# CIS 3145 Class Notes: Text Chapter 13

## Inner Classes, Enumeration & Documentation

**Objectives**

* Create additional classes per file and nested classes
* Create and use enumerations
* Create documentation for a project

**Multiple Classes per File**

Most of the time we will not put multiple classes in a single .java file. We will follow the “one class, one file” rule. On reason to follow this rule is it makes it easier to manage the code. When we work with Graphical User Interfaces (GUI) programs we will use inner classes and anonymous classes.

We can put multiple classes in a single .java file but only the **outer** class can be declared with the **public** modifier. Also the .java file will have the same name as this **outer** public class.

A class can be **nested** inside another class. This **inner** class **cannot** be **public** nor can it have **static** variables or methods. They can have their own **instance variables** and **methods**, plus they can access private variables of the outer class.

An **inner** class can be defined as an **anonymous** class inside curly brackets immediately after the object declaration. The code for the class goes in the curly brackets when an object is declared. If the **anonymous** class is used **only once** it can even be declared **immediately after the parentheses** for a method argument. It this case it is not associated with a variable.

**Create an enumeration**

Add the enumeration just like a class. It is best to put an enumeration in a separate file because it becomes easier to share the enumeration.

We use enumerations to make it easier to **read the code**. An enumeration is a data type of **integers**! The enumerations are a set of **related constants** thus we capitalize them to make them standout. The enumerations are **automatically** given integer values starting with **zero**.

When a variable is defined with an **enumeration** data type and this forces us to put only an acceptable enumerated value in the variable. The *autocomplete* feature of Netbeans helps populate the variable.

A great thing about the enumeration is that we can add our own methods or override existing methods such as the **toString**() method.

@Override

Public String toString()

{

String s = “”;

if (this.ordinal() == 0)

s = “UPS Next Day”;

else if (this.ordinal() == 1)

s = “UPS Second Day”;

else if (this.ordinal() == 2)

s = “UPS Ground Day”;

return s;

}

Complete exercise 13-2 (page 352) to practice creating enumerations.

**Javadoc generation**

Using /\*\* … \*/ immediately before a class, constructor, method, and public / protected fields will create entries in a javadoc HTML page. Right click a project and select “Generate Javadoc”. The files are stored in the dist\javadoc folder.

**Tags** are used in the javadoc comments to identify standard information about the program. The @author and @ version tags will describe the class while @param and @return tags are used to describe what a method does.