

CS 235 - Exam 1 Review Suggestions

last modified: 2015-09-30

- You are responsible for material covered in class sessions, lab exercises, and homeworks; but, here's a quick overview of especially important material.
 - Note that there is some redundancy below (some topics are "approached" from different directions in more than one section).
- You are permitted to bring into the exam 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 exam is closed-note, closed-book, and closed-computer.
- This will be a pencil-and-paper exam, but you will be reading and writing code, statements, and expressions in this format. There will be questions about concepts as well.
 - You could be asked to write Java expressions, statements, fragments, methods, or up to and including entire Java classes and applications; read the questions carefully, so you do not give more answer than you have to.
 - You may be asked questions **about** Java, or about various aspects of Java.
 - You could be asked what a Java fragment does or means; you could be given an expression or fragment or class, and asked its value or what it does or what it would output in a given situation.
 - You could be asked to modify an expression or fragment or class, or to correct an expression or fragment or class, as well.
- Note that the ability to read and make use of existing code is an important skill, and seems to be frequently useful in Java programming in particular.
 - A packet of Java code will be included along with the exam, both for reference and for use directly in some exam questions. For exam purposes, the usual comments will be removed -- however, unless clearly indicated to the contrary, you can safely assume that this example code is otherwise working, correct Java.
 - Note that one of these examples will happen be a small Java application with a GUI, that happens to include at least one `JButton` sensitive to user actions.
 - It is possible that you may have to diagnose what is wrong with provided buggy code, and how it might be fixed, and/or perhaps you could be asked to modify code.
 - You might be asked to complete incomplete code (you could be given partial code, and asked to complete or modify or debug it in some way).
- Your studying should include careful study of posted examples and notes as well as the homeworks thus far.

Java basics - general

- Remember that Java is **case-sensitive**; an answer may lose points if it uses the wrong case (if a class name does not start with an uppercase letter, for example, or if you write a Java keyword using all-uppercase letters).
- What are Sun's 11 "buzzwords" for Java?
 - simple
 - object-oriented
 - distributed
 - interpreted
 - robust
 - secure
 - architecture-neutral
 - portable
 - high-performance
 - multi-threaded
 - dynamic
 - Would you be asked to reproduce this entire list? No, although it would be fair game to ask you to give 3 or 4 off the list, or to explain why it is claimed that Java can be described by these; and, you should be familiar with these (at least alleged! 8-)) features/characteristics of Java.
- Basics of the JVM (Java Virtual Machine), bytecode vs. source code, how Java's translation and execution differs from that of C++
 - What is it that makes Java unusually portable?
- Consider the source code for a public Java class named `Thing`, placed within a file. What must be the file name for this file?
- Remember that **all** Java methods/statements must be placed within a class, unlike, say, C++.
- How can you tell if a Java class is also an application?
- Say you have a Java source code file `Lookity.java`.
 - What command can you use at a command-line prompt to compile this source code into Java bytecode?
 - How many `public` classes can be in this file?
 - If it so happens that there is only one class within this file, into what file will the resulting bytecode be placed?
 - If it so happens that there are multiple classes within this file, and if you were told what classes are within that file, into what files will the resulting bytecode be placed?

- If it so happens that the public class within this file is also an application, and the source code compiles successfully, then what command can you use at a command-line prompt to execute this Java application?
- `import` statement
 - What does this do? What are its repercussions on the class(es) in the same file with this statement?
 - Given a specific `import` statement, could you give the precise name of the package involved? Could you name (or describe) which class(es) can now be used without preceding their names with their package name?
- What is the difference between a package and a class?
- Be comfortable with Java's primitive types, especially `int`, `double`, `boolean`, and `char`.
 - 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.
 - (this also includes Java's "enhanced `for` loop")
 - Conditional expressions for `if`-statements, `while`-statements, and (old-style) `for`-statements are required to be of what type?
- Be able to read and understand `try-catch` statements; be able to write them to catch and handle exceptions as well.
 - How can you use `try-catch` to react gracefully to certain user errors? to certain exceptions?
- Be familiar with, and comfortable with, Java's scope rules.
 - When does a variable have to be a data field?
 - Why might you need to declare a variable before a `try-catch` or an `if`-statement?
- How can a Java application use any command-line arguments given when that application is executed? How would such arguments be given when that application is executed?
- What statement can you use to exit from a Java application at a given point?

Class Coding Standards

- You are expected to follow the class coding standards in your exam answers; and, there could be questions about these coding standards.
- These include both "enforced" coding standards (required by Java syntax) and "expected" coding standards.

