Nested and Composite Classes

Lecture 14
COP 3252 Summer 2017

May 30, 2017

Nested Classes

- The Java programming language allows you to define a class within another class. Such a class is called a nested class.
- Nested classes are divided into two categories: static and non-static.
- Nested classes that are declared static are called static nested classes. Non-static nested classes are called inner classes.

```
class OuterClass {
    ...
    static class StaticNestedClass {
        ...
    }
    class InnerClass {
        ...
    }
}
```

Access

- A nested class is a member of its enclosing class.
- Non-static nested classes (inner classes) have access to other members of the enclosing class, even if they are declared private.
- Static nested classes do not have access to other members of the enclosing class.
- As a member of the OuterClass, a nested class can be declared private, public, protected, or package private. (Outer classes can only be declared public or package private.)
- Why Use Nested Classes?
  - It is a way of logically grouping classes that are only used in one place
  - It increases encapsulation
  - It can lead to more readable and maintainable code

Static Nested Classes

- A static nested class is behaviorally a top-level class that has been nested in another top-level class for packaging convenience.
- A static nested class is associated with its outer class. cannot refer directly to instance variables or methods defined in its enclosing class: it can use them only through an object reference
- Static nested classes are accessed using the enclosing class name:
  `OuterClass.StaticNestedClass`
- For example, to create an object for the static nested class, use this syntax:

```
OuterClass.StaticNestedClass nestedObject =
            new OuterClass.StaticNestedClass();
```

Inner Classes

- An inner class is associated with an instance of its enclosing class and has direct access to that object's methods and fields.
- Also, because an inner class is associated with an instance, it cannot define any static members itself.
- To instantiate an inner class, you must first instantiate the outer class. Then, create the inner object within the outer object with this syntax:

```
OuterClass.InnerClass innerObject =
            outerObject.new InnerClass();
```

Composition of Classes

- Composition is preferred over inheritance when there is a "has-a" relationship between the two classes.
- Java composition is achieved by using instance variables that refers to other objects.
- Using composition, one can control the visibility of other object to client classes and reuse only what's needed.