Agenda

- Loop development & ex0
- Tools
  - Eclipse
  - Subversion
  - JUnit
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--;
        }
    }
}
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?

```java
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?

```java
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]
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 $a = [0, \ldots, n-1]$, reverse the elements in $a$

• pre:
  
  | a[0] | a[1] | \ldots | a[n-2] | a[n-1] |

• post:
  
  | a[n-1] | a[n-2] | \ldots | a[1] | a[0] |
Loop development example

• Given array $a = [0, \ldots, n-1]$, reverse the elements in $a$

- **pre:**
  - $a[0]$     $a[1]$    ...     $a[n-2]$     $a[n-1]$

- **loop-inv:**
  - $a[n-1]$    $a[1]$    ...    $a[n-2]$        $a[0]$

- **post:**
  - $a[n-1]$    $a[n-2]$    ...    $a[1]$        $a[0]$
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:

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

\[
L = 0; \\
R = n-1; \\
\text{while (}L < R\text{) } \{ \\
\quad \text{swap}(a[L], a[R]); \\
\quad L = L+1; \\
\quad R = R-1; \\
\}
\]
Loop development example

- **loop-inv:**

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

  
  \[
  \begin{array}{c}
  a[n-1] \quad a[1] \quad \ldots \quad a[n-2] \quad a[0]
  \end{array}
  \]

\[
L = 0;
R = n-1;
\]

\[
\text{// I) True before loop}
\]

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

...
Loop development example

- loop-inv:

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

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
}

\[ a[n-1] \quad a[1] \quad ... \quad a[n-2] \quad a[0] \]
• **loop-inv:**

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

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
  – Eclipse
  – Subversion
  – JUnit
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
• Quickly find variable uses/definitions
• 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
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
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
Common Actions

Most common commands:

• **Commit / checkin**
  – integrate changes *from* your working copy *into* the repository

• **Update**
  – integrate changes *into* your working copy *from* the repository
Common Actions

Most common commands:

• **Add, delete**
  - add or delete a file in the repository
  - just putting a new file in your working copy **does not add it to the repo**

Dropbox is a similar idea, but it adds every file, does commits for every change, and pulls anytime a file is changed elsewhere.
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/cse331/1au/tools/version control.html
How to use SVN

• Command line
  – 
  

• Subclipse Plugin for Eclipse

• GUI interfaces -- TortoiseSVN
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
  ```java
  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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assertTrue(test)</td>
<td>fails if the boolean test is false</td>
</tr>
<tr>
<td>assertFalse(test)</td>
<td>fails if the boolean test is true</td>
</tr>
<tr>
<td>assertEquals(expected, actual)</td>
<td>fails if the values are not equal</td>
</tr>
<tr>
<td>assertSame(expected, actual)</td>
<td>fails if the values are not the same (by ==)</td>
</tr>
<tr>
<td>assertNotSame(expected, actual)</td>
<td>fails if the values are the same (by ==)</td>
</tr>
<tr>
<td>assertNull(value)</td>
<td>fails if the given value is not null</td>
</tr>
<tr>
<td>assertNotNull(value)</td>
<td>fails if the given value is null</td>
</tr>
</tbody>
</table>

- And others: [http://www.junit.org/apidocs/org/junit/Assert.html](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:

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

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

```java
@BeforeClass
public static void name() { ... }
@AfterClass
public static void name() { ... }
```
JUnit and Eclipse

• Eclipse can easily run JUnit tests and report results.

• 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