- (However, an exam question may specify exceptions to this -- for example, a question might specify that an opening `javadoc` comment might not be required for that particular question's answer.)
- 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).
 - 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 writing `javadoc`-style comments. Note that these are expected before each class and before each method (although particular exam questions may specify that these are not required for that question).
 - also be comfortable writing `@author`, `@version`, `@param`, and `@return` comments, as applicable.
- Yes, you do need to follow course indentation standards in your exam 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, then what class is it a subclass of?
 - What are some of the methods defined by that class (and thus inherited/available in every Java class, although a class might override them to provide additional/more specific functionality)?
 - (Should definitely be aware of the inherited, and often-overridden, `toString` method)
 - In the following line, which is a subclass? which is a superclass?

```
public class AClass extends Bclass
```
 - What does the subclass inherit from the superclass? What types is a subclass instance considered to be?
 - You should be able to write a method that overrides one of its superclass' methods.
- 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 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.
- I could give you a description of a method in the style that the Java 8 API describes methods, and you should likewise be able to appropriately write a call to that method.
- What does the keyword `this` mean? How is it used?
- You should be 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.
 - (If there is a point in one's code where it is possible that an object may not be currently or yet instantiated, then note that it can be compared to `null` to check:)

```
if (myObj != null)
{
    myObj.doThis();
}

...

if ((myObj != null) && (myObj.getThis() != 27))
{
    ...
}
```
 - How do you instantiate an object to a new instance of its class, then, in Java?
- What are the four levels of visibility in Java? You should be most comfortable with `private` and `public`, but should also know that the other two levels exist.
 - When there are other classes in a Java source code file in addition to a `public` class, what `.class` files are created when that Java source code file is compiled?
- Know that arrays and `Strings` 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 happens if you try to concatenate a string with a non-string?
- You should be able to use a `main` method's array-of-strings parameter appropriately.
 - How can you find out the number of command-line arguments given?
 - How can you exit prematurely if (for example) you don't like the quantity (or type, etc.) of command-line arguments provided?
 - How can you iterate through all of the command-line arguments?
- 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.
 - I could have you write a class using that class, also, or write a class that would then be a subclass of that class.
- How can one obtain a `String` depiction of an object? (Not necessarily a "pretty" one, mind...)
- How can you specify formatted output using `System.out.printf`? How can you obtain a formatted `String` instance using `String.format`?
 - You should also understand the difference between these.

Java interfaces

- Consider Java interfaces -- what is an example of such an interface that we've used so far?
- 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 applications with GUI's (graphical user interfaces)

- What are AWT, Swing? What statements should you include (to save typing, at least) when you are using classes from the packages `java.awt`, `java.swing`, and `java.awt.event`?
 - Java AWT - Abstract Window Toolkit

- Java Swing - extends, builds upon, improves the AWT in various ways (especially in its components) -- but does **not** completely replace it
- What Swing classes have we been extending so far to create GUI applications?
- What window event should be implemented somehow, some way within your GUI application? (Hint: we've generally been doing it within the application's `main` method.) What have we usually chosen to have happen when this event occurs?
- What do you need to do to give a `JFrame` (or `JFrame` subclass) a particular size, and to cause it to be displayed? What is the default size of a `JFrame` (or `JFrame` subclass) if no size is explicitly set for it?
- You should be comfortable with the GUI approach we have been taking in class examples;
 - (a small public application class that creates an instance of a `JFrame` subclass, set up as shown in those examples within a small anonymous inner class if any of the components are sensitive to user actions,
 - in which the `JFrame` subclass creates an instance of a `JPanel` subclass,
 - in which the `JPanel` subclass includes a constructor that sets up the desired components,
 - and if any of those components are sensitive to user actions, contains one or more `*Listener` inner classes to handle the actions for any such components,
 - which so far have been buttons handled by `ActionListener` inner classes...)
- Given the code for a GUI application, you should be able to read it, and answer questions such as the following:
 - What does the application do?
 - What does it look like?
 - What will it look like when it starts?
 - What components will appear, in what order?
 - (although note that the only layout manager for which you are responsible for Exam 1 is `FlowLayout`)
 - What appears (if anything) on the Java console/to the screen?) (That is, be comfortable with `System.out.println(string_expression);`)
- What are some examples of components? (`JLabel`, `JButton`, `JTextField`)
What are some examples of containers? (`JFrame`, `JPanel`)
 - You should be able to recognize these and use these.
 - On a `JFrame` (or `JFrame` subclass), how do you add panels to that frame? (Remember that you should not add components directly to a frame!)
 - On a `JPanel` (or `JPanel` subclass), how do you add components to that panel?
- be familiar with how the `FlowLayout` layout manager works; you should be able to describe what a GUI application using this would look like, by reading its code; you should be able to write code

using `FlowLayout`.

- You should be comfortable with how we've used the `java.awt.Font` and `java.awt.Color` classes so far.
- You should be able to add a border to a component or container; you should know what package's classes need to be imported for this.

Java 1.1 Event Model

- What does it mean for a component to be sensitive to user actions? ...for a `JButton`, in particular, to be sensitive to user actions? What steps are necessary to make a component sensitive to user actions?
 - Make sure that you know all of the steps for making a `JButton` sensitive to user actions.
 - Be able, if asked, to modify given code to add a `JButton` sensitive to user actions that performs a specified action when the user clicks it.
- Given the code for a GUI application that includes components sensitive to user actions, you should be able to read it, and answer questions such as the following:
 - What happens when the user clicks a given button?
- What is meant by an input textfield? ...an output textfield? ...an input/output textfield?
 - What is/are the differences in how these are implemented?
 - How can you make a textfield uneditable? Why/when would one do that?
 - How can you make the text within a textfield right-, center-, or left-justified (specify its horizontal alignment)? Why/when would one do that?
 - How can I convert the `String` in a textfield to its numeric (`double` or `int`) equivalent? What happens if one tries to perform this conversion on a string not containing appropriate contents for this?
- How can I change the text within a textfield or label? ...see what text is currently in a textfield or on a label? ...change the foreground color of a component? ...set the font used for text within a component?
- How can I find out the source of an `ActionEvent` instance? In what method would I be likely to want to do this?