Computer Programming II Instructor: Greg Shaw

COP 3337

# Inner Classes

* We create an inner class by nesting one class definition within another.
* Inner classes are commonly used when one class requires the functionality of another (i.e., the ability to create and manipulate objects of that class), but wants to limit its *visibility*. I.e., if class B is defined within class A, then inner class B can be hidden from client code that has access to A. (We will see an example of a “hidden” inner class in the upcoming unit on Data Structures.)
* (Recall that classes must be declared **public** or use "friendly" access. Classes can't be declared **private**. An inner class declared with friendly access is effectively hidden from all client code not in the same package)
* Outer class methods are allowed to directly access the instance variables and methods of the inner class. It makes no difference whether they are declared **public**, **protected**, or **private**, or use "friendly" access
* An inner class may also be used to implement a “strategy interface,” also known as a “callback.” I.e. using an interface and its “realizing” classes to perform a common task among unrelated classes.
* Section 10.4 in the textbook provides a good example of using an inner class to implement a “callback”
* An inner class may also be nested inside a *method*. In that case, the class may access the local variables of the method only if they are declared **final**. This may sound like a huge restriction, but actually is not. The contents of a final variable may not be changed, of course, but in the case of object variables - which store addresses of objects - this means only that the variable cannot be made to point to a different object. However, the object to which it points may still be modified. (See *ParamPasser2.java*, online)