Final Exam Study Guide

Preparing for Final Exam

- Review the learning objectives
- Review the lecture slides
- Review zyBook, and the activities
- Review (and possibly rewrite) the lab exercises
- If you do not fully understand a topic, read the related textbook section
- Attend office hours to ask additional questions/clarifications
- Complete the practice problems

Final Exam Learning Objectives

- Problems, Algorithms, Programs
    - Describe challenges that exist when interacting with a computer.
    - Explain the following tasks: understanding the problem, designing an algorithm, writing a program.
    - Describe the problem that exists for a given scenario/description.
    - Document an algorithm using pseudocode for a given problem.
- Write-Compile-Execute
    - Explain the following steps when programming: write, compile, execute.
- Structure of Java
    - Describe the structure of a Java program and how the following entities relate to each other: classes, methods, statements.
    - Create a Java program for a given algorithm.
    - Explain the importance of documenting and commenting code.
    - Describe the importance of the `main` method.
- Program Errors
    - Identify program errors as syntax/compiler errors, logic errors, or runtime errors.
- Debugging
    - Debug a program to remove syntax errors, runtime errors, and/or logic errors.

- Data Types
    - Explain what data types are.
    - Describe primitive data types in Java.
- Expressions
    - Evaluate expressions containing `int`, `double`, or mixed types addition, subtraction, multiplication, division, and modulo operators.
    - Explain how promotion works in Java.
    - Explain how casting works in Java.
    - Write code containing primitive data types.
- Variables
    - Declare, initialize/assign value to, and use variables in code.
    - Design algorithms that use variables.
    - Trace values of variables through algorithm or a segment of code.
    - Write Java programs that use variables.
- Formatting Text with printf
    - Write code that used the `System.out.printf` to output a formatted string.
- Constants and Class Constants
    - Describe when constants should be used in code.
    - Write code that properly uses constants.
- Math class
    - Use basic methods for `Math` class (`abs`, `min`, `max`, `pow`, `sqrt`).
    - Write and trace code that uses `Math` class methods and constants.
- Random Numbers
    - Write and trace code that uses random numbers (`Random`).
- Equality, Relational, and Logical Operators
    - Evaluate expressions containing equality, relational, and logical operators.
    - Write code containing equality, relational operators, and logical operators.
    - Negate boolean expressions using De Morgan's Law.
- boolean Data Type
    - Evaluate boolean expressions, including short-circuit evaluations.

- Conditionals
    - Trace conditional (e.g., if or if-else) statements and provide output.
    - Write conditional (e.g., if or if-else) statements to perform an operation or produce specified output.
    - Distinguish between when to use each conditional structure.
- Strings
    - Construct new String objects.
    - Write and trace code containing String methods.
- Character Operations from Character Class
    - Write and trace code that uses Character class methods.
- Scanner Next methods
    - Construct new Scanner object for console input.
    - Write code to read user input from the console with nextInt, nextDouble, next, and nextLine methods.
    - Trace code containing Scanner nextInt, nextDouble, next, and nextLine methods.
- Scanner hasNext Methods
    - Write robust code that checks user input using Scanner hasNext methods: hasNextInt, hasNextDouble, hasNext, and hasNextLine methods.
- Scanner as Parameter
    - Write code that contains a Scanner as a parameter.
- Static Methods: Parameters and Return Values
    - State the syntax of a static method.
    - Write a static method.
    - Explain what parameters are and why using them.
    - Define and write method header with parameters.
    - Trace a method with parameter(s) to determine the output.
    - Write a method with parameter(s) for a given programming problem.
    - Describe what return values are.
    - Define and write method header with return type.
    - Write and trace a method with a return value.

