CS2110, Spring 2016. Preparing for Prelim 1

Prelim: 5:30-7:00 Tuesday, March 15, Kennedy Hall 116
(For students whose last name begins with A..Lib)

Prelim: 7:30-9:00 Tuesday, March 15, Kennedy Hall 116
(For students whose last name begins with Lie..Z)

If you have a conflict that prohibits you from taking the exam at the assigned time, you MUST complete assignment P1Conflict on the CMS before the end of Tuesday, March 9 (or, in one case, email Megan Gatch). (If you cannot take it at the assigned time, you can take it at the other time; we just need to know about it.) That assignment requires you to give us details. Read the statement about P1Conflict on the course exam webpage about what to say in assignment P1Conflict: www.cs.cornell.edu/courses/CS2110/2016sp/exams.html.

If you can take the prelim at the assigned time, do NOT complete assignment P1Conflict.

Review session: Sunday, March 13, Phillips 101, 1:00 - 3:00.

This handout explains what you have to know for the first prelim. The course website contains several previous CS2110 prelims. To prepare for the prelim, you can (1) practice writing Java programs/methods in Eclipse, (2) read the text, (3) memorize definitions, principles, and (4) study past prelims, on the course website, (5) Do what is suggested in piazza note on Study Habits.

In looking at past prelims, if you see a question that is outside the scope of the prelim as defined below, then skip that question. Please do not ask on the Piazza "is this topic covered on the prelim"? You shouldn't have to do that if you look at this handout for the answer. If you don't find the answer here, then ask.

A good summary of OO is provided in JavaSummary.pptx (or pdf): ~75 slides, with a 2-page index into the slides; use the slides rather than the pdf version so that you can use the animation in them to review stuff. Find a link to them on the Resources page of the course website.

Prelim 1 covers material all material in lectures/recitations through Tuesday, 3 March, on searching/sorting. Here is more detail:

1. Java strong typing. everything has to be declared before it can be used. The primitive types **int**, **double**, **char**, **boolean** (know the basic operations on them). The corresponding wrapper classes Integer, Double, Character, Boolean. You don't have to know the detailed methods in each wrapper class, but know the two reasons for having wrapper classes (be able to treat a primitive-type value as an object; provide useful static fields and methods). Understand casting between numeric types and the fact that **char** is a numeric type. Autoboxing and unboxing.

2. OO. This is a big one. Master the following:

(a) Declaration of a variable
(b) Declaration of a class and subclass
(c) What fields/methods a subclass object has
(d) Putting in the class invariant as a comment —the definitions of fields and constraints on them
(e) Access modifiers public and private
(f) Getter/setter methods
(g) Declarations of functions, procedures, constructors
(h) What the name of an object is: Foo@...
(i) Evaluation of a new-expression
(j) Value **null**
(k) Static versus non-static
(l) Constructors: purpose. Principle that superclass fields are initialized first. What the first statement in a constructor body must be. What Java inserts in a class if there is no constructor.
(m) Overriding methods

(n) Overloading method names

(o) Class Object, and the class hierarchy. What Object.toString() and Object.equals (Object) return.

(p) The uses of **this** and **super**: fields of **this**, calling other constructors of **this** class, calling constructors of **super** class, calling the **super**class's implementation of a method

(q) Casting among class types –downcasting and upcasting; the latter can be done automatically.

(r) Type of a variable v and its use in determining, say, whether v.m(…) is legal.

(s) Reason for making a class abstract; reason for making a method in an abstract class abstract.

(t) Four kinds of variable in Java: field, class variable (static), parameter, local variable

(u) Use of arrays (note: an array is an object): declaration of 1-2 dimensional arrays, length field, how one references an element (e.g. b[i]). Be able to write methods that use arrays, using appropriate syntax.

(v) Simple generic types and their use ---e.g. ArrayList<JFrame>, LinkedList<Integer> Java type-checking rules for calling a method that expects a generic type for one of its arguments.

(w) Interface declaration and implementing an interface —what that means. Casting with interfaces.

(x) Knowledge of interface Comparable and its abstract method.

(y) Exception handling: class Throwable; how to throw an exception; the try statement, with its try-block and catch-blocks.

3. Class String. You may be asked to write code that uses class String. Know methods charAt, indexOf, lastIndexOf, contains, substring, length. You are welcome to use other methods too, but we'll test on this subset.

4. Recursion. Know how to write a recursive function. Know the difference between how it executes (in terms of placing a frame for a call on the stack of stack frames) and how one understands a recursive function (Understand the body in terms of a recursive call doing what the specification says, not how it gets executed.). Know the steps in executing a method call.

5. Linked lists. Know the basic concept of a linked list, what a Node looks like. Be able to code simple methods dealing with linked lists. Understand the difference between a singly linked list and a doubly linked list. Basically, assignment A3 gave you this knowledge.

6. Loop invariants. Understand a loop in terms of a loop invariant and the four loopy questions: Start (make invariant true)? Stop (invariant together with false loop condition imply result)? Progress (loop body makes progress toward termination)? Invariant (repetend keeps the loop invariant true)? Be able to develop a loop given the precondition, postcondition, and loop invariant. Be able to generalize a precondition and postcondition given as array diagrams to a loop invariant.

7. Searching/sorting algorithms. Be able to develop these algorithms as presented in lecture: linear search, binary search, insertion sort, selection sort.