# CSE 331: Developer Tools

Section 2
01/16/2013

Modified by: David Mailhot
Originally by: Kellen Donohue, Krysta Yousoufian

# You Have Section Homework

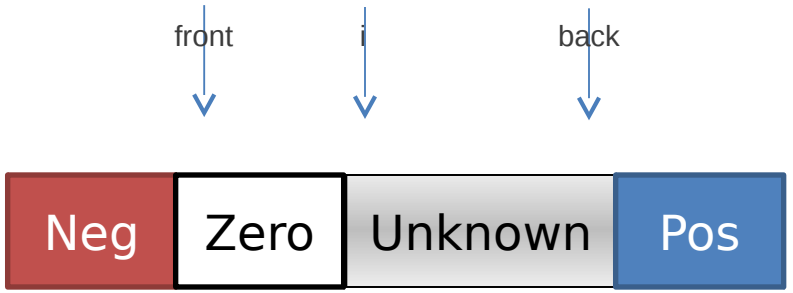
1) Download the zipped project linked on the website under section 2 and import it into eclipse
2) Run the PokemonTest.java JUnit test and **solve the typos in the code**
3) Generate javadoc and **complete the specification for Pokemon.battle()**
4) Upload the entire project directory to your cse331 repository

# Agenda

- Loop development review
- Tools
  - Eclipse
  - Subversion
  - JUnit

# What's wrong?

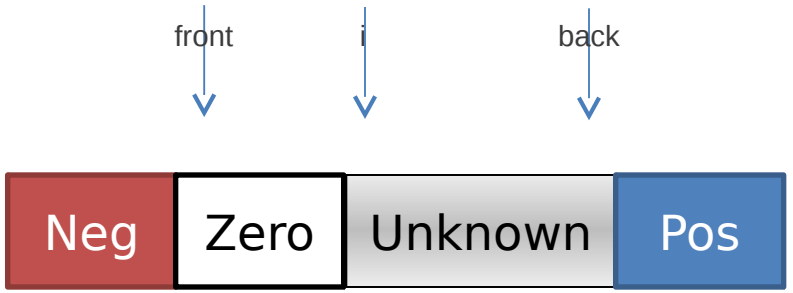
```
public static void partition(int[] b) {
    int frontIndex = 0;
    int backIndex = b.length - 1;
    for (int i = 0; i <= backIndex; i++) {
      if (b[i] < 0) {
        swap(b, frontIndex, i);
        frontIndex++;
      } else if (b[i] > 0) {
        swap(b, backIndex, i);
        backIndex--;
      }
    }
  }
```



# What's wrong?

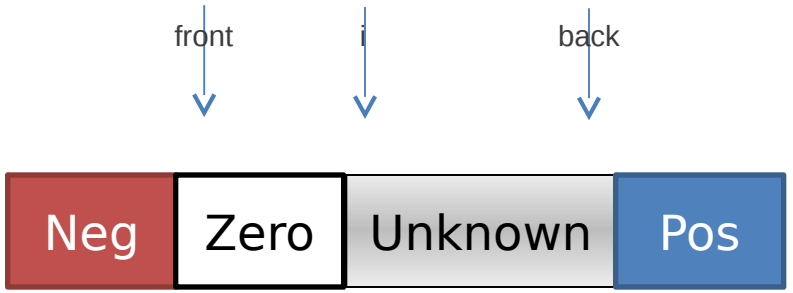
```
public static void partition(int[] b) {
    int frontIndex = 0;
    int backIndex = b.length - 1;
    for (int i = 0; i <= backIndex; i++) {
      if (b[i] < 0) {
        swap(b, frontIndex, i);
        frontIndex++;
      } else if (b[i] > 0) {
        swap(b, backIndex, i);
        backIndex--;
      }
    }
  }
```

[0, -1, 2, -3]



