CMPSC 101
Data Abstraction
Fall 2020

Exam One Study Guide
Exam Released: Thursday, October 8, 2020 at 3:00 pm
Exam Due: Friday, October 9, 2020 by midnight

Introduction

The exam will be "open notes" and "open book" and it will cover the following materials. Please review the "Course Schedule" on the Web site for the course to see the content and slides that we have covered to this date. Students may post questions about this material during our review session on Wednesday, October 7th.

- Chapter One in DSA, all sections (i.e., "Java Primer")
- Chapter Two in DSA, all sections (i.e., "Object-Oriented Design")
- Chapter Three in DSA, only Section 3.1 and 3.5.1, 3.6.1 (i.e., "Fundamental Data Structures")
- Using the commands in the terminal window (e.g., "`cd`" and "`ls`"); building and running Java programs with Gradle; knowledge of the basic commands for using "`git`" and GitHub.
- Your class notes and the discussion slides available from the course web site.
- Source code and writing for laboratory assignments 1–4 and practical assignments 1–3.

The exam will be a mix of questions that have a form such as fill in the blank, short answer, multiple choice, true/false, and completion. The emphasis will be on the following representative topics:

- Fundamental concepts in computing and the Java language (e.g., definitions and background)
- Practical laboratory techniques (e.g., editing, building, and running programs; effectively using files and directories; correctly using GitHub through the command-line `git` program)
- Understanding Java programs (e.g., given a short, perhaps even one line, source code segment written in Java, understand what it does and be able to precisely describe its output).
- Composing Java statements and programs, given a description of what should be done. Students should be completely comfortable writing short source code statements that are in nearly-correct form as Java code. While your program may contain small syntactic errors, it is not acceptable to "make up" features of the Java programming language that do not exist in the language itself—so, please do not call a "`solveQuestionThree()`" method!

No partial credit will be given for questions that are true/false, completion, or fill in the blank. Partial credit may be awarded for the questions that require a student to write a short answer. You are strongly encouraged to write short, precise, and correct responses to all of the questions.

Reminder Concerning the Honor Code and Code of Conduct

Students are required to fully adhere to the Honor Code and course's Code of Conduct during the completion of this exam. Relevant details about the Allegheny College Honor Code are provided on the syllabus. The Code of Conduct is included in the lab01 repository (`https://github.com/allegheny-computer-science-101-f2020/lab01-cs101f2020/blob/main/conduct.md`).

Detailed Review of Content

The listing of topics in the following subsections is not exhaustive; rather, it serves to illustrate the types of concepts that students should study as they prepare for the exam. Please see the instructor during office hours if you have questions about any of the content listed in this section.

Chapter One

- Basic syntax and semantics of the Java programming language
- How to use "`gradle`" to build, run, and grade all classes in a Java program
- The base types available for primitive variables in Java (e.g., `int` and `char`)
- How to use conditional logic (e.g., `if`) to influence a program's control flow
- How to use arithmetic expressions to calculate statistical values
- Both the correct and incorrect ways in which to swap data values in a Java program
- How to create and use classes and objects in the Java programming language
- How to define and call the methods and constructors of a Java class
- The way in which Java methods accept input and produce output
- The declaration and use of `String`, and arrays
- Type conversion operators and control flow constructs in Java
- Software engineering principles as expressed in Java (e.g., packages and JUnit tests)
- Practical strategies for the correct design, implementation, and testing of a Java program
- How to use multiple-line and single-line comments to document a Java program
- The ways in which the Google style guide supports the creation of easy-to-read Java programs

Chapter Two

- The goals, principles, and patterns of object-oriented design in the Java language
- An understanding of the principles known as abstraction, encapsulation, and modularity
- How to use inheritance hierarchies to create an "is a" relationship between Java classes
- The meaning and purpose of abstract classes and interfaces in the Java programming language
- How to create, catch, and handle exceptions thrown by a method in a Java program
- How to write Java source code that will read in values from a file while handling exceptions
- How to perform casting to convert a variable from one data type to another
- The similarities and differences between widening and narrowing conversions of data types
- The ways in which generics promote the implementation of reusable Java programs
- How to declare and use a generically typed array in a Java program
- How to use generics to implement a general-purpose array reversal program

Chapter Three

- How to use arrays to store primitive and reference variables
- The algorithms for sorting arrays
- The similarities and differences between single- and two-dimensional arrays
- How to declare, store data into, and retrieve data out of arrays
- How to effectively implement and test a sorting algorithm like the insertion sort and bubble sort
- How to use pseudo-random number generators in the method(s) of a Java program
- The meaning and purpose of different techniques for cloning data structures
- The resulting differences between the processes of array assignment and array cloning
- The similarities and differences between "deep" and "shallow" copy of an array