Use the following outline to guide your self-assessment and notetaking

Week 5 – Object Oriented Design

Software Development Activities (Ch 7.1)

Requirements

- Must specify what a program needs to accomplish
- What tasks should the program perform?
- Expressed in a functional specification document
- Includes characteristics that make the program useful for the end user
- Apply constraints to a program

Design

- Answers how the program will accomplish its requirements
- Define classes/Objects/methods needed, their relationships, and how they will interact
- Alternatives

Implementation

- Basically, putting the design into action (writing the source code)
- Translating the design into a specific programming language

Testing

- Ensure that the program will solve the intended problem given all conditions under which it should perform
- Involves running with various inputs and scrutinizing the results

Identifying Classes and Objects (Ch 7.2)

Finding your candidate classes

- Best way is to scrutinize a problem description and highlight key words (nouns)
    - Of course, not every noun would be a class, but it's a start
- Classes should generally use singular nouns
- May also consider associated behaviors of attribute. Ex: salary depends on title and rank
- Classes may also be reusable and could be extended as well

Responsibilities

- Generally we use verbs to denote the behavior of a class and methods
- Always consider multiple possibilities
- Not necessary to identify everything a class can do in the first stages
    - Just assign primary responsibilities and consider how they translate to particular methods

Static Class Members (Ch 7.3)

Static Keyword

- Used to indicate that a method or variable is shared across objects. Therefore, they can be invoked through the class name as opposed to an instance of the class.

Static variables

- Shared among all instances of a class
- Only one copy exists for all objects of the class, so changing one will reflect across all objects that have that variable
- Local variables in a method cannot be static

Static methods

- Can be invoked through the class name as opposed to an instance of the class
- No object state to maintain
- The main method is static because there is no need to instantiate an object containing it
- Cannot reference instance variables, but it can reference static variables

Class Relationships (Ch 7.4)

Dependency among classes

- Some classes can rely on another
    - If class A uses class B, then one or more methods of A invoke one or more methods of B
- Usually, occurs when one class instantiates the objects of another or when one object is passed as a method parameter to the other class
- We generally want to minimize multiple classes being dependent on one another

Dependency in same class

Aggregation

- Objects can be made up of other objects
    - Ex: Car made up of engine, chassis, wheels, mechanical parts, which could be considered separate objects
    - Car is therefore an aggregation
- An aggregate object is any object that has references to other objects as instance data
- Methods of the aggregate object invoke the methods of the compose objects

this keyword

- Refers to the object itself
    - Ex: this.position gets the current value of the position attribute in a ChessPiece class

Interfaces (Ch 7.5)

Defining

- It is a collection of constants and abstract methods (methods that do not have an implementation)
- Header of the method with the parameter list is followed by a semi-colon (no body)
- Cannot be instantiated

- Classes implement an interface by providing method implementations for abstract methods
    - Done using the implements keyword in the header and can implement more than one (comma-separated)

Comparable

- Defined in java.lang
- Only one method (compareTo) which takes an object as a parameter and returns an integer
    - Up to the designer to decide what it means for one object to be less than, greater than, or equal to another
    - Ex: strings have a compare to method to compare based on lexicographic ordering

Iterator

- Used by a class that represents a collection of objects, which allows a means to move through the collection one object at a time (like with a loop)
    - hasNext() – Determines whether or not there are items left to process in the collection
    - next() – returns the next object in the collection

Enumerated Types (Ch 7.6)

Relationship with static variables

- variables referencing an enum can only be assigned the values listed in the enum definition
    - Ex: (winter, spring, summer, fall) are references to Season objects stored within the season class
    - Time = Season.spring;
    - Enums can also have constructors

Method Design (Ch 7.7)

Decomposition

- Sometimes, we may need to use multiple methods because a service that an object provides is very complicated
- These methods are then called executed inside the main method, which help with readability

Helper methods

- Helper methods help break up large procedures into smaller ones by decomposing tasks in multiple methods. This helps with readability and can also be used to avoid repetition across multiple objects

Method Parameters

Method Overloading (Ch 7.8)

Why?

- Method overloading is used to perform similar methods on different data types

How

- The same name gets used for the methods
- The way to distinguish them is by the number, type, and order of parameters
- Name along with the parameters determine the method's signature and must be unique

Testing (Ch 7.9)

Reviews

- Several people meet to examine a design document or section of code
- Participants discuss merits, suggestions, improvements, and issues to address
- Way of identifying problems, not solving them

Defect Testing

- This is a way to find errors before the user does
- Consists of cases to test for the program
- Output is documented along with the test cases
- Not feasible to exhaustively test all possible input

Black Box

- Tests are developed without regards to the program's workings
- Based on user-provided input and program's output
- Often called equivalence category

White Box

- Tests are developed according to the program's workings
- This is done to ensure every path (condition) is executed at least once
- Often called statement coverage