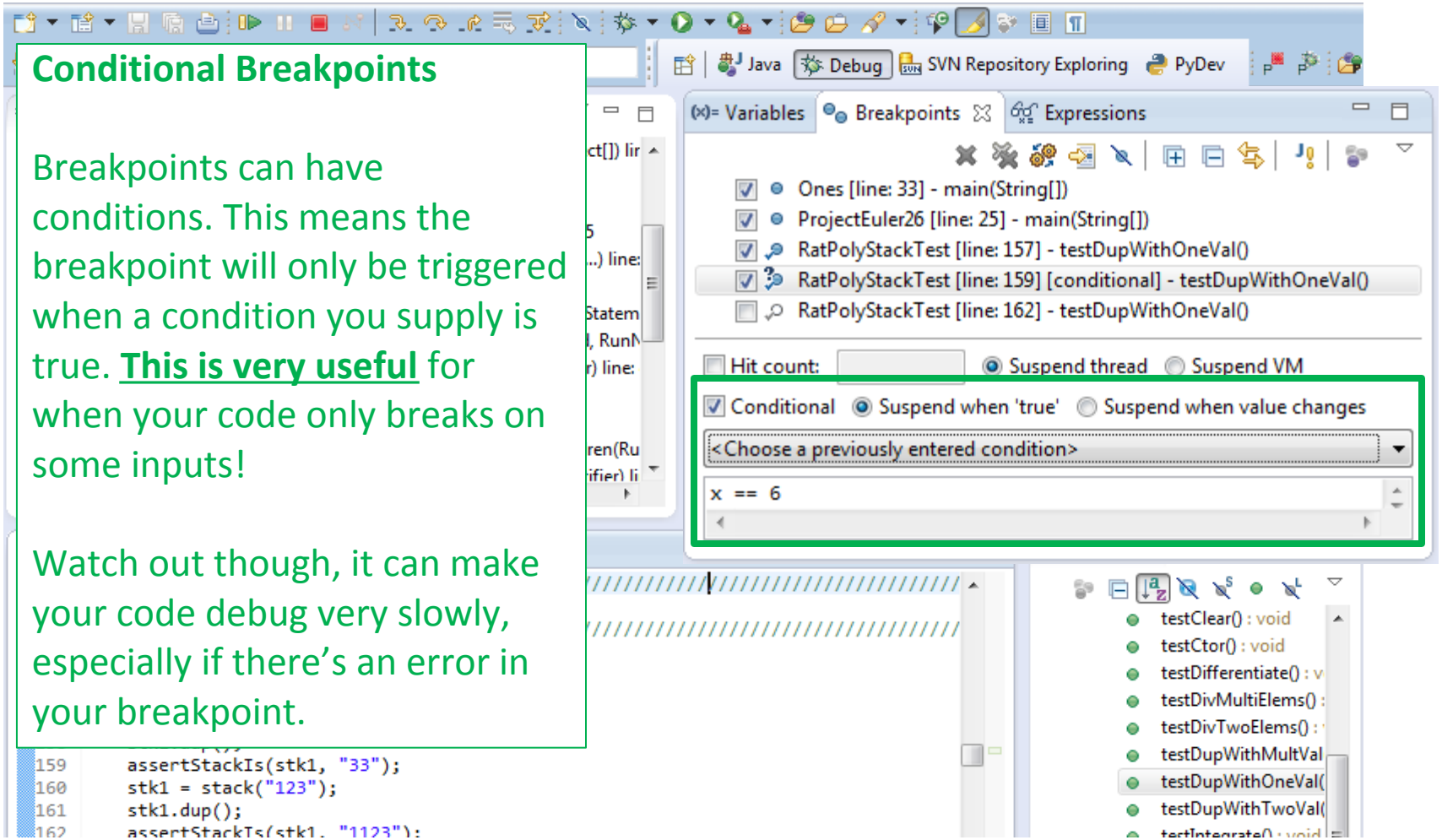
testClear() : void
testCtor() : void
testDifferentiate() : v
testDivMultiElems() :
testDivTwoElems() :
testDupWithMultVal
testDupWithOneVal(
testDupWithTwoVal(

Eclipse Debugging

Conditional Breakpoints

Breakpoints can have conditions. This means the breakpoint will only be triggered when a condition you supply is true. **This is very useful** for when your code only breaks on some inputs!

Watch out though, it can make your code debug very slowly, especially if there's an error in your breakpoint.

