JavaGuideLines

Overview

- Recall that all homeworks are to be done in pairs if possible. This directive includes both expository problems and programming problems.
- In Java, the solution to each programming problem must be placed in a separate file using the name (such as `ObjectList.dj1` given in the problem. Similarly, the corresponding JUnit test file should be placed in a separate test file (such as `ObjectListTest.dj1` Please do not split composite class definitions across files unless specifically directed to do so; it makes them very hard to read.

Formatting Expository problems

- A few Java homework problems may be formulated as conventional "free response" questions that require a short written answer. Submit each solution to free response problems as a text (ASCII) file named `<prob>.txt` where `<prob>` is the number of the problem. You can create this text file using your editor of choice (*e.g.*, NotePad on Windows machine, TextEdit on a Mac, or **emacs** on a Linux machine).
- At the top of each expository solution file, please put a header with the assignment number, your name and e-mail address, and your partner's name and e-mail address, like this:

```
COMP 211 HW #01
Christopher Warrington <chrisw@rice.edu>
Gregory Malecha <gmalecha@rice.edu>
```

Programming problems

- Nearly all of your homework problems will be programming problems.
- Over half (60%) of your grade on programming problems depends on writing correct code (30%) and comprehensive test suites (30%).
- Another major component of your grade (30%) is your program style, primarily how well you use OO design principles in writing your code.
- The remaining 10% is based on your documentation. Explicit documentation in Java is less critical that it is in Scheme because Java is object-oriented and statically typed. Most of your Scheme documentation involved defining and declaring program types and showing your recursive program decompositions (templates). In Java, the static type declarations and OO structure of your program reveals the same information. Only contracts (HTDP purpose statements) for classes and class members must be provided as documentation.
- All of the non-test code for a typical problem should be placed in a single file. The accompanying test code goes in a separate file. Use the names specified in the problem description. Typically, the test file for a problem placed in file . (where is dj0, dj1, or java) will be placed in a file called `...Test`. (For very large projects, DrJava can perform the "Test" command more quickly if all JUnit test files end with the character string "Test".)
- Large programs should be divided into separate files as suggested in the problem description.
- At the top of the solution file for a particular problem, please put a header with the assignment number, your name and e-mail address, and your partner's name and e-mail address, like this:

```
/* COMP 211 HW #07, Problem 4
 * Christopher Warrington <chrisw@rice.edu>
 * Gregory Malecha <gmalecha@rice.edu>
 */
```

Submitting your homework

Please submit your homework using the OwlSpace submission facility.

Java Coding Guidelines

- Write your program in OO style, using design patterns when appropriate.
- Document each class with a short description of what the class does. This documentation should immediately precede the class and be formatted using **javadoc** brackets /** ... */.
- Document each class member with a short description specifying what that field or method does. The documentation for a Java method should look almost exactly like the purpose statement for a Scheme function. This documentation should immediately precede each class member and be formatted using javadoc brackets /** ... */.
- If a class member is inherited from a superclass, no additional documentation is necessary unless the member definition has some unexpected characteristics. Hence, in a composite hierarchy, methods defined over the entire hierarchy are typically documented only in the root class of the hierarchy. For examples, see the sample programs posted on the main page accompanying the class lectures on OO design.