Welcome!

Administrivia

- Resources
  - Web page
    - http://faculty.washington.edu/rmoul/java/programming
  - Email
    - rmoul@uw.edu
  - Phone
    - (253) 657-9568
- Schedule
  - Lecture schedule
  - Office hours
  - I always respond to e-mail

Highlights From Syllabus

- 9 assignments
  - All assignment evenly weighted
  - Must complete all assignments
- Must attend 8 out of 10 lectures

Grading - Unforgivable Sins

- NullPointerException
  - Easily corrected
    - Stack trace identifies the exact line of code
    - This line must access a reference that is null
  - Common causes
    - Un-initialized variables, especially member variables
    - Chained methods calls, intermediate methods returns null
- Caught and ignored exceptions
  - Every caught exception must either
    - Be handled, the originating problem resolved
    - Generate a stack trace
  - Absent this - very difficult to diagnose/debug

Topics Covered

- Ant
- Junit & Inner Classes
- I/O
- Collections
- Object Serialization
- JavaBeans
- Java Database Access
- Simple Network Programming
- Threads
- Review

Using Ant

Java 2 Applications Programming

Objectives

- Learn how to build, package and distribute Java programs
- Introduction to rudiments of XML
- Introduction to Ant build tool

XML Basics

- XML (eXtensible Markup Language)
- Text files
- Tree-structured
- Composed of *tags* similar to HTML tags
  - HTML 4.0 is a dialect of XML
- In XML, you can invent your own tags

XML Tag Syntax

- The content of an XML document is structured by means of tags
- All tags must have an opening and closing delimiter
- Tags can be nested to achieve the tree structure

```
<tagName> content </tagName>
```

XML Tag Syntax (cont’d)

- **Tags can have attributes**

```
<tagName attribute="value">
```
content
```
</tagName>
```

- **Some tags are self-contained**

```
<tagName attribute="value"/>
```

XML Tag Syntax (cont’d)

- **Tags can be nested – tags can form the content of other tags**

```
<tagName attribute="value">
   <nestedTag attribute="value"/>
</tagName>
```

Introduction to Ant

- What Ant is:
  - 100% Java-based build tool
  - XML build file language
  - Cross-platform – anywhere Java runs, Ant runs
  - Designed to build and deploy Java applications
  - A system that understands Java-related tasks such as creating javadoc and jar files

Introduction to Ant (cont’d)

- What Ant is not:
  - An XML scripting language
  - A replacement for make
    - Make works well for C/C++ development and many other tasks, but is cumbersome and inefficient for Java
    - Ant knows nothing about C/C++ compilers, include files, etc.

Origin of Ant

- Created by James Duncan Davidson in 1998
- Created to build and deploy Java Servlet 2.1 reference implementation (Tomcat) on Solaris, Windows, MacOS and Linux
- Needed a build tool that worked identically on all platforms

Ant Building Blocks

- The Project
- Targets
- Tasks
- Data elements
  - Properties
  - DataTypes

Ant Projects

- The Project
  - Root element of the build.xml
  - One project per build.xml
  - The project tag surrounds all other elements

Ant Targets

- Large-scale goals of the build
- Examples:
  - compile
  - build
  - clean
- You define your own

Ant Tasks

- Smallest building blocks of Ant
- Define operations performed by a target
  - Perform the actual work
  - All operations are wrapped in tasks
- Many tasks are provided by Ant
- You can create your own

Using Ant

- Key Concepts
  - Projects
    - One per build file
  - Properties
    - Simple name value pairs
  - Targets
    - Sub-elements of the project which may be built separately
    - Each project has a default
    - Dependencies
      - Targets may be dependent on other targets
      - A target builds those it depends on first

Using Ant

- Key Concepts (cont.)
  - Tasks
    - Commands that may be executed
  - Filesets
    - Simple to complex file selection specifications

project

- **Attributes**
  - `name`, **project name**
  - `default`, **the default target (required)**
  - `basedir`, **base directory for path calculations ‘.’ by default**

```
<?xml version="1.0"?>

<!-- =================================================================== -->
<!-- Build file for 'JBoss Test'.                                        -->
<!-- =================================================================== -->
<project name="JBossTest" default="default" basedir=".">
   .
   .
   .
</project>
```

