# CSCI209 1st Exam Prep

## Content Covered

Java

- Characteristics
  - Statically typed, OO, compiled, …
- Benefits
- How to write/run programs

Syntax, semantics, application of

- Comments, including Javadoc
- Variable declaration (local, instance, static), defining/assigning
- Using primitive types
- Arrays
- Control structures: Conditionals, Loops
- Static fields, methods
- Methods, Overloaded methods
- Command-line arguments
- equals method (vs ==)
- Parameter passing: by value

Object-oriented programming concepts

- Classes, objects, constructors, methods, state/instance variables/fields
- Inheritance – parent/child classes
- Overriding methods
- Encapsulation and black-box programming
- Access modifiers
- Final fields

Standard streams

Packages

Version Control

- Benefits
- Git, GitHub

## Exam Structure

- In-class
- One letter-size page of notes for reference
- Sections: very short answer, short answer, coding

## What I expect from you on the exam:

- To know Java/OO-programming terminology
    - Need to be able to communicate with others about your design/implementation
- To be able to read, understand, write, and debug Java programs, with or without documentation.
- To be able to recall the information without looking up every question—or even multiple questions. If you need to look up answers, you will not complete the exam in time.
- To consider the point value of the question. If the question is worth 4 points, I'm only looking for a few sentences/bullet points.
- To keep track of the time remaining.
- To know the syntax for variable declarations/definitions, print statements, if statements, for loops, methods, class definitions and rules; basically, all the things that we've seen/written several times over.

## What I do NOT expect from you on the exam:

- Knowledge of general (rather than the specific ones we've been talking about so far) software development concepts, *ilities
    - Next exam
- Similarities to, differences from Python
    - Next exam
- To know the API for Java classes that we have not 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 anything along the lines of, "The benefits of using arrays are the following: …" I just want to know 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

## Suggestions on how to prepare:

- Exam is *terminology heavy*. Much of the terminology, you have been using since CSCI111/112; some is new. Make sure you know the terminology (much of it is in the list above). I also hope that, with all the practice we've done with using and reviewing terminology, your knowledge of the terminology should be close to automatic.
- Practice reading through programs, tracing through them, and saying what the output should be
    - Review your assignments
- Read through slides for vocabulary, review questions, exercises

## Honor Code Rules:

**No discussion of the exam after you start/take the exam until after Monday's class**

- Nothing about if it was hard or easy or how long it took you or if you studied enough or …
- NOTHING

## Practice Programming

- Practice in the textbook (let me know how helpful that is)
- Go back to your CS-111 or CS-112 courses and try to write those programs in Java instead. As long as the problems don't require using a dictionary or special [Python] libraries, you should be able to write that code in Java. (We'll get to Java's version of dictionaries soon.)
- Change the FileExtensionFinder to have a static method that takes as a parameter the file name and returns the extension, lowercased.
- Write the Card and Deck classes from the first in-class exercise in Java. Don't look at the code and just translate. Instead, consider if you were to develop the Card class, how would you write it? Think about our steps in development. Make sure to test. Try to write from scratch, not using examples, to start. Then, write a game with the classes, e.g., War.

## Answering Short Answer Questions

- Make it clear that you truly understand the material and are not just using some terms and hope they're in the right order. Think of this like if you were answering an interviewer's questions. They don't know what you "should" know.
    - Feedback to student, "You omitted X." Student says, "Well, of course I know X. It was covered in class." I don't *know* that you know it. Neither will an interviewer.
- In recent semesters, students have included in their answers material that we have not covered (perhaps from prior knowledge or from using AI to supplement their understanding). You are expected to only include material that was covered in class.
- Focus on the most important/distinguishing characteristics/traits
    - Example: if there is a 4-point question "What is Java? Explain a benefit of 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. That Java bytecode can be executed by any machine that has a JVM."
- Be explicit in connecting the dots: For example, if you state that X is a benefit, you then should say how/why that is a benefit and how it is accomplished. It often doesn't have to be a long explanation. It just needs to convince me that you understand why it's a benefit.
    - 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.

- The *limitations* of something are the things that, even if that something is implemented perfectly, it still has limits. For example, a limitation to adopting Java programs is the amount of information/syntax you need to be able to write a simple program. (Although new features are coming!)