- Passing Parameters
    - Describe how parameters are passed to methods based on their types.
- Flow of Control
    - Describe the order in which statements in a Java program are executed.
    - Describe why flow of control is important.
    - Trace code with an understanding of flow of control.
- Method Overloading
    - Explain method overloading and apply it in writing methods with parameters.
- Verify Parameter Values
    - Write code that throws exception with a `throw` statement.
- Returning within a Conditional
    - Write code that returns within a conditional.
    - Trace code that contains multiple return statements.
- while Loop
    - Trace a `while` loop and provide output along with number of times loop body executes.
    - Write a `while` loop to perform an operation or produce specified output.
- do-while Loop
    - Trace a `do-while` loop and provide output along with number of times loop body executes.
    - Write a `do-while` loop to perform an operation or produce specified output.
- Assertions
    - Identify the various assertions in code as being either always true, never true, or sometimes true/sometimes false at various points in program execution.
- For Loop
    - Explain the benefit of using a `for` loop.
    - Describe the structure of `for` loop and flow of control.
    - Trace a `for` loop and provide output.
    - Write a `for` loop to perform an operation or produce specified output.

- Nested Loops
    - Trace nested loops and provide output.
    - Write nested loops to perform an operation or produce specified output.
- Scope
    - Identify the scope of a variable.
- Array Basics
    - Describe the benefits of using arrays.
    - Initialize and construct an array of primitive types.
    - Write and trace code for accessing elements of an array.
    - Trace code for traversing an array.
    - Write code for traversing an array without throwing an ArrayIndexOutOfBoundsException.
- Parallel Arrays
    - Write and trace code for parallel arrays.
- Arrays Class
    - Write and trace code that uses Arrays class, such as equals, toString, copyOf, sort, binarySearch, fill.
- Array Parameters
    - Understand the basics of passing an array as a parameter.
- Modifying Arrays
    - Understand the basics of modifying arrays.
    - Write and trace code that modifies arrays, such as swap array elements, resize an array, sort an array.
    - Describe the process and perform selection sort and insertion sort on an array.
    - Describe the process and perform binary search on an array given a target.
- Returning Arrays
    - Write and trace code with methods that return arrays.
- Array Sizes
    - Write and trace code with perfect size arrays.
    - Write and trace code with oversize arrays.

- for-each Loop
    - Trace a `for-each` loop and provide output.
    - Write a `for-each` loop to perform an operation or produce specified output.
    - Describe the limitations of the `for-each` loop.
- Multi-dimensional Arrays
    - Write and trace code for multi-dimensional arrays.
- Array of Objects
    - Initialize and construct an array of objects.
    - Write and trace code using array of objects.
- File Input (Token-Based Processing and Line-Based Processing)
    - Write and trace code that uses a `Scanner` to read from a file.
    - Write and trace code that uses a `Scanner` to tokenize a `String`
    - Write and trace code for file input using token-based processing.
    - Write and trace code for file input using line-based processing.
    - Describe the difference between token-based and line-based processing.
- File Output
    - Write and trace code for file output.
- throws Clause
    - Describe when a `throws` clause is needed.
    - Write and trace code that uses a `throws` clause.
- Object Basics
    - Describe the difference between a class and an object.
    - Describe the difference between procedural decomposition and object-oriented programming.
    - Describe the concepts of abstraction and encapsulation.
    - Describe the benefits of the use of abstraction and encapsulation.
- Constructors
    - Describe the use of default constructors.
    - Write constructors for new data types.
    - Overload constructors.

- Instance Fields and Methods
    - Write code declaring fields/instance variables in a class that defines a new data type.
    - Write code containing instance methods in a class that defines a new data type.
    - Describe the use of implicit parameter this.
    - Write code containing implicit parameter this.
    - Write code for an encapsulated object type.
    - Describe the use of accessors/getters and mutators/setters
    - Write code for accessors/getters and mutators/setters
- Static Fields and Methods
    - Write code for object with static fields.
    - Write code for object with static methods.
- Object Methods
    - Override toString method for an object.
    - Override equals method for an object.
- Interacting Classes and Object-Oriented Design
    - Design a simple class given the description.
    - Design interacting classes.
    - Describe and explain the four major principles of OOP
- Inheritance and Polymorphism
    - Describe the concepts of inheritance and polymorphism.
    - Describe the benefits of the use of inheritance and polymorphism.
    - Describe the concepts of class hierarchies: superclass and subclass/derived class
    - Write code containing superclass and subclass/derived class
    - Describe the use of implicit parameter this and reference variable super.
    - Write code containing implicit parameter this and reference variable super.
    - Describe the is-a and has-a relationships
    - Describe the difference between overloading vs. overriding
    - Write and trace code that contains overridden and overloaded methods