My Objectives for Course CSCI142

Ben K Steele
email: bks@cs.rit.edu

This attempts to provide a sort of roadmap for the objectives, or what you should learn in this course.

1 Course Description (SIS)
This course delves further into problem solving by continuing the discussion of data structure use and design, but now from an object-oriented perspective. Key topics include more information on tree and graph structures, nested data structures, objects, classes, inheritance, interfaces, object-oriented collection class libraries for abstract data types (e.g. stacks, queues, maps, and trees), and static vs. dynamic data types. Concepts of object-oriented design are a large part of the course. Software qualities related to object orientation, namely cohesion, minimal coupling, modifiability, and extensibility, are all introduced in this course, as well as a few elementary object-oriented design patterns. Input and output streams, graphical user interfaces, and exception handling are covered. Students will also be introduced to a modern integrated software development environment (IDE). Programming projects will be required.

1.1 A ‘Dissection’ of the Description
The description tells you that we will be covering this stuff:
• object-oriented programming approaches and techniques;
• cohesion, minimal coupling, modifiability, and extensibility concepts;
• further uses and applications of lists, stacks, queues, maps, and trees;
• graph data structures;
• introductory design patterns;
• graphical user interfaces;
• Input/Output (I/O), and
• integrated development environments (IDEs) for programming.

2 ‘Chunks’ of Topics (week numbers are in parentheses)
2.1 Object-Oriented Programming with the Java language (1-5)
By the end of this unit, you should be able to explain, analyze, and apply these concepts to the design and implementation of basic object-oriented programs:
1. classes, objects, interfaces, inheritance and abstraction;
2. fields, functions, methods and messages;
3. encapsulation and data-hiding; and ‘programming to the interface’;
4. cohesion, coupling, modifiability, and extensibility; and
5. collections and generic types.
2.2 Graph Data Structures (6-9)

By the end of this unit, you should be able to explain, analyze, and apply these concepts to the design and implementation of basic graph-based programs:
1. conceptual graph representations;
2. graph representations in computer code;
3. graph search algorithms (depth-first, breadth-first, . . . ), and
4. backtracking algorithms.

2.3 Graphical User Interfaces, Concurrency and Communications (10-15)

By the end of this unit, you should be able to explain, analyze, and apply these concepts to the design and implementation of basic GUI-based and/or concurrent programs:
1. basic GUI components and layouts in Java;
2. event-driven program design;
3. lambdas for writing very short ‘callback’ methods;
4. threading to handle concurrent events, and
5. I/O and network communications.

2.4 Cross-cutting Concerns

Finally, there are a number of continuing, cross-cutting concerns that will be discussed throughout the term:
1. using IDEs;
2. using the ‘git’ version control tool (starting in week 2);
3. testing;
4. debugging;
5. complexity;
6. performance;
7. using (UML) pictures to show the conceptual structure of programs, and
8. continuing to use both recursion and iteration to solve problems.