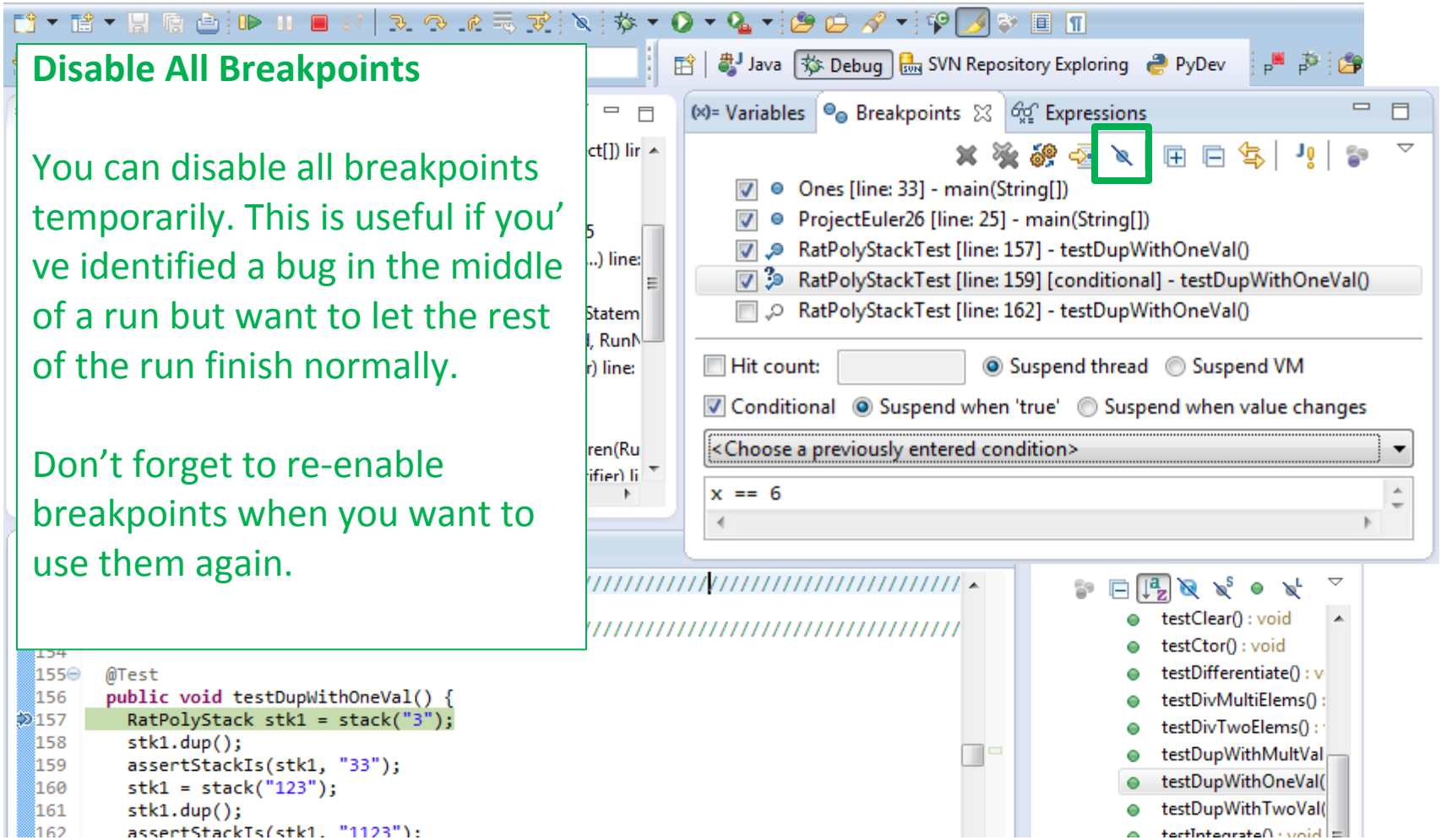


Eclipse Debugging

Disable All Breakpoints

You can disable all breakpoints temporarily. This is useful if you've identified a bug in the middle of a run but want to let the rest of the run finish normally.

Don't forget to re-enable breakpoints when you want to use them again.



Eclipse Debugging

Break on Java Exception

Eclipse can break whenever a specific exception is thrown. This can be useful to trace an exception that is being "translated" by library code.

