CS 302 Final Exam Review

When: Saturday, May 13, 2006 from 7:45 am – 9:45 am
Where: 107 Psychology Building

IMPORTANT – MUST BRING
• UW Student-ID – It is required we take your ID when you pick up the exam, and then give it back to you at the end.
• Several #2 Pencils and a sharpener if needed – portions of the exam are scantron.

TYPE OF QUESTIONS (WRITTEN)
• Code Tracing questions require you to demonstrate your understanding of Java code that we provide by showing the output/results generated, diagramming the code's execution (e.g. memory diagrams), or describing what the code does in general.
• Code Writing questions require you demonstrate your ability to solve problems by coding a solution using your knowledge of Java syntax and semantics. You may need to use code that we provide, finish code that we have started for you, and/or use code that you have written in a previous question to answer a later question.
• Diagramming questions require you to demonstrate your understanding of different computer science and object-oriented programming concepts by interpreting or drawing specific types of diagrams. Some types of diagrams you will need to understand and create throughout the course are: class diagrams, object diagrams, memory diagrams, and control flow diagrams.
• Algorithm questions require you to explain your solution to a problem without using code. Instead, the steps required to solve the problem are described in English or pseudo-code.
• Identification or Definition questions require you to use your knowledge of terminology to select the best answers or explain the importance of concepts.

WHAT TO PREPARE FOR
• Exam officially covers all of Exam 1 & 2 material, plus chapters 10, 11, 13, 15, 16 & 17, (look on webpage for sections), CodeLab, weekly labs, and programming assignments 1-4. Read those sections of the textbook; don’t memorize examples but understand them. Understand bold terms. If you do those two things, you are well on your way.
• Form a study group; email me or the class if you are confused. Remember, I don’t grade based on perception. If you email me a “stupid” question, I’ll be happy to answer it. It is better you ask me than I see you get it wrong on the exam.
• Old exams will again be posted on Learn@UW. Take a look at them and ask questions about them. Note that you will need to look at copies of the second exam from previous semester for examples of questions on testing (chapter 10).
THINGS TO CONCENTRATE ON

Note: this is not binding; it does not include everything, nor is everything on here on the exam. Feel free to talk with me if you're unsure about the correct answer to any of these questions.

• What is the difference between unit and system testing? Between black box and white box testing?
• What is regression testing and what does it try to prevent?
• What is a unit test and a test harness?
• What are three ways to provide test cases?
• How do you verify if the output for your test cases is correct?
• What does it mean to have good test coverage? How does it relate to the terms above?
• In a debugger, what is a breakpoint?
• What does it mean to step over or step into?
• What do you use to express common and essential operations across varying classes?
• How do you define an interface? How do you implement one?
• What can be defined in an interface? What cannot be defined inside an interface?
• Can you implement multiple interfaces?
• How do you show implementation of an interface in UML?
• When converting between interface types and classes, which way does implicit conversion happen? Which way requires me to type cast?
• How does polymorphism work?
• What methods can I call on an identifier that is an Interface type? How does it know which actual method to call?
• What is a method signature? Method prototype?
• How can we check if the actual class of an object is a certain class? How can we print out the name of the class?
• Is-A hierarchy – which is subclass, which is super?
• What does a child inherit from its parents? What doesn’t it inherit?
• Can you inherit from more than one class?
• What is the ultimate parent class of all objects? What methods does it provide?
• How do we show inheritance in UML?
• What one advantage is shared by both Inheritance and Interfaces?
• How do they differ?
• How do we add new methods? Use inherited methods? Override inherited methods?
• What can be done with fields?
• How do we use super to get around the problem with private fields? How do we call superclass methods?
• What happens if we don’t call super()? Where does super() need to go in the subclass constructor?
• What is the difference between method overriding and overloading?
• Understand the ramifications of inheriting protected, private, and public fields.
• How do we convert between subclasses and superclasses? Why don’t we just treat everything with an identifier of type Object?
• How do we define toString()? How about if we want to define in superclass and have it work for subclasses too (adv 13.4)
• Understand how to throw an exception
• How do you propagate an exception? Resolve?
• How do we know which exceptions are required to be resolved and which ones we have the option resolving? Where can we find this out?
• Why are some exceptions checked and others not?
• What happens to a method of an exception is thrown? What happens if an exception is never caught?
• What happens to the try block if an exception is thrown, caught, and resolved?
• Why do we use Exceptions instead of other error checking techniques?
• Does the order of catch statements matter?
• What is the use of the finally clause? When does it execute?
• How can we use the inheritance hierarchy of Exceptions to our advantage?
• What methods do all exceptions inherit from Exception?
• Exceptions are objects
• Understand how to read and write to a text file using Scanner, FileReader, File, and PrintWriter.
• What are the two ways to parse through a file when reading? (split vs. Scanner) How is each used?
• What is the buffer and why do we have to close a PrintWriter object when we are done?
• How can we use the command line?
• What are the general steps to the software life cycle?
• What is the input for each step? Output?
• What are the different models built around this?
• What is the general purpose of Extreme Programming? Is it successful?
• What are prototypes?
• What are the general tips for discovering classes? Methods?
• What are the 4 relationships between classes? How do we depict them in UML?
• How do we show attributes and methods in UML diagrams?
• What are CRC cards and how do they help us?

I know this looks like a lot, but don't let it overwhelm you. Pick a chapter of the book or 1/3 of an old exam to look at every day and it will come easier. Don't forget to ask questions.