CSCI209 2nd Exam Prep

Content Covered

Similarities to, differences between Java and Python

Files

Streams

- Classifications of stream classes
- How to use
- Justification for design

Standard streams (not Java-specific)

Jar Files

Class path

Software Development

- Development Models – waterfall, iterative/spiral, test-driven development

Testing

- Different levels of testing (unit, integration, system, …)
- Black-box testing vs. White-box testing
- Code coverage
  - Coverage criteria
  - Strengths and limitations
- JUnit testing framework
- How to write good unit tests

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
- Dependency Inversion Principle
- Appropriate solutions
- Defending solutions using appropriate terminology and design principles

- DRY (Don't repeat yourself)

Code smells

Refactoring
- Resolving code smells using abstraction

Design patterns
- Composition
- Strategy
- MVC
- Factory

Exam Structure
- Online, timed (70 minutes)
- Open book/notes/slides
- **Not** open internet
- Sections: very short answer, short answer, applied

What I expect from you on the exam:
- To know Java/OO-programming/design terminology
    - Need to be able to communicate with others about your design/implementation
- To focus on the most important/distinguishing characteristics/traits
    - Example: if there is a 4-point question is "What is Java? Explain a benefit of Java.", do not start copying all of the content from any slides you can find about Java. You should know the highlights and be able to summarize what is Java, e.g., "Java is a compiled, statically typed programming language." Followed by "Benefit: Java is portable because it is compiled into bytecode, which is executed by a JVM. Java bytecode can be executed by any machine that has a JVM."
    - Example: if the question is "What are the benefits of using arrays?", do not answer with any variation of "Because arrays are beneficial" or "Because arrays store data." Tell me *why* arrays are good (e.g., "arrays allow us to group data of the same type together, which makes passing them to methods and processing larger amounts of data easier.") You want to make clear that you don't just know the terms but you also understand their meaning and implications.
- To design a solution and be able to defend it
- To be able to read and understand 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)

- To be able to recall the information without looking up every question. If you need to look up answers, you will not complete the exam in time.
- To consider the point value of the question. Since the exam is online, I can't give hints about the length of answers by changing the allocated space. If the question is worth 4 points and is an "essay" question, I'm only looking for a few sentences/bullet points.
- To keep an eye on the time remaining.
- To know the syntax for variable declarations/definitions, if statements, for loops, methods, class definitions, print statements; that the `Object` class defines the methods `equals` and `toString`; basically all the things that we've seen/written many times over.

What I do NOT expect from you on the exam:

- To know the API for Java classes that we have covered/used.
- Perfect essays, complete sentences
    - Example: if the question is, "What are the benefits of using arrays?", I do not expect you to answer with, "The benefits of using arrays are the following: …" I just want to know about what follows "the following".
- To write comments for code unless specifically requested
    - However, write comments if they help you to organize your thoughts.
- To compile your code/solutions

Suggestions on how to prepare:

- Exam is ***terminology heavy***. Make sure you know the terminology (much of it is in the list above). I hope that, with all the practice we've done with using and reviewing terminology, your knowledge of the terminology should be close to automatic.
- 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/behavior should be

Honor Code Rules:

- **No discussion of the exam after you start/take the exam**
    - Nothing about if it was hard or easy or how long it took you or if you studied enough or …
    - NOTHING