`property`

- **Attributes**
  - `name`, **property name (required)**
  - `value`, **the default target**

  or
  - `file`, **file to load properties from (required)**
- **Expansion**
  - **Notation:** `"${<property_name>}"`

property

```
  <!-- ================================================================== -->
  <!-- Initialization of all property settings                           -->
  <!-- ================================================================== -->
  <target name="init">
    <!-- Load the platform specific properties -->
    <property file="env.properties"/>

    <!-- Set the base directories -->
    <property name="build.dir"        value="build"/>
    <property name="lib.dir"          value="lib"/>
    <property name="src.dir"          value="src"/>
    <property name="web.dir"          value="web"/>
    <property name="dd.dir"           value="dd"/>
    <property name="deploy.dir"
              value="${JBOSS_HOME}/server/default/deploy"/>
    <property name="jboss-client.dir" value="${JBOSS_HOME}/client"/>
  .
  .
  .
</target>
```

target

- **Attributes**
  - name, **target name (required)**
  - depends, **targets this target depends on**

```
<!-- =================================================================== -->
<!-- Makes sure the needed directory structure is in place              -->
<!-- =================================================================== -->
<target name="prepare" depends="init">
  <mkdir dir="${build.dir}"/>
  <mkdir dir="${lib.dir}"/>
</target>
```

`fileset`

- **Attributes**
  - `dir`, **root directory for paths (required)**
  - `includes`, **files to include**
  - `excludes`, **files to exclude**
- **Nested Elements**
  - `include`
    - **name attribute identifies a file to include**
  - `exclude`
    - **name attribute identifies a file to exclude**

Include and Exclude Patterns

- **The syntax for wildcard characters in *`include`* and *`exclude`* attributes**
  - `*` **matches zero or more characters**
    - `*.java` **matches** `Account.java` **and** `Person.java`
  - `?` **matches one character**
    - **File?.java matches FileA.java and FileB.java, but not FileTest.java**
  - `**` **matches zero or more directories**
    - `/xml/**` **matches all files and directories under** `/xml/`
    - `**/*.java` **matches all java files in all directories from the project root down**

fileset

```
<!-- ================================================================== -->
<!-- Assembly of the client part of the application                     -->
<!-- ================================================================== -->
<target name="client" depends="compile">
  <jar jarfile="${build.dir}/client.jar">
    <fileset dir="${lib.dir}">
      <include name="test/client/**"/>
      <include name="test/ejb/**"/>
      <exclude name="test/ejb/*Bean.*"/>
    </fileset>
  </jar>
</target>
```

Tasks

- Tasks of particular interest
  - `mkdir`
  - `copy`
  - `delete`
  - `javac`
  - `jar`
  - `ant`
  - `antcall`

mkdir

- **Attributes**
  - `dir`, **directory to create (required)**

```
<!-- ============================================================ -->
<!-- Makes sure the needed directory structure is in place        -->
<!-- ============================================================ -->
<target name="prepare" depends="init">
  <mkdir dir="${build.dir}"/>
  <mkdir dir="${lib.dir}"/>
</target>
```

copy

- **Attributes**
  - `file`, **file to copy (required, or nested** `fileset`**)**
  - `tofile`, **destination (required, or** `todir`**)**
  - `todir`, **destination (required, or** `tofile`**)**
- **Nested Elements**
  - `fileset`

```
<!-- =================================================================== -->
<!-- Deploy the ear file                                                 -->
<!-- =================================================================== -->
<target name="deploy" depends="ear,client">
  <copy file="${build.dir}/JBossTest.ear" todir="${deploy.dir}"/>
</target>
```

delete

- **Attributes**
  - `file`, **file to delete (required, or** `dir`**)**
  - `dir`, **directory to delete (required, or** `file`**)**
- **Nested Elements**
  - `fileset`

```
<!-- =================================================================== -->
<!-- Removes disposable files and directories                            -->
<!-- =================================================================== -->
<target name="clean" depends="init">
  <delete dir="${lib.dir}"/>
  <delete dir="${build.dir}"/>
</target>
```

