

**Introduction to
Object Oriented Programming 2E
Timothy A. Budd**

Chapter 17

Visibility and Dependency

Connections – The Bane of Large Scale Programming

Difficulties in developing large scale programs are often not so much a matter of algorithmic complexity as they are of communication complexity.

If several programmers are working together on a project, need to control the amount of information one programmer must have about the code being developed by a second programmer.

Visibility

Visibility is an attribute of *names*.

- Names of variables, functions, fields, whatever.
- If you can't name something, you can't manipulate it.
- Languages already have a variety of mechanisms for the control of name visibility.
- OOP languages introduce a few new twists.

Dependency

Dependency describes the degree to which one software component relies on another component to perform its responsibilities.

A high degree of dependency obviously limits code reuse – moving one component to a new project.

Coupling and Cohesion

Ideas from the Software Engineering Community, pre-dating OOP.

- **Coupling** refers to the extent to which one component uses another to perform actions. Generally a goal is to reduce coupling between software.
- **Cohesion** refers to the extent to which the actions of a component seem to be tied together in purpose. Generally a goal is to increase cohesion within a software component.

Varieties of Coupling

- Internal data coupling
- Global data coupling
- Control (or sequence) coupling
- Parameter coupling
- Subclass coupling

Varieties of Cohesion

- Coincidental cohesion
- Logical cohesion
- Temporal cohesion
- Communication cohesion
- Sequential cohesion
- Functional cohesion
- Data cohesion

Limiting Coupling – the Law of Demeter

The law of demeter is an attempt to limit the way in which one component can interact with another component.

Law of Demter. In any Method M attached to a class C, only methods defined by the following classes may be used:

1. The instance variable classes of C.
2. The argument classes of method M (including C); note that global objects or objects created inside the method M are considered arguments to M.

Rewritten in terms of messages

Law of Demeter (weak form). Inside a method, it is only permitted to access or send messages to the following objects:

1. The arguments associated with the method being executed (including the self object).
2. Instance variables for the receiver of the method.
3. Global variables.
4. Temporary variables created inside the method.

What is ruled out

Basically, what is ruled out by the law of demeter is one object going in and directly manipulating the internal data values of another object.

Instead, all access to data values in another component should be made through procedures – thereby reducing data coupling to the weaker parameter coupling.

Class-Level versus Object-Level Visibility

Question: Are sisters and brothers allowed to look at each others private data fields?

An answer of YES is class-level visibility (C++) and answer of NO is object-level visibility.

Active Values

The creation of active values is a good illustration of why parameter coupling is better than direct manipulation.

Suppose we have an existing program and we just want to observe a data value – see when it gets set and changed.

Solution – create a new subclass that just changes those methods that set or read the data value.

Public, Subclass and Private Faces

We have several times noted that object have a public and private face – inheritance introduces a third alternative, the subclass face.

- Public features are those aspects that users of the software component must have access to.
- Private features are those aspects that the implementor of the software component must have access to.
- Protected features are those aspects that implementors of child classes can have access to.

Intentional Dependency

Sometimes want code to depend upon another class.

Sometimes want this even if the dependee doesn't know the dependant. (example, a model and its display).

Can be managed by having a separate “dependency manager”. When an object changes, it tells the manager “notify my dependants”.