The Anatomy of a Class

The Name of the Class

The name of the class is important for several reasons. The obvious reason is to identify the class itself. The choice of descriptive name is important because it provides information about what the class does and how it interacts within larger systems.

The name is also important when considering language constraints. For example, in Java, the public class name must be the same as the filename. If these names do not match, the application won't work.

Attributes

Attributes represent the state of the object because they store the information about the object.

The keyword '**private**' signifies that a method or a variable can be accessed only within the declaring object.

The '**static**' keyword signifies that there will be only one copy of this attribute for al the objects instantiated by this class.

Constructors

It is always good idea to initialize attributes in the constructors. It's also a good programming practice to then test the value of an attribute to see whether it is null. This can save you a lot of headaches later if the attribute or object was not set properly. If you set a reference to null in the constructor, you can later check to see whether it is still null when you attempt to use it. An exception might be generated if you treat an uninitialized reference as if it were properly initialized.

Consider another example: if you have an Employee class that includes a spouse attribute, you better make provisions for the situation when an employee is not married. By initially setting the attribute to null, you can then check for this status.

Accessors

There are many times when an object needs to access another object's attributes; however, it does not need to do it directly.

A class should be very protective of its attributes. For example, you do not want **object a** to have the capacity to inspect or change the attributes of **object b** without object b having control. There are several reasons for this; the most important reasons boil down to data integrity and efficient debugging.

If name were public and any class could change it , you would have to go searching through all the possible code, trying to find places that reference and change name. However, if you let only a cabbie object change name, you would have to look only in the cabbie class. This access is provided by a type of method called an **accessor**. Sometimes accessors are referred to a getters and setters, and sometimes they're simply called get() and set().

Static Attributes

If an attribute is static, and the class provides a setters for that attribute, any object that invokes the setter will change the single copy. Thus, the value for the attribute will change for all objects.

Public Interface Methods

Both the constructors and the accessor methods are declared as public and are part of the public interface. They are singled out because of their specific importance to the construction of the class.

Private Implementation Methods

Private methods are meant to be part of the implementation and not the public interface. These private methods are called from the class itself.

Private methods are strictly part of the implementation and are nor accessible by other classes.