# What's wrong?

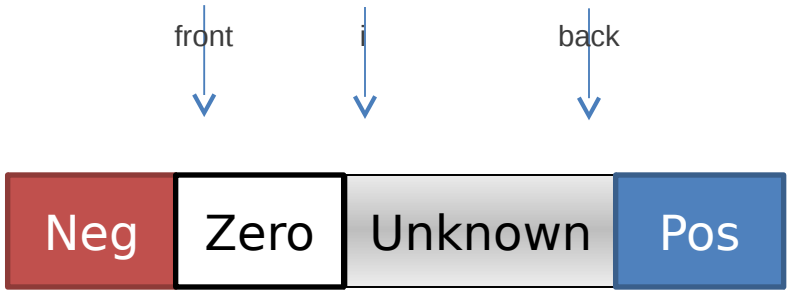
```
public static void partition(int[] b) {
    int frontIndex = 0;
    int backIndex = b.length - 1;
    for (int i = 0; i <= backIndex; i++) {
      if (b[i] < 0) {
        swap(b, frontIndex, i);
        frontIndex++;
      } else if (b[i] > 0) {
        swap(b, backIndex, i);
        backIndex--;
      }
    }
  }
```




[0, -1, 2, -3] => [-1, 0, 2, -3]

# What's wrong?

```
public static void partition(int[] b) {
    int frontIndex = 0;
    int backIndex = b.length - 1;
    for (int i = 0; i <= backIndex; i++) {
      if (b[i] < 0) {
        swap(b, frontIndex, i);
        frontIndex++;
      } else if (b[i] > 0) {
        swap(b, backIndex, i);
        backIndex--;
      }
    }
  }
```




[0, -1, 2, -3] => [-1, 0, 2, -3] => [-1, 0, -3, 2]

# What’s wrong?

```
public static void partition(int[] b) {
    int frontIndex = 0;
    int backIndex = b.length - 1;
    for (int i = 0; i <= backIndex; i++) {
      if (b[i] < 0) {
        swap(b, frontIndex, i);
        frontIndex++;
      } else if (b[i] > 0) {
        swap(b, backIndex, i);
        backIndex--;
        i--;
      }
    }
  }
```

# Loop development example

- Given array a = [0, ..., n-1], reverse the elements in a
- pre:


- post:

# Loop development example

- Given array a = [0, ..., n-1], reverse the elements in a

- pre:


- loop-inv:


- post:

# Loop development example

- loop-inv:


```
L = 0;
R = n-1;
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;
}
```

# Loop development example

- loop-inv:


```
L = 0;
R = n-1;
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;
}
```

# Loop development example

- loop-inv:


[0..L-1] and [R+1..n-1] are reversed, rest normal

```
L = 0;
R = n-1;
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;
}
```

# Loop development example

- loop-inv:


[0..L-1] and [R+1..n-1] are reversed, rest normal

```
L = 0;
R = n-1;                    // I) True before loop
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;
}
```

# Loop development example

- loop-inv:


[0..L-1] and [R+1..n-1] are reversed, rest normal

```
L = 0;
R = n-1;                    // I) True before loop
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;                  // II) True inductively
}
```

# Loop development example

- loop-inv:


[0..L-1] and [R+1..n-1] are reversed, rest normal

```
L = 0;
R = n-1;                    // I) True before loop
while (L < R) {
  swap(a[L],a[R]);
  L = L+1;
  R = R-1;                // II) True inductively
}                            // III) True after loop
```

