Object-Oriented Programming & Design

CSCI 4448
University of Colorado
Fall 2002

David Leberknight
Ron LeMaster

These course notes are also available on-line, in .pdf format, free of charge:
http://www.SoftwareFederation.com/cs4448.html

David Leberknight

David Leberknight is a Principal Partner with Software Federation, Inc., based in Boulder, Colorado. Dave works mostly as a consultant specializing in object-oriented software systems. He has been a computer programmer since 1980, working on a wide variety of projects. Dave holds an M.S. degree in Computational Linguistics from Carnegie Mellon University, and a B.S. degree in Electrical Engineering from Cornell University.

Dave can be reached for further training and consulting through:

The Software Federation, Inc.
DavidLeberknight@gmail.com

Ron LeMaster

Ron LeMaster is a Principal Consultant with BEA Systems. He has over twenty years experience as a professional software developer, and currently is focused on Enterprise Java Beans applications. Ron holds an M.S. degree in Computer Science from the University of Minnesota, and a B.S. degree in Mechanical Engineering from the University of Colorado.

Ron can be contacted at:

4462 Wellington Rd.
Boulder, CO 80301
ronlemaster@earthlink.net

Course Objectives

The goal is to build software systems of the highest quality. We will study object-oriented design, since it is perhaps the best way to achieve this goal. We will look at the role that OO design plays in the big picture of software development, and we will look at successful architectures and contemporary technologies that will help us to implement our designs.

We will cover a wide range of topics, including requirements analysis, the Unified Modeling Language (UML) for visual object modeling, design & design patterns, implementation in the Java programming language (with just enough C++ to be able to understand the examples in the *Design Patterns* book), system architecture, middleware technology, databases, XML, and a little project management. All programming assignments will use Java.

It is expected that students will have intermediate programming skills, with detailed knowledge of one or more programming languages (preferably C, C++, and especially Java). Prior knowledge of Java is not required, but students who do not have any Java skills will have to do a little independent work on the side to come up to speed on the fundamentals. Typical students will be Computer Science seniors who are already able to write well-structured, usable programs. Students will learn the object-oriented paradigm, and gain an appreciation for why some designs are better than others. Experience is required to really master the art of OO design, but this course should provide a solid foundation needed for further study and for working on object-oriented projects.

Requirements

- It is your responsibility to look at the course web site at least once per week to see the announcements and reading assignments which will be posted there:

  http://www.SoftwareFederation.com/cs4448.html

- Every enrolled student must send an email to both Dave & Ron a.s.a.p. (see the course web site for mail-to links). Email will be used for class announcements.
- The subject line for all emails to the instructors or the T.A. (all semester) must begin with "CSCI-4448". Each email sent to an instructor that does not have CSCI-4448 as part of the subject will result in a deduction of 5 points from the semester total.
- Students are encouraged to ask the instructors short questions via email.
- It is always OK to visit office hours without an appointment (first come, first served).
- Regrading: the instructors reserve the right to deduct extra points from a student's work if they feel the student is making frivolous requests for regrading.
- The instructor's office hours will be posted on the course web site as soon as they are known.
- All Java assignments must be written using versions of the Java JDK 1.1.8 or 1.3.1. Any Java applets (if assigned) must compile and run with the JDK 1.1.8. (so that they run on most commercial browsers). Other programming assignments will require using the JDK 1.3.1.
- Any violation of CU's honor code (cheating, etc.) will result in an F and a note to the Dean.

Written Assignments

All written work must be submitted in hard copy form, either typed or printed in ink. Graphics may be hand-drawn, but use of a computer-based drawing tool is preferred (see the course web site for some suggestions). Hand-drawn graphics should be drawn with a template, rather than free-hand. The instructors reserve the option of rejecting any hand-drawn work that is illegible.

All homework assignments are due at the *beginning* of class on the due date. The student's name and the date of submission should be printed at the top of all submitted work. Late homework will not be accepted unless arrangements have been made with one of the instructors at least two days prior to the assignment's due date, or the student can produce a documented medical excuse.

All assignments are to be worked on *individually* unless otherwise specified (some assignments will be specified to be worked on in teams of two).

The instructor reserves the option of rejecting assignments that appear not to be the original work of the student submitting it, or that is not produced in a professional manner. Submitting work that is not that of the student's will result in him or her failing the course, with a letter sent to the dean.