Java | Debug | SVN Repository Exploring | PyDev

Variables | Breakpoints | Expressions

- Ones [line: 33] - main(String[])
- ProjectEuler26 [line: 25] - main(String[])
- RatPolyStackTest [line: 157] - testDupWithOneVal()
- RatPolyStackTest [line: 159] [conditional] - testDupWithOneVal()
- RatPolyStackTest [line: 162] - testDupWithOneVal()

Hit count: | Suspend thread | Suspend VM

Conditional | Suspend when 'true' | Suspend when value changes

<Choose a previously entered condition>

x == 6

ParentRunner$1.schedule(Runnable) line: 60

BlockJUnit4ClassRunner(ParentRunner<T>).runChildren(Ru

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

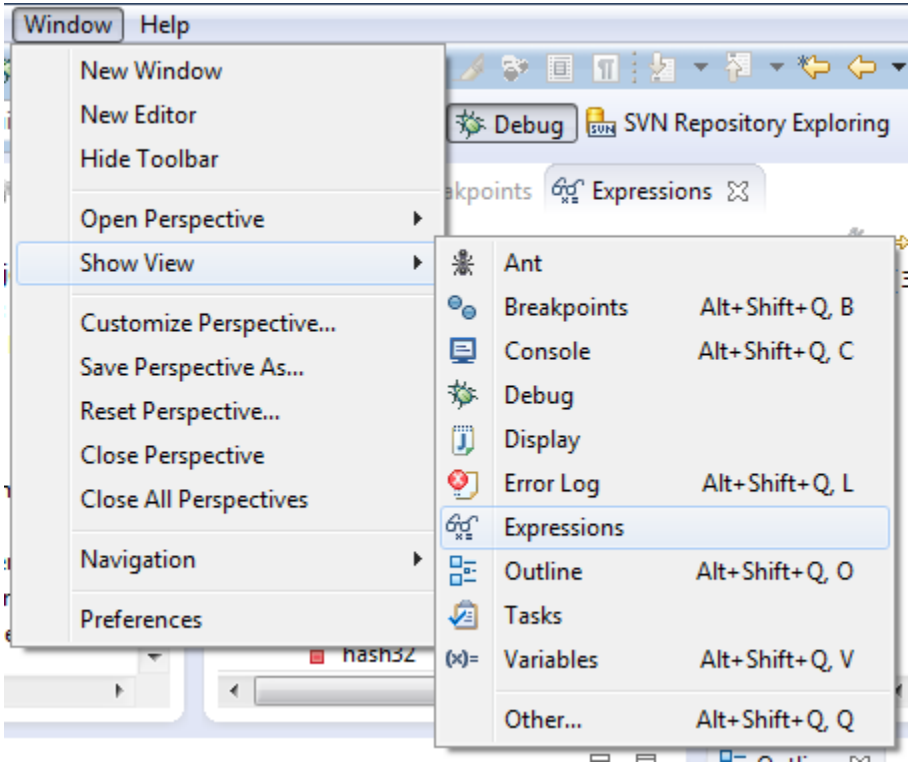
testClear() : void
testCtor() : void
testDifferentiate() : v
testDivMultiElems() :
testDivTwoElems() :
testDupWithMultVal
testDupWithOneVal(
testDupWithTwoVal(

Eclipse Debugging

Expressions Window

Used to show the results of custom expressions you provide, and can change any time.

Not shown by default but highly recommended.



