SECTION 2:
CODE REASONING +
PROGRAMMING
TOOLS

cse331-staff@cs.washington.edu

slides borrowed and adapted from Alex Mariakis and CSE 390a
OUTLINE

• Reasoning about code
• Developer tools
  • Eclipse and Java versions
  • ssh
  • Version control
REASONING ABOUT CODE

• Two purposes
  • Prove our code is correct
  • Understand why code is correct
• Forward reasoning: determine what follows from initial conditions
• Backward reasoning: determine sufficient conditions to obtain a certain result
FORWARD REASONING

// {x >= 0, y >= 0}
y = 16;
//
x = x + y
//
x = sqrt(x)
//
y = y - x
//
FORWARD REASONING

// {x >= 0, y >= 0}
y = 16;
// {x >= 0, y = 16}
x = x + y
//
x = sqrt(x)
//
y = y - x
//
FORWARD REASONING

// {x >= 0, y >= 0}
y = 16;
// {x >= 0, y = 16}
x = x + y
// {x >= 16, y = 16}
x = sqrt(x)
//
y = y - x
//
FORWARD REASONING

// {x >= 0, y >= 0}
y = 16;
// {x >= 0, y = 16}
x = x + y
// {x >= 16, y = 16}
x = sqrt(x)
// {x >= 4, y = 16}
y = y - x
//
FORWARD REASONING

// {x >= 0, y >= 0}
y = 16;

// {x >= 0, y = 16}
x = x + y

// {x >= 16, y = 16}
x = sqrt(x)

// {x >= 4, y = 16}
y = y - x

// {x >= 4, y <= 12}
FORWARD REASONING

// {true}  
if (x>0) {  
//  
abs = x
//
}
else {  
//
abs = -x
//  
}
}//
//
FORWARD REASONING

// {true}
if (x>0) {
    // {x > 0}
    abs = x
    //
}
else {
    // {x <= 0}
    abs = -x
    //
}
//
//
FORWARD
REASONING

// {true}
if (x>0) {
    // {x > 0}
    abs = x
    // {x > 0, abs = x}
}
else {
    // {x <= 0}
    abs = -x
    // {x <= 0, abs = -x}
}

//
//
FORWARD REASONING

// {true}
if (x>0) {
    // {x > 0}
    abs = x
    // {x > 0, abs = x}
}
else {
    // {x <= 0}
    abs = -x
    // {x <= 0, abs = -x}
}
// {x > 0, abs = x OR x <= 0, abs = -x}
//
FORWARD REASONING

// {true}
if (x>0) {
    // {x > 0}
    abs = x
    // {x > 0, abs = x}
}
else {
    // {x <= 0}
    abs = -x
    // {x <= 0, abs = -x}
}

// {x > 0, abs = x OR x <= 0, abs = -x}
// {abs = |x|}
//
a = x + b;
//
c = 2b - 4
//
x = a + c
// {x > 0}
BACKWARD REASONING

//
a = x + b;
//
c = 2b - 4
// {a + c > 0}
x = a + c
// {x > 0}
BACKWARD REASONING

//

a = x + b;
// {a + 2b - 4 > 0}
c = 2b - 4
// {a + c > 0}
x = a + c
// {x > 0}
BACKWARD REASONING

// {x + 3b - 4 > 0}
a = x + b;

// {a + 2b - 4 > 0}
c = 2b - 4

// {a + c > 0}
x = a + c

// {x > 0}
IMPLICATION

- Hoare triples are just an extension of logical implication
  - Hoare triple: \{P\} S \{Q\}
  - P \rightarrow Q after statement S

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P → Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
IMPLICATION

- Hoare triples are just an extension of logical implication
  - Hoare triple: \{P\} S \{Q\}
  - P → Q after statement S
- Everything implies true
- False implies everything

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P → Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
WEAKER VS. STRONGER

• If $P_1 \rightarrow P_2$, then

  • $P_1$ is stronger than $P_2$
  • $P_2$ is weaker than $P_1$

• Weaker statements are more general, stronger statements say more

• Stronger statements are more restrictive

• Ex: $x = 16$ is stronger than $x > 0$

• Ex: “Alex is an awesome TA” is stronger than “Alex is a TA”
**WEAKEST PRECONDITION**

- The most lenient assumptions such that a postcondition will be satisfied

- If $P^*$ is the weakest precondition for $\{P\} S \{Q\}$, then $P \rightarrow P^*$ for all $P$ that make the Hoare triple valid

- $WP = wp(S, Q)$, which can be found using backward reasoning
  - Ex: $wp(x = y+4, x > 0) = y+4>0$
DEVELOPER TOOLS

- Eclipse and Java versions
- Remote access
- Version control redux
ECLIPSE

• Get Java 7

• Important: Java separates compile and execution, eg:
  • javac Example.java  produces  Example.class
  • Both compile and execute have to be the same Java!
WHAT IS AN SSH CLIENT?

