Final Exam Review Sheet

This test will be roughly the same length as the midterm with 10 questions. The questions will have a format similar to that on the quizzes from the second half of the semester. You will be able to open the API and use that as a reference, but you won't be able to use Eclipse. Below is a list of topics we have covered and what I would expect you to be able to do with each. I can't promise that this list covers everything on the exam, but I try to stick to it.

Note that you need to be able to write reasonable Java code and understand it enough to trace through it. Unlike a compiler, I don't put red underlines under every syntax error. You need to be close enough that I can tell what you are doing and since you will have the API you should know all the right method names. Obviously, the code I provide on the test can help you in constructing your own.

Primitive Types – I don't expect you to know all the primitive types, but you should know the majors ones: int, double, char, and boolean. You need to know when and where to use them.

Reference Types – These are all the types in the libraries or in your code defined by classes or interfaces. You need to know how to declare variables of these types and instantiate new objects. If you can't make a proper Scanner for the exam we have problems.

Methods – Be able to write and call methods. Know the dot notation for calling methods on objects or in classes. Part of writing methods is knowing how to pass arguments into them and return values from them. This also includes the use of local variables in the methods.

Conditionals – Know the syntax and usage of the if statement as well as the boolean operators &&, ||, and !. Be able to use these in code or trace them. I'm not going to ask you to write switch statements, but I can't rule out the possibility that they might appear in code that you have to trace. If they do, they will be fairly straightforward and I won't do anything tricky like leave out a break statement.

Loops – Know the three types of loops in Java as well as how to write/use each one. Understand the difference between pre-check and post-check loops and which Java loops are which type.

Files – Know how to use the File class, how to read a file using a Scanner, and how to write to a file with a PrintWriter. You should also understand conceptually why we like to have programs be able to use files.

Exceptions – Know what these are and why we use them. I'm not going to ask you to write code that involves them. So you can ignore the exceptions if you write any file code. However, if an exception appears in code that you have to trace you need to be able to deal with it. So you don't have to memorize the syntax, but you do need to understand it when you see it and know what it means.

Arrays – You need to understand what arrays are and how to use them in Java. You should know the syntax for using an array because odds are good you will be asked to write code involving one or more. The test will not include multidimensional arrays, only the 1-D variety. That is true for both coding and tracing.

Lists – Know how lists differ from arrays and when you would use a list instead of an array. I'm not going to ask you the relative strengths and weaknesses of LinkedLists vs. ArrayLists because we didn't go into that in detail in class. However, you could be asked to trace or produce code that involves lists.

Inheritance – Know what inheritance is and what it provides us in Java. Be able to trace polymorphic code that takes advantage of the subtyping in inheritance. I'm generally not going to give you problems that require writing multiple classes so it isn't likely you will be writing a full inheritance hierarchy. However, I could ask you to write a class that inherits either from code I provide or something in the libraries.

GUIs – Know the basic way in which GUIs are set up and how you get it so that the user can interact with them. I could ask you to write simple code that produces a GUI. You should understand how inheritance is significant in the GUI libraries.