# Agenda

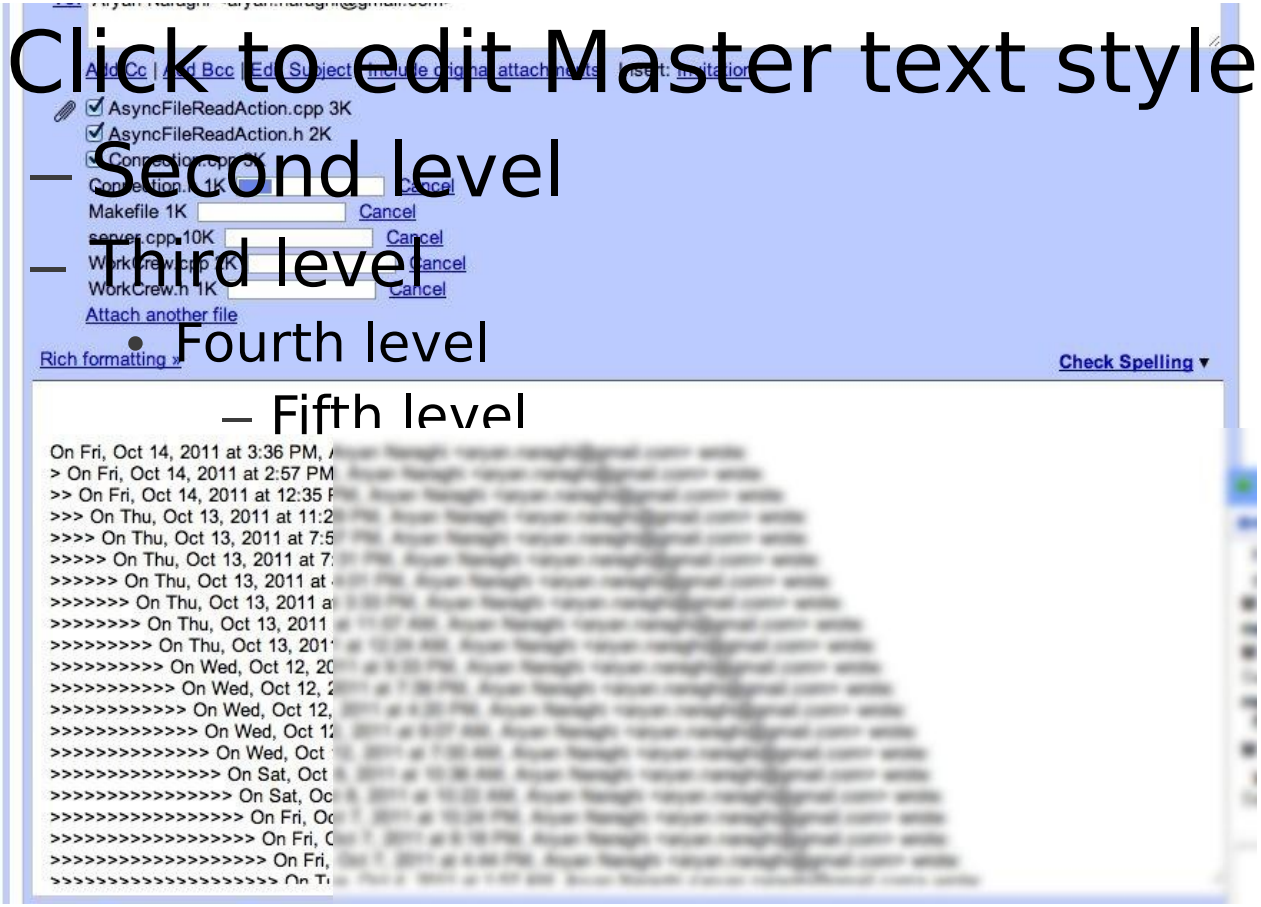
- Loop development on ex0
- Tools
  - Subversion
  - Eclipse
  - Subclipse
  - JUnit

# Version Control

- System for tracking changes to code
  - Essential for managing big projects
  - Learn it now – you WILL use it again and again!
- Makes it easy to:
  - See a history of changes
  - Revert back to an older version of your code
  - Back up your work
  - Work on code in a team
  - Work on different machines
- You'll use Subversion (SVN) this quarter
  - There are others: Mercurial, Git, CVS, …

