Lecture 9

- Topics Covered
  - Inheritance
Inheritance Terminology

• Parent class
  – Base Class, Super Class

• Child class
  – Derived Class, Sub Class

• In general, the idea is derived class inherits data members and methods from the base class
  – Let’s look at the specifics of what is inherited in a java inheritance hierarchy
Inheriting from a base class

- Syntax that allows inheritance
  - `public class Derived extends Base {...}`
Modifiers and inheritance

- **private**
  - Visible only to its defining class
  - Memory is still allocated for private variables, but no visibility in derived class methods

- **public**
  - Visible to any class

- **protected**
  - Visible to defining class and all subclasses
Inherited Methods

- If a method is public or protected in the base class, it is inherited in the derived class
- The exception is when the derived class implements its own version... this is called overriding
  - When a derived class overrides a base class method, the base class method is hidden
Constructors and inheritance

- Constructors of the base class are not inherited by its derived classes
- However, there is a way to invoke the base class constructors from the derived class constructors
  - If you want to invoke the default constructor of the base class inside a derived class constructor, use the following statement
    - `super();`
    - If `super()` is used, it must be the first statement of the derived class’ constructor
    - If a different base class constructor is desired, then use the word `super` with the appropriate number and type of parameters
    - If no base class constructor is specified, the default constructor will always be called
Calling base class methods

- If you override a base class method in a derived class, it is typically to add functionality based upon new data members in the derived class. However, it is often still desirable to run the code in the base class method.

- If you call the base class method inside the derived class’ version of the method, you must use super in the following way
  - `super.methodName();`
Example

• Let’s look at a TV Remote class and extend it
Potential Downfalls

• If a base class changes, could make problems for derived class
  – Safer to make calls to base class methods rather than reiterate the code from the methods
Exercise

- On one computer work with your programming team
  - Decide on a new feature for a Remote and extend the NumberRemote to add this feature