1. Object-Oriented Design

(a) Principles of Object Oriented Design

i. understand why object-oriented designs differ can handle changing requirements better than designs gained from functional decomposition

ii. know what it means to say that an object is an "entity with responsibility" and be able to contrast that with the definition of object as "data + behavior" (i.e., understand why it is important to be able to abstract away a class’s implementation and think about its interface)

iii. know basic OO vocabulary: abstract class, class, concrete class, encapsulation, inheritance, instance, instantiation, interface, polymorphism, attribute, class, constructor, derived, class, member, method, object, superclass

iv. know what encapsulation is, be able to give examples of it, and understand why it is useful

v. know what polymorphism is, be able to give examples of it, and understand why it is useful

vi. know what inheritance is, be able to give examples of it, and understand why it is useful

vii. understand the role of abstract classes as a conceptual classification

viii. understand what coupling is, be able to give examples of loose coupling, and be able to explain why loose coupling is desirable

ix. understand what cohesion is, and be able to explain why high cohesion is desirable

x. be able to explain what the difference between cohesion and coupling is

(b) UML

i. know how to represent classes, attributes (fields), methods, accessibility, generalization (‘is a’), association (including ‘has a’ — aggregation and composition), dependency (‘uses’), and cardinality in a UML class diagram

ii. given a Java program, be able to draw a UML class diagram showing its classes, their attributes, and their relationships

(c) CAD / CAM case study

i. know what a CAD / CAM system is

ii. know what a numerically controlled machine is

iii. know what is different between v1 and v2 of the CAD / CAM system

iv. understand the roles of the expert system, CAD/CAM system, the geometry extractor, and the numerically controlled machine in the

v. understand why it was necessary to decouple the expert system from the CAD/CAM system

vi. know what the author’s initial solution to the CAD / CAM problem is

vii. understand why that solution exhibits redundancy, tight coupling, and weak cohesion

(d) Design Patterns

i. know that idea for design patterns came from Christopher Alexander’s work in architecture

ii. know that the ‘Gang of Four’ wrote “Design Patterns: Elements of Reusable Object-Oriented Software”, which was instrumental in introducing design patterns to software developers

iii. know that ‘Design Patterns: Elements of Reusable Object-Oriented Software’ contains a description of what design pattern are

iv. know that ‘Design Patterns: Elements of Reusable Object-Oriented Software’ contains a catalog of design patterns already being used

v. know that each design pattern should (1) have a name, (2) describe some problem in context, (3) describe a solution to that problem, and (4) describe the consequences of applying that solution

vi. understand how design patterns raise the level of abstraction

vii. understand how design patterns help us learn from the experience of others

viii. understand how design patterns can illustrate important design principles

ix. know what it means to “design to interfaces” and what benefits this could bring

x. know what it means to “favor aggregation over inheritance” and what benefits this could bring
xi. know what it means to “find what varies and encapsulate” and what benefits this could bring

(e) Facade Design Pattern
i. know the intent, problem, solution, participants and collaborators, consequences, and implementation of the design pattern as described in DPE
ii. know that a facade provides a new, simpler interface to an existing subsystem
iii. be able to identify situations in which the Facade pattern is applicable
iv. be able to identify situations in which the Facade pattern is not applicable

(f) Adapter Design Pattern
i. know the intent, problem, solution, participants and collaborators, consequences, and implementation of the Adapter design pattern as described in DPE
ii. understand how the Adapter pattern facilitates polymorphism
iii. be able to identify situations in which the Adapter pattern is applicable
iv. be able to identify situations in which the Adapter pattern is not applicable
v. be able to draw UML class diagrams illustrating the Adapter design pattern
vi. be able to recognize Adapter classes being used support polymorphism in a UML class diagram
vii. be able to distinguish between the Facade and Adapter design patterns

2. Software Development

(a) Agile Development Processes
i. know how Agile development differs from the waterfall life-cycle
ii. be able to list several characteristics of the Agile development (time-boxed iterations, pervasive use of unit testing, integrated customer collaboration, working code after each iteration)

(b) Use Cases
i. know what a use case is
ii. know what an actor is
iii. know what an interaction is
iv. be able to write a use case scenario

(c) Eclipse
i. know what an Eclipse workspace is
ii. know what an Eclipse project is
iii. know how to create, edit, compile, run Java classes/programs in Eclipse

(d) CVS (w/Eclipse)
i. know what a CVS repository is
ii. know what a CVS module is
iii. know what it means to checkout a CVS module
iv. be able to update an Eclipse project from a CVS repository
v. be able to use Eclipse to commit changes to a CVS repository

(e) Testing
i. know what white box and block box testing are
ii. know what a unit test is
iii. be able to create JUnit test cases in Eclipse
iv. be able to run JUnit test cases in Eclipse

(f) Chat Program
i. be able to write a use case scenario for the chat system
ii. understand what a client / server architecture is
iii. know some of the advantages of choosing a client / server architecture for the chat client
iv. know some of the disadvantage of choosing a client / server architecture