# Don’t be this guy

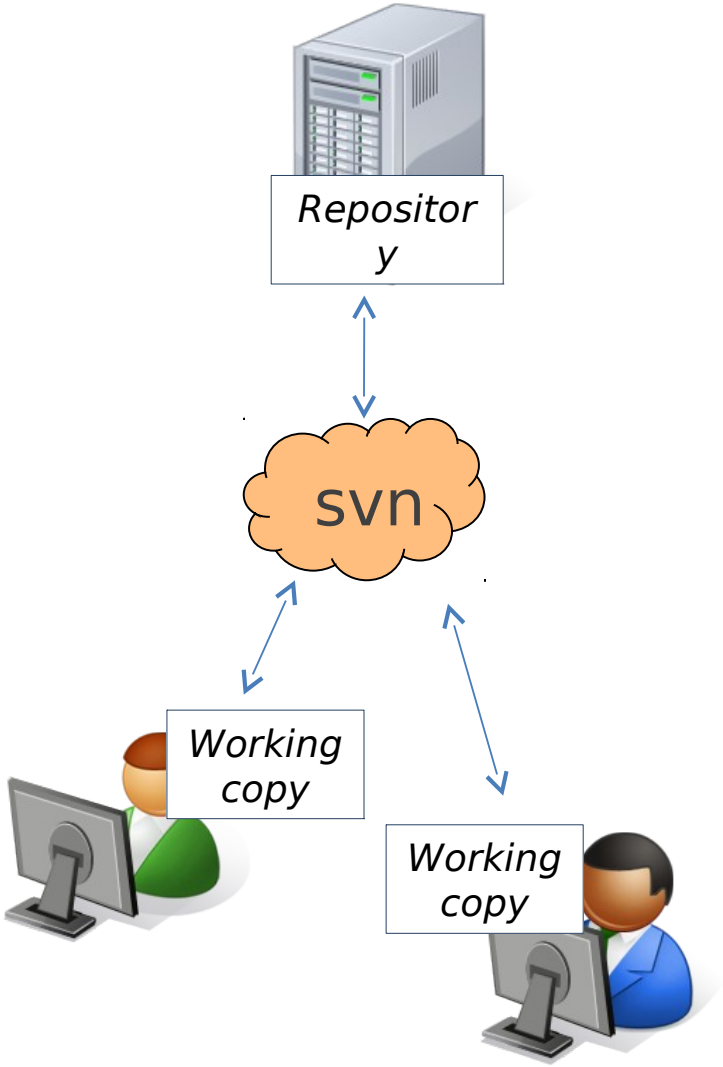
- Click to edit Master text styles
  - Second level
    - Third level
      - Fourth level
        - Fifth level



# Organization

- A *repository* stores the master copy of the project
  - Someone creates the repo for a new project
  - Then nobody touches this copy directly
  - Lives on a server everyone can access
- Each person checks out their own working copy
  - Makes a local copy of the repo
  - You'll always work off of this copy
  - The version control system syncs the repo and working copy

svn checkout



# SVN Workflow

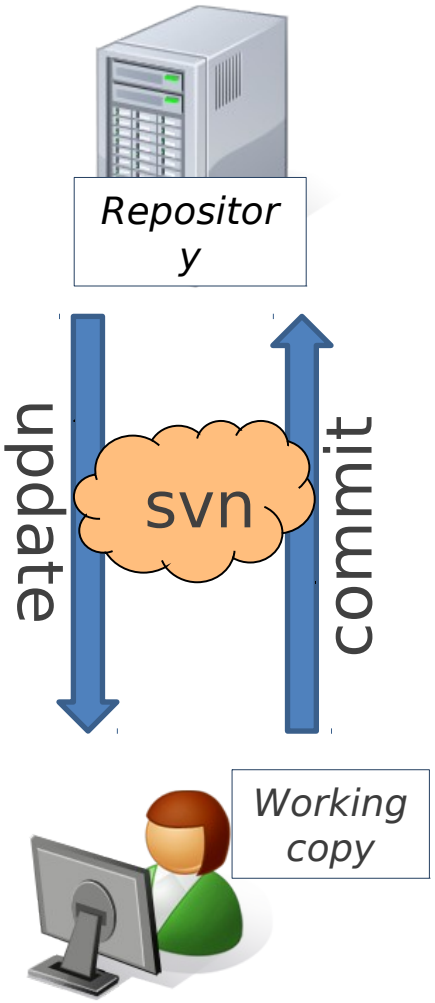
Most common commands:

- Status
  - Displays the status of your local files (the working copy)

```
svn status
```

