The Object-Oriented Thought Process

Chapter 02

How to Think in Terms of Objects

How to Think In Terms of Objects

Three important things you can do to develop a good sense of the OO thought process are:

- Knowing the difference between the interface and implementation
- Thinking more abstractly
- Giving the user the minimal interface possible

Interface vs. Implementation

When designing a class:

- What the user needs to know and what the user does not need to know are of vital importance.
- The data hiding mechanism inherent with encapsulation is the means by which nonessential data is hidden from the user.

The Interface

The services presented to an end user compose the interface.

- In the best case, only the services the end user needs are presented.
- As a general rule, the interface to a class should contain only what the user needs to know.

The Implementation

The implementation details are hidden from the user.

- A change to the implementation should not require a change to the user's code.
- If the interface does not change, the user does not care whether or not the implementation is changed.

Abstract Thinking

One of the main advantages of OO programming is that classes can be reused.

- Reusable classes tend to have interfaces that are more abstract than concrete.
- Concrete interfaces tend to be very specific.
- Abstract interfaces are more general.

The Minimal User Interface

Give the users only what they absolutely need.

- It is better to have to add interfaces because users really need it than to give the users more interfaces than they need.
- Public interfaces define what the users can access.
- It is vital to design classes from a user's perspective and not from an information systems viewpoint.
- Make sure when you are designing a class that you go over the requirements and the design with the people who will actually use it—not just developers.

Determining the Users

Who are the users?

- The first impulse is to say the *customers.*
  - *This is only about half right.*
- *Although the customers* are certainly users, the developers must be a partner with the users.
  - In short, users can have unrealistic expectations.

Object Behavior

After the users are identified:

- You must determine the behaviors of the objects.
- From the viewpoint of all the users, begin identifying the purpose of each object and what it must do to perform properly.
- Note that many of the initial choices will not survive the final cut of the public interface.

Environmental Constraints

Environmental constraints are almost always a factor.

- Computer hardware might limit software functionality.
- A system might not be connected to a network, or a company might use a specific type of printer.

The Public Interfaces

Think about how the object is used and not how it is built.

- You might discover that the object needs more interfaces
- Nailing down the final interface is an iterative process
- It is often recommended that each interface model only one behavior.

Indentifying The Implementation

Technically, anything that is not a public interface can be considered the implementation.

- This means that the user will never see any of the methods that are considered part of the implementation.
  - Including the method's signature (which includes the name of the method and the parameter list), as well as the actual code inside the method.