- Uses the secure shell protocol (SSH) to connect to a remote computer
  - Enables you to work on a lab machine from home
  - Similar to remote desktop
- **Windows users:** PuTTY and WinSCP
  - PuTTY: ssh connection
  - WinSCP: transfer or edit files
- **Mac/Linux users:** Terminal application
  - Go to Applications/Utilities/Terminal
  - Type in “ssh cseNetID@attu.cs.washington.edu”
  - “ssh -XY cseNetID@attu.cs.washington.edu” lets you use GUIs
PUTTY

PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to
Host Name (or IP address) Port
meganca@attu.cs.washington.edu 22

Connection type:
- Raw
- Telnet
- Rlogin
- SSH
- Serial

Load, save or delete a stored session
Saved Sessions
Default Settings
meganca@helsinki.cs.washington.edu

Close window on exit:
- Always
- Never
- Only on clean exit

PuTTY Command Line:

```
putty
```

Connection:

Attu.cs.washington.edu - PuTTY

Using username "meganca".
meganca@attu.cs.washington.edu's password:

Please remove core files when you are done with them, as they tend to take up a lot of space on the disk. If everyone removes them when they are done debugging, there is going to be a lot more disk space to go around.

Thanks!

[meganca@attu ~]$
Terminal (Linux, Mac)

```
meganca@charmander:~$ ssh meganca@attu.cs.washington.edu
meganca@attu.cs.washington.edu's password:

Use passwd to change your password.
Use chsh to change your shell.

Contact support@cs if you need assistance.

Please remove core files when you are done with them, as they tend to take up a lot of space on the disk. If everyone removes them when they are done debugging, there is going to be a lot more disk space to go around. Thanks!

[meganca@attu3 ~]$  
```
DEMO #1

WHAT IS UNIX?

Multiuser modular operating system

- Traditionally command-line based
- Mac OS X is Unix-based!

<table>
<thead>
<tr>
<th>Command</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwd</td>
<td>prints the name of the working directory</td>
</tr>
<tr>
<td>ls</td>
<td>lists the files in a directory (i.e., lists stuff)</td>
</tr>
<tr>
<td>cd</td>
<td>changes a directory</td>
</tr>
<tr>
<td>cp</td>
<td>copies a file or directory</td>
</tr>
<tr>
<td>mv</td>
<td>move/rename a file or directory</td>
</tr>
<tr>
<td>rm</td>
<td>removes a file</td>
</tr>
<tr>
<td>mkdir</td>
<td>make a new directory</td>
</tr>
<tr>
<td>rmdir</td>
<td>remove an empty directory</td>
</tr>
<tr>
<td>man</td>
<td>pulls up the manual pages</td>
</tr>
</tbody>
</table>
VERSION CONTROL

Repository

update

svn

commit

Working copy
331 VERSION CONTROL

create

check out

Repository

Working copy for grading

update

Working copy

add

commit
331 VERSION CONTROL

- Your repo is at /projects/instr/14au/cse331/YourCSENetID/REPOS/cse331
- Only check out once (unless you’re working in a lot of places)
- Don’t forget to add files!!
- Check in your work!
VERSION CONTROL: GUI
### VERSION CONTROL: COMMAND-LINE

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svn co repo</code></td>
<td>check out</td>
</tr>
<tr>
<td><code>svn ci [files]</code></td>
<td>commit / check in changed files</td>
</tr>
<tr>
<td><code>svn add files</code></td>
<td>schedule files to be added at next commit</td>
</tr>
<tr>
<td><code>svn help [command]</code></td>
<td>get help info about a particular command</td>
</tr>
<tr>
<td><code>svn merge source1 source2</code></td>
<td>merge changes</td>
</tr>
<tr>
<td><code>svn revert files</code></td>
<td>restore local copy to repo's version</td>
</tr>
<tr>
<td><code>svn resolve files</code></td>
<td>resolve merging conflicts</td>
</tr>
<tr>
<td><code>svn update [files]</code></td>
<td>update local copy to latest version</td>
</tr>
</tbody>
</table>

**Others:** blame, changelist, cleanup, diff, export, ls/mv/rm/mkdir, lock/unlock, log, propset
DEMO #2