COMP 2000 Final Review sp24

Topics

- The Object-Oriented Paradigm
  - Metaphor
  - Perspectives & Roles (Implementor, Client, Designer)
  - Abstraction & Composition
- Objects, Classes, & Interfaces
  - Attributes, Methods, & Constructors
  - Definitions of “Interface”
  - Packages
  - Design: object responsibilities
- Inheritance
  - Abstract classes & methods
  - Class & Interface
  - Polymorphism
- Exceptions
- Design by Contract
  - Obligations & Benefits
  - Precondition, postconditions, requires, ensures
  - Javadocs
- Testing
  - Stages and Types of testing
  - JUnit & Assertions
  - Test-driven development
- GUIs
  - Swing and awt
  - JFrame & JPanel & other JWidgets
  - ActionListeners
- User-Interface Design
  - MVC
  - Being a butler
  - Respect physical and mental effort
- Frameworks
  - Concept & principles
  - JCF
- UML
  - Class diagrams
  - Use cases
  - Object diagrams
  - Sequence diagrams
- Design Patterns
  - Iterator, Strategy, Observer
  - Decorator, Adapter, Singleton, MVC
- Refactoring
- File IO

COMP 2000 Final Review sp24

Things to think about

[The following questions are not intended to be exhaustive of what will be on the exam, but should be used as study guides. For further study, ask similar questions about the rest of the topics listed above and in the syllabus/schedule. Then ask more questions about what you find in the readings & PowerPoints.]

Compare and contrast the roles of composition/aggregation (“has-a”) and generalization/specialization (“is-a”) in aiding in the design of computer systems.

In what way do objects and classes exhibit abstraction?

What is a reference to an object? Explain why a local variable might need to reference an object. Explain why one object might need to reference another object.

How does a local variable that is specified in the parameter list get initialized? If you try to update the parameter, what happens?

How does a specification differ from an implementation? Which should be done first? Why?

What is an interface? Why are interfaces useful? How is an interface similar to an abstract class? How is it not similar?

What is multiple inheritance? In Java, how does multiple inheritance relate to interfaces?

Can you declare a reference variable that references an interface? If so, how can that be, because you can’t create a new instance of an interface type in Java? How could you initialize the reference variable?

What is refactoring? Why is it useful? What does it mean for code to smell?

Why are design patterns organized into behavioral, structural, and creational categories?

What is the distinction between User-Interface Design and Graphical User Interfaces?

Describe the trade-offs in either keeping the Controller and the View separate vs. consolidating them into a single component.

What are the principles or rules of thumb that govern how you should choose whether something should be declared public, private, protected, or package?

If someone asked you to define Object-Oriented Design, what would you say?