CS1110, Fall 2010. Preparing for Prelim 1
Thursday, 7 October. 7:30–9:00PM, Olin 155 (students with last names A-K) and Olin 255 (L-Z)
Review session: Sunday, 3 Oct. 1:00-3:00PM, in Phillips 101.

This handout explains what you have to know for the first prelim.
We have posted several previous CS1110 prelims to the course website. File KeyConcepts.doc summarizes the key concepts and syntax of Java. Download them and print them if you want. Try out your solutions on DrJava — this gives you fluency and the compiler helps you correct your own work. (Why take programming exams on paper? Many company interviews consist of the candidate being asked to write and defend their code at a whiteboard!) You can also try to reproduce the programs we have written in lecture, all of which are also posted on the course website.

Classes and objects

You need to know and be able to use the concepts that we have taught through the lecture on 28 September. Recursion will not be covered on the prelim. You need to know definitions (like what is the assignment statement and how is it executed) as well as the syntax of the Java constructs that we use often. We don't deduct points for minor lapses on syntax, but complete misunderstandings of syntax will be penalized.

You may be asked to write a class definition.

You will be asked to write a function that manipulates strings in some way.

Terms and their meanings

Below, we summarize the terms you should know. You should be able to state the definition of a term like "assignment statement" or "parameter" clearly and precisely. For example, for a Java statement, you should know its syntax and how to execute it. This information is given in boxes at the top of appropriate pages in the textbook.

• **Expressions**: Types **int**, **double**, **boolean**, **char** (their ranges and basic operations). Casting between primitive types. Narrower type, wider type. You should know these basic methods of class String, as discussed in Lab 04: length(), charAt(i), substring(i), and substring(i,j). If any other functions are needed, we will give their specifications. You will be asked to write a method that manipulates strings.

• **Variables**: variable, declaration of a variable, assignment statement**.** Four kinds of variable: field, static variable, parameter, and local variable; know where each is declared and what its scope is. See last page of this document.

• **Methods**: Three kinds of method: procedure, function, constructor. Syntax of a method definition. Parameter of a method. Local variable of a method. Scope of a parameter and a local variable. Be able to write a simple method.

• **Method calls**: How to call each kind of method. Argument of a method call. Restrictions on arguments based on the type of the corresponding parameter. Frame for a method call. Be able to execute a method call using the 4 steps: draw the frame, store argument values in the parameters, execute the method body, erase the frame.

• **If-statement and if-else statement.** Their syntax and how they are executed.

• **Block**. Its syntax and how it is executed. It is just a statement of the form "{ … }".

• **Classes**. What is a class? Class definition. Instance (folder, or object) of a class. The name of a folder. Components: fields and methods. Static and non-static components of a class (where do they go). The new-expression and what it is used for. You should be able evaluate a new-expression by hand, drawing the new object, executing the constructor call, and yielding the name of the new object.

• **Subclasses**. How to define a subclass. Inheritance and overriding. Constructors in a subclass: the first statement must be a call **super**(…); on a constructor of the superclass or a call **this**(…) of another constructor in this class. If missing, **super**(); is used. You should be able to draw an object of a class or subclass, given the class definition, including the partition for Object, the superest class of them all.

What **this** means (it evaluates to the name of the object in which it occurs). Keyword **super** does the same thing but only for the partitions above the one in which it occurs.

• **Bottom-up (overriding) and inside-out rules** for determining what variable or method a reference like b or m(…)

refers to. See text, pp 144, 83, 109.

• **Wrapper classes and class Vector.** You should know what these are for and how to use them. We will provide specifications of any methods of these classes that you will have to use.