Master Syllabus for TCSS 305
Version: September 2009

The **primary goals** of the course are to:

1. Reinforce and apply ideas introduced in TCSS 143, primarily object-oriented principles: inheritance, polymorphism, class design, system design, abstract classes.
2. Expose students to the process of reasoning about programs in a more disciplined way. This can include skills such as determining what a program does, determining why it is correct (or not), debugging it, and testing it.
3. Expose students to the process of designing programs of medium complexity (at least 6 interacting classes).
4. Introduce GUI and event-driven programming.
5. Reinforce and expand good programming practices:
    a. commenting and documentation
    b. testing and debugging as part of the software engineering process
    c. use of exceptions and assertions

Prerequisites: TCSS 143

Topics covered

1. Object-oriented design
    a. creation of object hierarchies
    b. creation and use of interacting objects to solve problems
    c. encapsulation, information hiding, abstract data types
    d. patterns (MVC, optionally others)
2. GUI and event-driven programming
    a. GUI components (e.g., Swing)
    b. GUI layout techniques
    c. event handling
    d. 2-D graphics primitives
3. Software engineering techniques
    a. documentation (e.g., Javadoc)
    b. debugging (strategies) and unit testing (e.g., JUnit, TestNG)
    c. basic (single-user) version control (e.g., Subversion)
    d. Design by Contract (method preconditions/postconditions, class invariants)

General Student Preconditions

1. A solid understanding of basic programming principles (TCSS 142 material): statements, variables, iteration, conditionals, method calls, parameters, primitive vs. reference types, syntax, type, casting, documentation.
2. Familiarity with and novice skill at applying object-oriented principles:
    a. classes, methods, constructors
    b. method overloading, inheritance, superclasses, subclasses, overriding, scope, interfaces
    c. polymorphism, abstract classes
3. Familiarity with and novice skill at applying other language features (exceptions, file I/O)
4. Familiarity with and novice skill at applying debugging and testing techniques.
5. Familiarity with basic algorithms: sequential search, finding the maximum item in an array, basic sorting algorithms (insertion sort).
6. Familiarity with recursion and recursive methods.
7. Familiarity with basic data structures: arrays, two-dimensional arrays, linked lists.

General Student Postconditions

1. Ability to apply object-oriented principles to design and document a good quality, medium complexity program meeting the following criteria when given some guidance on the design:
    - contains 6 or more classes
    - involves some sort of object hierarchy
    - includes interfaces and/or abstract classes
    - is presented via a GUI

    An example would be a simplified *Chess* game for two human players, which could be presented either graphically or as a console application.
2. Ability to implement and debug a good quality, medium complexity object-oriented program meeting the criteria described above.
3. Ability to design and implement unit tests for a medium complexity object-oriented program meeting the criteria described above.
4. Familiarity with data structures that will be discussed in more depth in TCSS 342: lists, sets, maps/dictionaries.

.