- Add
  - Add a new local file to your working copy

```
svn add [file]
```



# SVN Workflow

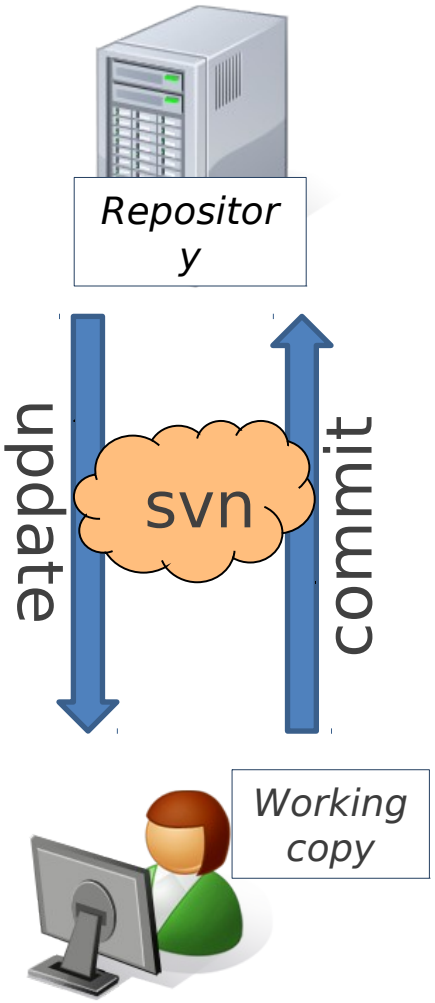
Most common commands:

- Commit / checkin
  - Push changes from your working copy to the repository

```
svn commit
```

- Update
  - Pull changes from the repository to your working copy

```
svn update
```



# This Quarter

- Use Subversion for your homework assignments
- We distribute starter code by adding it to your repo
- You turn in your files by **adding** them to the repo and **committing** your changes
- Run validator tool to make sure you added everything correctly, etc.
- See the version control handout:

  http://www.cs.washington.edu/education/courses/cse331/12au/tools/versioncontrol.html

# How to use SVN

- Command line (Mac, Linux)
  - `svn help` – List commands
  - `svn help checkout` – Options for checkout
- Subclipse Plugin for Eclipse (All OS's)
- GUI interfaces – TortoiseSVN (Windows)

# How to get your Code

Your SVN repository on attu is located at:
svn+ssh://**Your-CSE-Net-ID**@attu.cs.washington.edu/projects/instr/13wi/cse331/**Your-CSE-Net-ID**/REPOS

## Command line:

svn checkout svn+ssh://**Your-CSE-Net-ID**@attu.cs.washington.edu/projects/instr/13wi/cse331/**Your-CSE-Net-ID**/REPOS

## Subclipse:

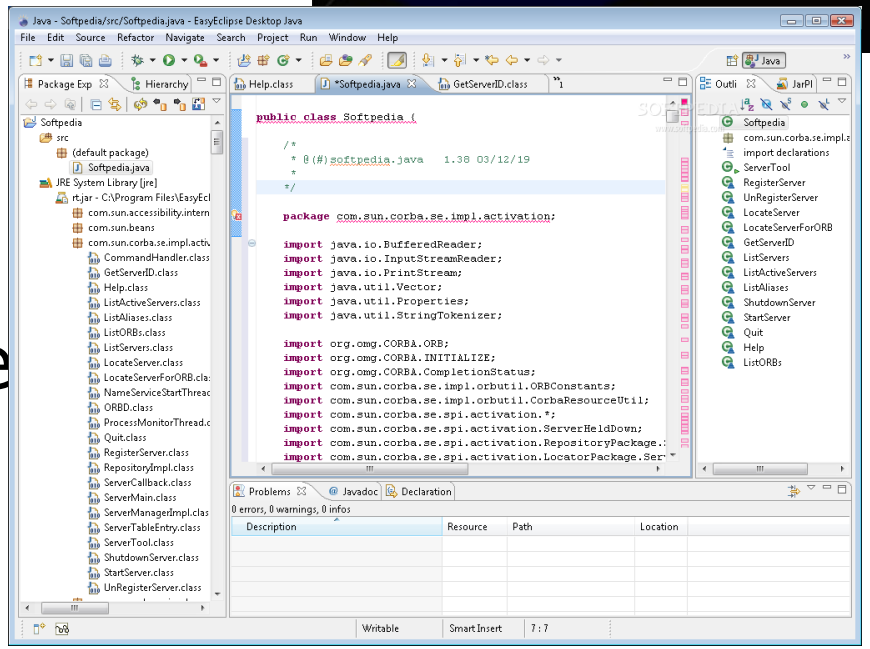
'File' -> 'Import' -> 'SVN' -> 'Checkout...'

