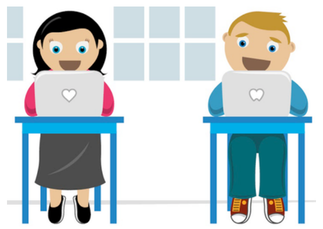
Automated Unit Testing

Code Sharing (Simple Team Development)



- Goal
  - Share your code with others (e.g., me), so everyone can build and run your code in the same way.
- Challenge
  - Different people use different environments (e.g., OSes, IDEs).
  - How can we make sure that everyone
    - follows the same directory structure,
    - imports the same set of JAR files (libraries) with right version numbers, and
    - run the same set of test cases?

- It is too time-consuming and error-prone to
  - Write a document that specifies the expected directory structure, JAR files and test cases to be used.
  - Have everyone meet these expectations *manually* on their environments.
- Automated build
  - Intends to satisfy those expectations *automatically* (not manually)
  - A few well-known tools: Ant, Maven, Gradle, etc.

Automated Build with Ant

- Use Ant (http://ant.apache.org/) to build all of your Java programs in every HW.
  - Turn in *.java and a build script (e.g. build.xml).
    - Turn in a ***single*** build script that
      - configures all settings (e.g., class paths, a directory of source code, a directory to generate binary code),
      - compiles all source code from scratch,
      - generates binary code (*.class files), and
      - runs compiled code
    - DO **NOT** turn in byte code (class files).
    - DO **NOT** use any other ways for configurations and compilation.
      - DO NOT set up CLASSPATH and other paths manually with a GUI/IDE
      - DO NOT set up directory structure manually with a GUI/IDE
      - DO NOT click the “compile/run” button manually on an IDE

- Goal: Fully automate configuration and compilation process to
  - Speed up your configuration and compilation process.
  - Avoid potential human errors in your configuration and compilation process.
  - Make it easier for other people (e.g., code reviewers, team mates) to understand and run your project.

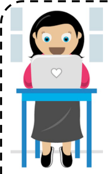
- To grade your work, I will simply run your build script with the “ant” command (on my command-line shell) in the directory where your build script is located.
  - You can name your build script as you like.
    - No need to name it build.xml.
      - I will type: ant -f abc.xml
  - If the “ant” command fails, I will **NOT** grade your work.

HW 1-1: Step 1

- Implement `Calculator`
  - In the package: `edu.umb.cs680.hw01`
- Follow the expected directory structure.
  - `<proj dir>/src/edu/umb/cs680/hw01/Calculator.java`
  - `<proj dir>/bin/edu/umb/cs680/hw01/Calculator.class`
- Use Ant to build and run `Calculator`
  - Set up the directory where `Calculator.class` is placed.
    - `<proj dir>/bin/edu/umb/cs680/hw01`
  - Set up CLASSPATH
    - `<proj dir>/bin`
  - Compile `Calculator.java` and generate `Calculator.class` to `<proj dir>/bin/edu/.../hw01`
    - Use `<javac>` task
  - Run `Calculator.class`
    - Use `<java>` task to run Calculator’s main()

Important Notes

- Use the ANT_HOME and PATH environment variables to specify the location of the “ant” command (i.e., ant.sh or ant.bat)
  - ANT_HOME
    - Reference the top directory of an Ant distribution
  - PATH
    - Reference the location of the “ant” command
  - c.f. http://ant.apache.org/manual/
- You can assume my OS configures ANT_HOME and PATH properly.



Your Machine/OS

Environment variables:
ANT_HOME=C:\ant
PATH=%ANT_HOME%\bin; …

My Machine/OS

Environment variables:
ANT_HOME=~/code/ant/
PATH=${ANT_HOME}/bin:…

Shared across two machines

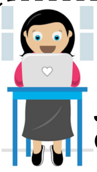
The “ant” command starts running based on your env variables and executes your build script on your machine.

Your build.xml

The “ant” command starts running based on my env variables and executes your build script on my machine.

- Never set up these absolute paths in your build script.
  - Do it in your OS setup. Assume I do the same properly on my OS.
- Keep your build script OS-independent.

- Never include absolute paths in your build script.
  - My OS would not be able to recognize them.
  - Always use relative paths.
- Keep your build script OS-independent.



Your Machine/OS

Java code:
C:\java\edu\umb\...\hw01\foo.java

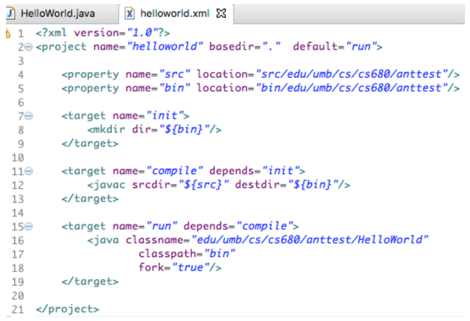
My Machine/OS

Your build.xml
<project name="calculator” **basedir=”C:\java”** …>

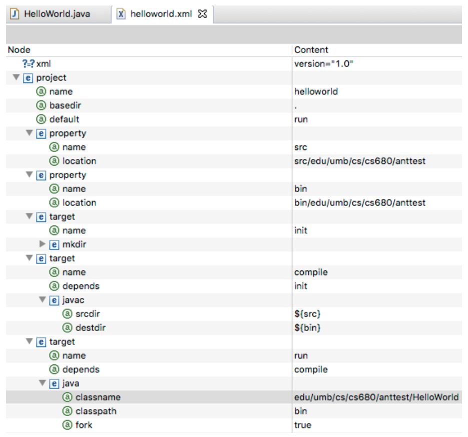
Your OS can recognize this absolute path, but mine cannot.
Correct it to: **basedir=”.”** and place your build script at C:\java

Ant in IDE

- You can use Ant that is available in your IDE (e.g. Eclipse).
  - However, I will run your build script on a shell.
  - Make sure that your build script works on your shell.



```
<?xml version="1.0"?>
<project name="helloworld" basedir="."  default="run">

    <property name="src" location="src/edu/umb/cs/cs680/anttest"/>
    <property name="bin" location="bin/edu/umb/cs/cs680/anttest"/>

    <target name="init">
        <mkdir dir="${bin}"/>
    </target>

    <target name="compile" depends="init">
        <javac srcdir="${src}" destdir="${bin}"/>
    </target>

    <target name="run" depends="compile">
        <java classname="edu/umb/cs/cs680/anttest/HelloWorld"
                 classpath="bin"
                 fork="true"/>
    </target>

</project>
```



Node	Content
?-? xml	version="1.0"
project	
name	helloworld
basedir	.
default	run
property	
name	src
location	src/edu/umb/cs/cs680/anttest
property	
name	bin
location	bin/edu/umb/cs/cs680/anttest
target	
name	init
mkdir	
target	
name	compile
depends	init
javac	
srcdir	${src}
destdir	${bin}
target	
name	run
depends	compile
java	
classname	edu/umb/cs/cs680/anttest/HelloWorld
classpath	bin
fork	true