CS2110, Spring 2014. Preparing for Prelim 1

Prelim: Tuesday, 11 March; Kennedy, Call Auditorium
5:30-7:00PM if your student ID number is odd
7:30–9:00PM if your student ID number is even

If your ID # is odd and you have a conflict with the earlier time,
contact Maria; with her permission you can come at 7:30-9:00pm

Review session: Sunday, 9 March, Venue: HOLLISTER B14 from 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.

A good summary of OO is provided in JavaSummary.pptx (or pdf): ~7 slides, with a 2-page index into the slides; use the slides themselves 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. The prelim covers material all material in lectures/recitations before GUIs. 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

(i) Evaluation of a new-expression

(j) Value **null**

(k) Static versus non-static

(l) Two uses of **this** and **super**

(m) 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.

(n) Overloading method names

(o) Overriding methods

(p) Class Object, and the class hierarchy.

What Object.toString() and Object.equals (Object) return.

(q) Casting among class types –widening and narrowing, the latter can be done automatically.

(r) Static type of a variable and its use in determining, say, whether C.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<Gene>, Vector<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. Basic understanding of the collection hierarchy: List<T>, Set<T>, ArrayList<T>, HashSet<T>.

(y) Exception handling: What the superset throwable class is (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. Know how to use **class ArrayList** —how one creates an object of that class, adds elements to it, deletes elements, and accesses its size. If any other methods are needed to answer a question, we will define them for you.
5. **Recursion and memoization**. 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. You are using mathematical induction!). Know how to construct a lookup table using memoization.
6. **Grammars**. You will not be asked to write or read methods that parse grammars. But you should know the basic ideas behind a grammar as a set of rules for syntactic entities (like <noun>, <sentence>) and be able to determine whether some sequence of words belongs to (can be parsed as) some syntactic entity.
7. **Linked lists**. Know the basic concept of a linked list, what a ListNode looks like. Be able to code simple methods dealing with linked lists. Understand the difference between a singly-linked list, a linked list with header, and a double linked list.
8. **Trees**. Know the basic concept of a tree, a binary tree and a general tree. Know what the node of a binary tree looks like. Understand the concepts of preorder, inorder, and postorder of a binary tree.
9. **Binary search trees (BS**T). Know the definition of a binary search tree and a balanced binary search tree. Be able to write a recursive function that searches a binary search tree for a value in an efficient manner. Understand how nodes are added to a BST, and how the order of adding nodes will be reflected in the shape of the tree that gets built. You do not have to know about Huffman Trees and Suffix Trees.
10. **GUIs will NOT be covered on prelim 1. The GUI topics will be on prelim 2 but not prelim 1.**