Create new

Url: svn+ssh://**Your-CSE-Net-ID**@attu.cs.washington.edu/projects/instr/13wi/cse331/**Your-CSE-Net-ID**/REPOS

# Eclipse

- Eclipse is a multi-platform, open-source IDE
- Build, edit, run, test, distribute your code from one program

# Nice features of Eclipse

- Code generation
- Easy refactoring/renaming
- Helpful autocomplete
- Easily see relevant documentation (F3, Ctrl+Shift+T)
- Quickly find variable uses/definitions (hover)
- Debugging
- Good integration with other tools

# Demo

# Getting Eclipse

- It’s already installed on CSE Lab Machines
  - Open a terminal – type `eclipse &`
- Working from home (instructions in tools handout)
  1. Download Java JDK (Version 7)
  2. Set JAVA_HOME environment variable
  3. Download Eclipse

# Alternatives

- Other IDEs: jEdit, Netbeans
- vim / Emacs / gedit / Notepad++ / Textmate
  & command line
- If you've only used one environment before – try Eclipse
- Course staff will support Eclipse – something else and you're (more) on your own

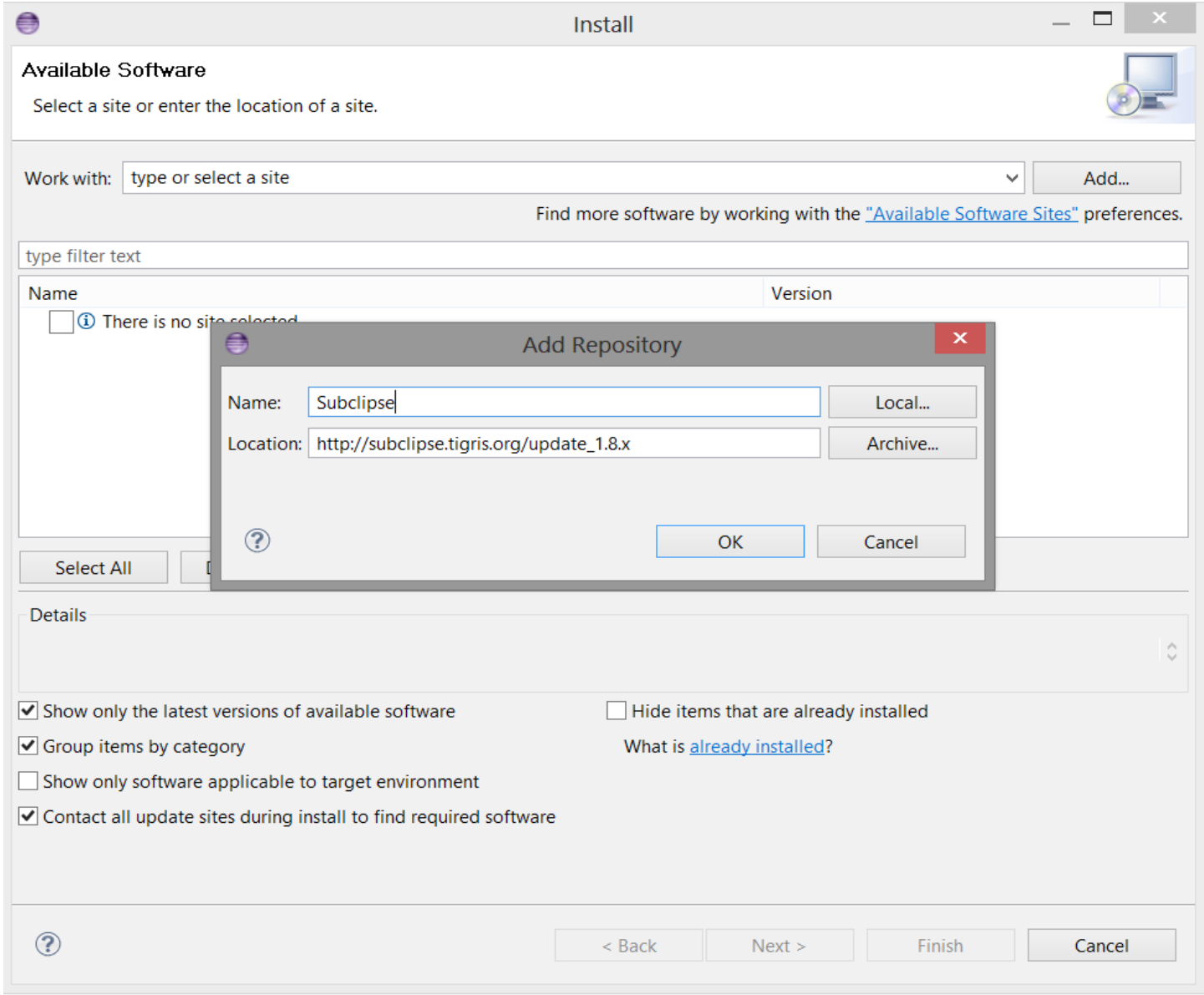
# Installing Subclipse

Open Eclipse

'Help' menu -> 'Install New Software'

'Add' button
Name: Subclipse
Location: http://subclipse.tigris.org/update_1.8.x

'Next', 'Agree', 'Finish', etc


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
There is no site selected
Add Repository
Name:
Subclipse
Local...
Location:
http://subclipse.tigris.org/update_1.8.x
Archive...
OK
Cancel
Select All
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

# Subclipse Demo

# JUnit

- You wrote a lot of code in Eclipse, and committed it all in Subversion – but does it work?
  - And will it work tomorrow?
  - If there's a bug how do we know it's fixed?
  - If something else changes will our code break?
- Unit tests can assuage these fears
- JUnit is a unit-testing framework for Java we will use extensively this quarter

# A JUnit test class

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

public class PointTest {
    ...

    @Test
    public void testDistance() { // a test case method
        ...
    }
}
```

A method with `@Test` is flagged as a JUnit test case.

