

CS 4773 – OBJECT-ORIENTED SYSTEMS
Final Review Topics

Fundamentals

- What is object-oriented programming?
- What are the alternatives to object-oriented programming?
- Name at least four object-oriented programming languages.
- What does it mean for an object to “have responsibilities”?

Important OO terms: object, instance, method, message, class, field/attribute, constructor, inheritance, polymorphism, overloading, overriding. *(Each term deserves to be on a sheet of paper on its own, but there are limits to how much I can stress this point.)*

Design Patterns

- What is a design pattern?
- What are the parts of a design pattern?
- What is the Façade pattern?
- Given the intent of the Adapter pattern, can you describe a case of its use?
- Given the intent of the Strategy pattern, can you describe a case of its use?
- Given the intent of the Bridge pattern, can you describe a case of its use?
- Given the intent of the Abstract Factory pattern, can you describe a case of its use?
- What is the Decorator pattern?
- Given the intent of the Observer pattern, can you describe a case of its use?
- What is the Model-View-Controller pattern?
- What is the Iterator pattern? Can you implement it for a given data structure?
- Given the intent of the Visitor pattern, can you describe a case of its use?
- Given the intent of the Template Method pattern, can you complete an implementation using it?
- What is the Singleton pattern?
- What are the advantages of using factories?
- Given the intent of the Factory Method pattern, can you describe a case of its use?

Other Object-Oriented Concepts

- What is delegation?
- What is the difference between a framework and an API?
- What is covariance?
- What is contravariance?
- What is the Open-Closed Principle?
- What is the Liskov Substitution Principle?
- What is the difference between structural subtyping and nominal subtyping?
- What properties does the subtyping relation have? How is it written?
- What issues are there with writing a “proper” equals method?
- What is an inner class? How is it useful in design and implementation?
- What is an anonymous inner class? How is it useful in design and implementation?
- What is an event based model?
- How are wildcards used in Java? What problem do they solve?
- What is the set interpretation of types in OO?

UML

What is the value of UML?

What is UML used for?

Create a Message Sequence Diagram/Interaction Diagram from a description.

Create a Class Diagram from a description. You are expected to know how to notate: a class, an abstract class, an interface, the *is-a* relation, aggregation, multiplicities.

Software Engineering

What is requirements analysis?

What is a use case?

What is a concern?

What is a crosscutting concern?

What is scattering?

What is coupling? Tight coupling? Loose coupling?

What is encapsulation?

What is information hiding?

What is abstraction?

What is cohesion?

How can you design for change?

What is a call back?

Note on the Types of Questions

Be able to provide cogent, complete, and correct definitions of key terms.

Look at code and find the bug.

Look at code and explain what it does.

Look at code and create UML diagrams from it.

Work with pseudo-Java programs, with modified semantics (such as using structural subtyping).

Be able to understand a problem similar to the display update problem (Lecture 8), which means understanding why the problem is a problem, and what its solution is.

Questions related to looking at code will use design patterns, language features, and concepts we've looked at.

Note: For every term or concept mentioned on this sheet, you are expected to know the context of that concept, its advantages, its disadvantages, and how it relates to other concepts.