CPSC 210 Software Construction
Mid-term #1 Study Guide

Thursday January 28, 2010

1. At the start of each reading, there is a check-mark list about the major concepts covered by the reading. Review these points and make sure you can do the actions listed. For example, the reading about Polymorphism, Dispatch and Type Substitution lists as an action:
    - Trace how Java source code involving polymorphism, inheritance and dynamic dispatching will execute.

    Make sure you can do this kind of tracing of a given set of Java code.

2. Go through the posted exercises.

3. Here is a list of concepts that may be covered on the midterm.
    - Explain the control-flow within a Java method by drawing a control-flow graph to represent the flow of control in a given Java method.
    - Explain how control flows between given Java methods using a call graph.
    - Given a straightforward definition of a problem (like the restaurant payment system example we did in class) and a partial data abstraction, complete the specification for the data abstraction using REQUIRES, MODIFIES and EFFECTS clauses for the operations of the abstraction.
    - Explain whether a given data abstraction (with a specification) is mutable or immutable.
    - Extract a type hierarchy from some given Java code.
    - Identify whether overriding or overloading is being used some given Java code.
    - Determine apparent and actual types for variables for some given Java code.
    - Identify if a subtype is substitutable for a supertype given specifications for each type.
    - Trace Java code involving overriding and overloading.
    - Given some Java code and an identified point at which there may be a bug, generate two hypotheses of what might be wrong.
    - Given a specification for a data abstraction, describe unit tests to test the operations of the abstraction.
    - Given a specification of a data abstraction, identify how an implementation of the abstraction may be made more robust (i.e., where should operations with restrictive REQUIRES clauses be supplemented for checking for illegal values and throwing exceptions.)
    - Trace some given Java code involving exceptions.
    - Explain whether a given invariant for a data abstraction is appropriate or could be improved.
    - Determine if a given Java implementation of a class could be used in a for-each loop and if so, explain why.

4. The following concepts will not be covered:
    - UML sequence diagrams for representing control-flow in a Java method
    - You will not be asked to complete the implementation of a specification for a data abstraction (e.g., choosing fields, etc.)

5. You should have solid working knowledge of the following Java concepts:
    - class
    - field
    - method
    - object
    - for-each loop (e.g., for (Player p: players))
    - if-else statement
    - try-catch statement
    - assert
    - that a Collection class can aggregate objects of a particular type (e.g., List<Channel>)

6. You will not be asked to write code or figure out what complicated methods do (e.g., calculations of moon phases or any such other complicated code).