- All `@Test` methods run when JUnit runs your test class.

# Verifying Behavior with Assertions

- Assertions: special JUnit methods
- Verifies that a value matches expectations
  - `assertEquals(42, meaningOfLife());` *fails if meaningOfLife() != 42*
  - `assertTrue(list.isEmpty());` *fails if list.isEmpty() is false*
- If the value isn't what it should be, the test fails
  - Test immediately terminates
  - Other tests in the test class are still run as normal
  - Results show details of failed tests

# Using Assertions

| | |
|---|---|
| `assertTrue(`**`test`**`)` | fails if the boolean test is `false` |
| `assertFalse(`**`test`**`)` | fails if the boolean test is `true` |
| `assertEquals(`**`expected`**`, `**`actual`**`)` | fails if the values are not equal |
| `assertSame(`**`expected`**`, `**`actual`**`)` | fails if the values are not the same (by ==) |
| `assertNotSame(`**`expected`**`, `**`actual`**`)` | fails if the values *are* the same (by ==) |
| `assertNull(`**`value`**`)` | fails if the given value is *not* `null` |
| `assertNotNull(`**`value`**`)` | fails if 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:
  - e.g. `assertEquals("`**`message`**`", `**`expected`**`, `**`actual`**`)`

# Checking for Exceptions

- Verify that a method throws an exception
- Place above method:

```
@Test(expected=IllegalArgumentException.class)
```

- Test passes if specified exception is thrown, fails otherwise

- Only time it's OK to write a test with no asserts!

```
// Try to access the first item in an empty ArrayList
@Test(expected=IndexOutOfBoundsException.class)
 public void test() {
    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

- 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() { ... }
```

# JUnit and Eclipse

- Eclipse can easily run JUnit tests and report results.

  *Open the java file*

  *Run -> Run As -> JUnit Test*

- This is when the Eclipse debugger is especially helpful!

- Demo

# Putting it all together

- HW3 out later today or tomorrow
  - Mostly environment setup & introduction
  - Uses all tools described here
  - Tools handouts on website soon
  - If you get stuck, ask for help!
  - Message board