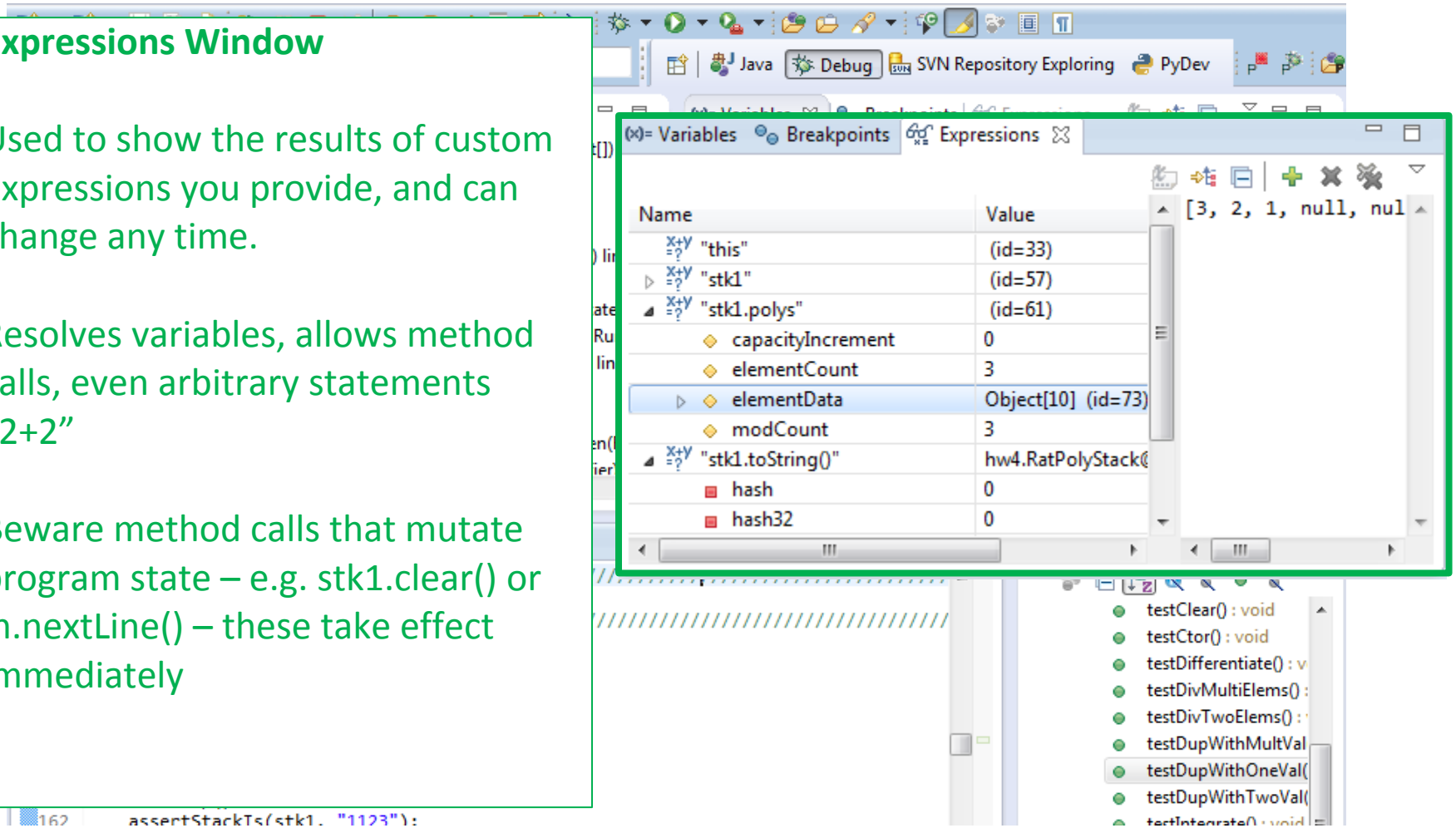
Eclipse Debugging

Expressions Window

Used to show the results of custom expressions you provide, and can change any time.

Resolves variables, allows method calls, even arbitrary statements "2+2"

Beware method calls that mutate program state – e.g. stk1.clear() or in.nextLine() – these take effect immediately



Eclipse Debugging

Expressions Window

These persist across projects, so clear out old ones as necessary.

Java | Debug | SVN Repository Exploring | PyDev

FrameworkMethod.invokeExplosively(Object, Object...)
InvokeMethod.evaluate() line: 20
BlockJUnit4ClassRunner(ParentRunner<T>).runLeaf(State
BlockJUnit4ClassRunner.runChild(FrameworkMethod, Ru
BlockJUnit4ClassRunner.runChild(Object, RunNotifier) lin
ParentRunner$3.run() line: 231
ParentRunner$1.schedule(Runnable) line: 60
BlockJUnit4ClassRunner(ParentRunner<T>).runChildren(

Variables | Breakpoints | Expressions

Name	Value
"this"	(id=33)
"stk1"	(id=57)
"stk1.polys"	(id=61)
capacityIncrement	0
elementCount	3
elementData	Object[10] (id=73)
modCount	3
"stk1.toString()"	hw4.RatPolyStack@
hash	0
hash32	0

[3, 2, 1, null, nul

RatPolyStackTest.java

```
151  ////////////////////////////////////////////////////////////////////////////////
152  ////  Duplicate
153  ////////////////////////////////////////////////////////////////////////////////
154
155  @Test
156  public void testDupWithOneVal() {
157    RatPolyStack stk1 = stack("3");
158    stk1.dup();
159    assertStackIs(stk1, "33");
160    stk1 = stack("123");
161    stk1.dup();
162    assertStackIs(stk1, "1123");
```

testClear() : void
testCtor() : void
testDifferentiate() : v
testDivMultiElems() :
testDivTwoElems() :
testDupWithMultVal
testDupWithOneVal(
testDupWithTwoVal(

Eclipse Debugging

- Demo