CS 444 - Quiz 1 Review Suggestions - Spring 2015

last modified: 2015-02-24

- You are responsible for material covered in class sessions and individual assignments; but, here's a quick overview of especially important material related to this upcoming quiz.
- You are permitted to bring into the quiz a single piece of paper (8.5" by 11") on which you have **handwritten** whatever you wish on one or both sides. This paper must include your name, it must be handwritten by you, and it will **not** be returned.
  - Other than this piece of paper, the quiz is closed-note, closed-book, and closed-computer.
- This is a quiz on Java basics that we have covered thus far and that you should have been using in your Java and leJOS programming in this class.

Java basics - general

- Remember that Java is **case-sensitive**; an answer may lose points if it uses the wrong case (for example, if a class name does not start with an uppercase letter, or if you write a Java keyword using all-uppercase).
- Remember that **all** Java methods must be placed within a class, unlike, say, C++.
- In a file with what name does a public Java class go? How do you compile a Java class from the command line? What results from compiling a Java class?
  - What is the relationship between the name of a public Java class and the name of the file its source code is in? ...and the name of the file its bytecode is in?
  - for example: consider the source code for a public Java class named `Thing`, placed within a file. What must be the file name for this file?
  - know how the Java compiler and Java Virtual Machine work together in Java translation and execution; why can Java be said to be both compiled and interpreted?
- How can you tell if a Java class is a Java application? How can you execute a Java application?
- You are responsible for those Java features that have been discussed in class, as well as for those Java features that have been used in Java and leJOS posted course examples and in course assignments.
  - You are responsible for knowing the Java naming and capitalization conventions discussed, also.
- what is a Java package?
- If you know you are using classes from a particular package, what statement can you put in a Java class file to be able to use relative class names instead of fully-qualified class names within that file?
- Need to be able to tell what given Java code would do; may need to be able to modify it, debug it, or complete it. Should be able to write Java expressions and statements as well.
- Be comfortable with Java's primitive types, especially `int`, `double`, and `boolean`.
  - Make sure that you are aware that these are not classes, and their instances are not objects.
  - Be aware of the difference between Java's `boolean` type and C++'s `bool` type; make sure that you can use Java's `boolean` type appropriately.

- Be comfortable writing and reading code using Java's versions of the basic structures (branching, looping, procedure, and sequence).
  - This includes `if`-statements, `while`-statements, and `for`-statements, as well as calling `void` methods and non-`void` methods.
  - Conditional expressions for `if`-statements, `while`-statements, and `for`-statements are required to be of what type?

Class coding standards

- You are expected to follow the class coding standards in your quiz answers.
- These include both "enforced" coding standards (required by Java syntax) and "expected" coding standards.
- Some examples (but note that this is **not** a comprehensive list):
  - A file containing a public Java class's source code must have as its name the name of the class followed by `.java` (enforced).
  - Java class names and interface names should begin with an uppercase letter (expected, although not enforced).
  - Java variable/class instance/object names and method names should begin with a lowercase letter (expected, although not enforced).

    (Constructor method names are an exception to this, of course, as they are the same as the class name, and thus start with an uppercase letter.)
  - A constant value should be declared as a named constant, declared as `static` and `final`, and should be written in all-uppercase (expected).
- Be comfortable reading and writing `javadoc`-style comments. Note that these are expected before each class and before each method.
  - also be comfortable writing `@author`, `@version`, `@param`, and `@return` comments, as applicable.
- Yes, you do need to follow course indentation standards in your quiz answers (and you might indeed lose some credit on your answers if you do not).

Java basics - more on objects

- How do you declare/define a class in Java? Be able, also, to read a class declaration/definition.
- subclasses, superclasses
  - If a class is not explicitly defined as being a subclass of another class, of what class is it a subclass?
  - What types is a subclass instance considered to be?
- Given appropriate information, you should be able to give all of the types for a given Java expression.
- I could give you source code for Java class(es)...
  - ...and ask you what classes are defined by that source code, what file it has to be stored in, what data

fields/members are in each class, what methods are defined in each class, what classes (or what package's/s' classes) does it import, what packages are referred to, what classes are used in that code?

- I could give a statement or expression containing a call to a method within that code (or executing that application, if it is a Java application), and ask you what that statement would do, or what the value of that expression would be.
- I could ask you to create an instance of that class, or to appropriately call a method from that class.

- You should comfortable with the basics of reading/writing Java constructor methods, accessor methods, modifier methods, and other methods.
- You should be comfortable declaring and using private data fields/members within a class, and understanding Java's scope rules well enough to declare them (and local variables) appropriately.
- Know that when an object is declared, it is not yet instantiated (it doesn't yet reference anything); it must be instantiated before any of its methods are called.
  - what IS an object variable's value if it has not yet been instantiated/if it doesn't currently reference an object?
  - How do you instantiate an object to a new instance of its class, then, in Java?
- Know that arrays and `String`s are objects in Java.
  - You should be able to declare, instantiate, and use arrays; given a Java array, you should be able to write an expression to determine the number of elements in that array.
  - You should be comfortable using `String` objects; you should know the preferred way to compare two `String` objects for equivalence.
  - (What are you actually comparing when you compare two objects using ==?)
  - How can you concatenate strings in Java?
- What does it mean if a Java method or data field/member is declared to be `static`? If you do not currently have an object of that class currently declared, how can you call/refer to such a method or data field/member?
- If I described a (fictitious or real) Java package, a Java class within that package, one or more of that class's constructors, and one or more of that class's methods, you should be able to read/write statements declaring and instantiating objects of that class, or calls or references to static members of that class, or method calls on objects of that class.

Java interfaces

- What is a Java interface? How many different interfaces may a single Java class implement?
- How do you indicate that a class is implementing a particular interface?
  - If a class implements an interface, what is it then expected/required to provide within its body?
- If a class implements an interface, how does that change the types an instance of that class is considered to be?
  - Given an instance of such a class, you should be able to identify all of the types for that instance.

Java 1.1 Event Model

- What does it mean for an (appropriate) object to be sensitive to user actions? What steps are necessary to make such an object sensitive to user actions?
- How can you implement a listener interface? What must you be sure to include when you do so?