4 Moving to Eclipse IDE

For this assignment you can only use the Java language features we have covered in class or in the labs. The details are described in a separate document.

Portfolio Problems

Convert the following programs into Eclipse projects:

Problems:

1. Problem 15.8 on page 175
2. Problem 15.11 on page 176
3. This problem continues the work on mobiles we have started during the lectures. The file **mobile-methods-lecture.java** contains the data definitions, examples of data, and the method `totalWeight` we have designed in class.

   Design the method `draw()` that consumes a `Canvas` and a `Posn` that represents the point where the top of the mobile will hang. The method draws the mobile with black lines for the struts, and for the hanging lines. For a simple mobile, there should be a disk of the appropriate color and with the size proportionate to its weight shown at the end of the line.

Note: Translating the test cases from the ProfessorJ syntax to the syntax used by the *tester* library can be tedious and time consuming. The *javalib* web site provides a very simple, possibly not 100 % accurate, Java program that reads a ProfessorJ file and produces a new file where the test cases have been converted to the *tester* library style. The web site includes instructions for how to do it. Feel free to try it — but you are not required to do so.

If you do use the converter program, please, report any problems on the wiki. (Thanks)

Pair Programming Assignment

4.1 Problem

Covert the *Shooting Star* game so it runs in Eclipse. Add test cases for the situation where the expected outcome is one of several possible values (see a note below).

4.2 Problem

Meet with your partner and look at the two games each of you have designed with your previous partner.

A. Choose the game that you like better. Implement the game in Eclipse.

   If you think the game is too complex, feel free to simplify it.

B. Draw a complete class diagram for your game — including not only all fields, but also all methods defined in each class, or declared in each interface. Do this on paper — we will ask for it during the *Project Review*.

C. Write a user's guide to your game. It should be sufficiently clear so the grader will understand how to play your game.

   Add this as a block comment at the beginning of your game. Bring a printed copy of the *User's Guide* to the *Project Review*.

Note: Remember that we care a lot about the program design. It is better to design a simpler game, with good comments and a complete test suite, than to design a complex unreadable program that just *runs*.
Note: With the *tester* library you can test whether the actual value is one of several expected values, or whether it is within a given range of values. See examples of the use of the `checkOneOf` and `checkNumRange` in the *tester* User's Guide under **Random Choice** and **Range Check**.