Programming Assignments

All programming assignments shall be submitted as a single jar file. This jar file is to be sent to the instructors and to the TA via email before class begins on the due date. The following instructions specify how your build environment should be structured and your jar file should be built. *It is important that you follow these instructions carefully.* Any work that does not conform to this scheme will be penalized 20%, off the top!

The name of the jar file shall be **HW<n>.jar** where **<n>** is the number of the assignment.

Each Java package name must begin with **csci4448.hw<n>,** where **<n>** is the number of the assignment (e.g.: hw2's Foo class must be named: **csci4448.hw2.Foo**).

Project Directory Structure

You should have a base directory for the entire homework project (e.g.: **hw2**). This directory **MUST** contain a file with the name **AAA-<last name><first initial>.txt** for each member of the team that worked on the assignment (e.g. AAA-leberknightd.txt). The file(s) can be empty. They are just there so that the grader can tell who the jar file belongs to.

Under your base directory you should have the following three directories:

src
: This directory is the root of all your source packages. Only the source files that go into your final project should be in this directory. There should be no backup, temporary, or unused source files under this directory

classes
: This directory is the root of all your class file packages.

api
: This is the directory into which all your Javadoc will go.

Here is an example of how this might look. The base project directory is **hw2.**

```
hw2
  src
    csci4448
      hw2
        video
          <java source files for the csci4448.hw2.video package>
        customer
          <java source files for the csci4448.hw2.customer package>
  classes
    <the compiler will place class files under this directory>
  api
    <the javadoc utility will put files under this directory>
```

Building Your Code

You can put your build/make/ant files (or JBuilder project file) in the base project directory, or some other directory of your choosing, other than the three directories listed above. Set up your build environment (JBuilder project properties) to find the source files under the src directory, and to put the compiled class files inside the classes directory.

The Jar Command
The jar command will produce the proper jar file when executed from the base project directory. Make sure you do this after you have created the javadoc files, so they will be included in the jar file. Be sure that these directories contain only the final version of your source code & classes.

```
c:\jdk1.3.1\bin\jar cfv hw2.jar src api -C classes .
```

Be sure to make the "-C" uppercase, and include the "." at the end of the command, separated by a space from the parameter before it. In other words, use this syntax exactly.

The command shown above assumes you **`jar.exe`** is in the directory **`c:\jdk1.3.1\bin`**. If you have installed the Java JDK in some other directory than **`c:\jdk1.3.1`**, change the command appropriately.

Note: If the jar command has run properly, if you extract the contents of the jar file, you will see the AAA-<last name><first initial>.txt file(s), the src and api directories, but NOT the classes directory; instead you will see the csci4448 directory, containing the sub-package directories, and the compiled class files. This is important, since Java assumes that jar files contain compiled class files at the highest possible level in the jar file's internal directory structure.

Javadoc
Some assignments will require that you submit documentation produced by the **`javadoc`** utility. The following example command will produce the proper javadoc files for the source packages listed on the command line (in this case, **`csci4448.hw2.video`** and **`csci4448.hw2.customer)`**. It should be executed from the base project directory.

The javadoc files will be created in a directory named **`api`**, which you must create directly under the base project directory before you execute the command. If your Java JDK is located somewhere other than **`c:\jdk1.3.1`**, modify the command line accordingly.

```
c:\jdk1.3.1\bin\javadoc -private -d api -sourcepath src
csci4448.hw2.video csci4448.hw2.customer
```

This command will create javadoc for the packages **`csci4448.hw2.video`** and **`csci4448.hw2.customer`**. If you have other packages in your solution, be sure to list them on the command line. Check out the results of running this command by going to the **`api`** directory and opening the file **`index.html`** in a web browser. (You can do this on a Windows machine by double-clicking on it in the Explorer.) You should see something whose form looks very much like the API for the Java JDK.

Readings & Grading

Required Reading:

- *UML Explained* by Kendall Scott, Addison-Wesley 2001 <ISBN 0-201-72182-1>
- *Design Patterns* by Gamma, et al., Addison-Wesley 1995 <ISBN 0-201-63361-2>
- There are many good Java books & free documentation on-line (see the course web site for links)… You will be tested on your knowledge of Java, but there is no required Java reading.

Recommended Reading:

- The Java Programming Language - Third Edition by Arnold, Gosling & Holmes Addison-Wesley 1998 <ISBN 0-201-70433-1>

Grading:

- There will be 1000 points for the semester, distributed as follows (subject to change):
  - Mid-term Exam – 200 pts
  - Final Exam – 200 pts
  - Homework Assignments – 600 pts

Table of Contents