javac

- **Attributes**
  - `srcdir`, **location of source (required or nested `src` elements)**
  - `destdir`, **destination for class files**
  - `includes`, **files to include**
  - `excludes`, **files to exclude from compilation**
  - `classpath`, **classpath to use**
- **Nested Elements**
  - `src`, **path attribute specifies a source directory**
  - `include`, **name attribute identifies files to include**
  - `exclude`, **name attribute identifies files to exclude**

javac

```
<!-- =================================================================== -->
<!-- Compile the classes                                                  -->
<!-- =================================================================== -->
<target name="compile" depends="prepare">
  <echo message="Classpath is: ${classpath}"/>
  <javac srcdir="${src.dir}"
         destdir="${lib.dir}"
         includes="**/*.java"
         classpath="${classpath}"/>
  <rmic base="${lib.dir}" classname="test.client.CallingCardImpl"
        stubversion="1.2"/>
</target>
```

`jar`

- **Attributes**
  - `jarfile`, **file to create (required)**
  - `basedir`, **directory to obtain files from**
  - `includes`, **files to include**
  - `excludes`, **files to exclude**
  - `manifest`, **manifest file to use**
- **Nested Elements**
  - `fileset`, **files to include, may be a** `zipfileset`

Manifest file

- Contains attributes about the contents of the jar file
- Executable jar file, manifest identifies the “main class”

```
Main-Class: com.scg.domain.Client
```

jar

```
<!-- =================================================================== -->
<!-- Assembly of the client part of the application                      -->
<!-- =================================================================== -->
<target name="client" depends="compile">
  <jar jarfile="${build.dir}/client.jar" manifest="manifest.txt">
    <fileset dir="${lib.dir}">
      <include name="test/client/**"/>
      <include name="test/ejb/**"/>
      <exclude name="test/ejb/*Bean.*"/>
    </fileset>
  </jar>
</target>
```

jar

```
<!-- =================================================================== -->
<!-- Assembly of the client part of the application                      -->
<!-- =================================================================== -->
<target name="client" depends="compile">
  <jar jarfile="${build.dir}/client.jar">
    <fileset dir="${lib.dir}">
      <include name="test/client/**"/>
      <include name="test/ejb/**"/>
      <exclude name="test/ejb/*Bean.*"/>
    </fileset>

    <manifest>
      <attribute name="Built-By" value="${student.name}"/>
      <attribute name="Main-Class" value="${main.class}"/>
    </manifest>

  </jar>
</target>
```

ant

```
<!-- =================================================================== -->
<!-- Assembly of the classes into a jar file                             -->
<!-- =================================================================== -->
<target name="jar-sub"
        description="Assembles the subproject classes into a jar file">
  <ant antfile="subproject/build.xml" target="jar" >
    <property name="classes.dir" value="classes"/>
  </ant>
</target>
```

antcall

```
<!-- ============================================================ -->
<!– Subprojects                                                    -->
<!-- ============================================================ -->
 <target name="worker">
  <ant dir="Assertions/AssertEx" target="${target.name}"/>
  <ant dir="Base64/Base64Ex" target="${target.name}"/>
  <ant dir="Classloader/ClassLoader" target="${target.name}"/>
  <ant dir="Classloader/ImageLoader" target="${target.name}"/>
  <ant dir="Collections/TreeSet" target="${target.name}"/>
</target>

<!-- ============================================================ -->
<!-- Assembly of the classes into a jar file                       -->
<!-- ============================================================ -->
<target name="jar"
        description="Assembles the classes into the jar file">
  <antcall target="worker">
    <param name="target.name" value="jar"/>
  </antcall>
</target>
```

Running Ant

- `ant [option [option…]] [target [target…]]`
  - If target is not specified the default target is built
- most used options:
  - `-help`
  - `-projecthelp`
  - `-version`
  - `-quiet | verbose`
  - `-buildfile` *`filename`* (defaults to build.xml)
    `-f` *`filename`*
  - `-D`*`property=value`*