CSCI209 2nd Midterm Prep

Similarities to, differences from Python

Streams
- Classifications of stream classes
- How to use
- Justification for design
- Standard streams

Software Development
Development Models – waterfall, iterative, spiral
Prototypes

Testing
Different levels of testing (unit, integration, system, …)
Black-box testing vs. White-box testing
Coverage criteria
JUnit testing framework
- How to write “good” JUnit test cases

Design Principles
Design goals
Open-closed principle
- Open to extension, closed to modification
- Liskov substitution principle
DRY (Don’t repeat yourself)
DRY (Don’t repeat yourself)
Shy code (avoiding coupling)
Tell the Other Guy (Tell, Don’t Ask)
Single Responsibility Principle
Code smells
Metrics to quantify code design qualities
Appropriate solutions
Defending solutions using appropriate terminology and design principles
DRY (Don’t repeat yourself)

Refactoring
Resolving code smells using abstraction

GUI Development
Common components, containers
Layout Managers
Event-driven programming
- ActionListeners, handling events, inner classes, anonymous inner classes, adapter classes
Jar Files
Deploying Java applications
How to create, extract, use

Class path

What I expect from you on the exam:

- To know Java/ OO-programming/design terminology
- To design a solution and be able to defend it
- To be able to read, understand, and write Java programs, with or without documentation
- To be able to write a program (given an algorithm or creating your own algorithm, given a problem) or class
  - Syntax must be very close to correct (correct keywords, punctuation, special characters, variable naming, operations)
  - Since it's on paper, there is some leniency

Suggestions on how to prepare:

- Exam is **terminology heavy**. Make sure you know the terminology (much of it is in the list above).
- Read through slides for vocabulary, review questions, exercises
- Think about the various designs we have discussed in certain situations and what the tradeoffs are to each design.
- Practice reading through programs, tracing through them, and saying what the output should be
  - Review your assignments