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

Week 7 – Inheritance and Polymorphism

Creating Subclasses (Ch 9.1)

Deriving means …

- Creating multiple classes from the same blueprint (the parent class)
    - Ex: creating a house in different locations for different people
    - These classes are related by methods and attributes but they may do different or additional things

Reuse benefits

- No need to write the same features over again from scratch
- A programmer can easily add new variables and methods to the derived class or modify the inherited ones

Is-a relationship

- The child version of the class is a more specific version of the parent
    - Ex: Horses are mammals but not all mammals are horses
    - All squares are rectangles but not all rectangles are squares

The `protected` visibility modifier

- Private methods and variables are only visible to the parent class
    - Cannot be referenced outside of the class, the child class, or an object of the child class
- When a variable or method is declared with protected visibility, a derived class can reference it, but it is not visible anywhere else
- Retains encapsulation

The `super` reference

- Constructors of the parent class are not inherited
    - Cannot invoke it directly in the child class to get the attributes
    - Used to access the members of a parent class (the parent constructor)
- What it refers to depends on the class in which it's used
- A change made in the parent class is automatically reflected in the child class

Multiple inheritance

- Describes objects that are in between 2 categories or classes
    - Ex: pickup truck is somewhat like a car and a truck
- Java does not support multiple inheritance. Instead, interfaces are used for this purpose, since a class can implement multiple interfaces

Overriding Methods (Ch 9.2)

- The override happens in the parent class

- Child class overrides the parent's version in favor of its own
- Super reference invokes the parent's version
- Child classes cannot override a final method

Shadowing variables

- Child classes can declare a variable with the same name as the one inherited from the parent, but it is not recommended since variables are already available to the child class and this will cause confusion (multiple declarations)
- Similar to overriding but for variables

Class Hierarchies (Ch 9.3)

Parents and ancestors

- The child of one class can be the parent of one or more classes, creating a hierarchy
- Multiple classes can be derived from a single parent
- 2 children of the same parent are called siblings
- Common features are located as high in a class hierarchy as possible
- Inheritance is transitive

The `Object` class

- All classes are derived from this class by default
- All public methods of the object class are inherited (clone, toString, equals, etc.
  - Defining these methods in a class will override it

Abstract classes

- Cannot be instantiated
- Uses the abstract modifier
- Represents a concept on which other classes can build their definitions
- Similar to an interface, but it can contain methods that are not abstract and also can contain data declarations other than constants
- These classes do not have to contain abstract methods
- Classes derived from an abstract parent must override all of its parent's abstract methods

Interface hierarchies

- Hierarchies can also be applied to interfaces so that one interface can be derived from one another
- The do not overlap because an interface cannot be used to derive a class and a class cannot be used to derive an interface

Visibility (Ch 9.4)

Private members affect child classes …..

- Private members are not visible to child classes but can be referenced indirectly.

Designing for Inheritance (Ch 9.5)

Why?

- Leads to a more elegant design
    - Keep in mind that every derivation is an is a relationship
    - Class hierarchy should be reusable
    - Find common classes and objects and push common features as high in the class hierarchy as appropriate for consistency and ease of maintenance
    - Don't shadow variables
    - Override methods as appropriate to tailor or change the functionality of a child
    - Allow each class to manage its own data
    - Use visibility modifiers to provide needed access in derived classes
    - Use abstract classes to specify a common class interface
    - Override general methods appropriately so that inherited versions don't cause unintentional problems
    - Design class hierarchy to fit the needs of the application

Restricting Inheritance

- Uses the final modifier to ensure that a class is stand-alone and cannot be extended
- Use this when a child class could possibly be used to change functionality that the programmer wants to be handled in a specific way

Late Binding (Ch 10.1)

Polymorphism means …

- Having many forms
    - Polymorphic reference – A reference variable that can refer to different types of objects at different points in time

Late binding means …

- The commitment made to execute a method invocation is determined at run time

Polymorphism via Inheritance (Ch 10.2)

Reference variables and inheritance

- Can refer to any object created from any class related to it by inheritance
- A parent object can also be assigned to a child reference, but it requires an explicit cast. This is less useful and can cause problems
- Object type at runtime determines which version of a method is invoked

Polymorphism via Interfaces (Ch 10.3)

Using an interface name to declare an object

- Interfaces can be used to refer to any object of that class that implements that interface
    - Ex: named interface is speaker, defined class is philosopher
    - Speaker can reference the Philosopher object

        - Speaker current = new Philosopher();
    - Works because every philosopher is a speaker

How interface references can be used in polymorphism

- We can create polymorphic references that can refer to any one of a set of objects as long as they are related by inheritance
- When using an interface reference, we can only invoke the methods implemented in that interface even if there are other methods defined for that class