CSE 143 Midterm 2 Topic List – Nov. 8, 2004

Here is a brief checklist of the additional topics that have been covered since the first midterm and might appear on Monday's exam. You remain responsible for everything earlier in the course, but the second midterm will have somewhat more emphasis on newer topics. You do not need to memorize details of classes like streams and `String` – brief reference information will be included in the test if it is needed to answer questions.

- Programming by contract
  - Preconditions, postconditions, invariants (particularly class invariants)
  - Throwing exceptions – `throw` statement and exception objects; using standard exceptions to signal errors (`NullPointerException`, etc.)
  - `assert` statement
  - Appropriate use of `throw` vs `assert` – when is it appropriate to use each
- Exception handling
  - What happens when an exception is thrown
  - Use of `try-catch` to handle exceptions; order of `catch` clauses
  - Checked vs unchecked exceptions
  - `throws` clause in method headings – when needed; what it means
- Streams
  - Data representation – interpretation of the underlying bits (don't memorize things like bit codes for characters or numeric limits of `int`, `long`, `double`, etc., but do know the basic ideas)
  - Stream model – Java's basic organization of I/O
  - Character streams (`Reader`/`Writer`) vs byte streams (`InputStream`/ `OutputStream`)
  - Opening and closing streams
  - Files and their relationship to streams; basic use of `JFileChooser`
  - Be able to write code to open files and read/write them using `BufferedReader` and `PrintWriter` streams
- Collections – particularly the Java collection classes
  - Key collection interfaces (`Collection`, `List`, `Set`, `Map`) and implementations (`ArrayList`, `HashMap`, etc.
  - Iterators and how to use them
- Simple implementation of lists using arrays
  - Simple array list – use of dynamic allocation to expand the capacity
  - Class invariants for a list; size vs capacity; using arrays, subscripts, etc.
  - Companion iterator implementations and how they are related to the associated containers
- Testing
  - Be able to design tests for data structures like the simple array list
  - You're not responsible for remembering the details of the JUnit framework, but given a list of available tests (`assertEquals`, `assertTrue`, etc.), you should be able to write tests for simple data structures or other objects.
- Anything related to project 2, including files and streams, basic string operations like `indexOf`, `substring`, using `HashMap`s, etc.