14.2 Information hiding and encapsulation

As mentioned, access modifiers may be use to separate representative parts of a program from non-representative parts. The term *information hiding* is often used for a similar purpose. Information hiding is a software design principle that is used to limit accessibility of attributes in a program component.

The term *component* is a general term used for such program elements – in this book a component corresponds to an object-descriptor. The term *client* is often use for components that use a given component. We use the terms component and client below, but a component is as mentioned an object-descriptor, that may define a module, class or singular object.

The primary goal of information hiding is to prevent extensive modification to clients whenever the implementation details of a component are changed. This is done by hiding implementation details that are not relevant for the use of a given component.

The term *encapsulation* is often used in connection to information hiding. One may think of information hiding is a design principle and encapsulation as a technical term. A component like a class is said to *encapsulate* attributes and hiding implementation details. A component hides information by encapsulating the information into a component which presents an interface

The term *interface* of a component is another important term in software design. The interface of a component is the set of attributes that are accessible by clients of the component. The attributes of a given component defined as `%public` thus constitutes the interface of the component. Clients of the interface of a component will access the component purely through its interface. This way, if the implementation changes, the clients do not have to change.

The term *Application Programming Interface* (API) is another common term. It refers to the interface of a component, but is most often used for software modules and/programs used by other modules or programs, but technically it is an interface.

In many schools of software design, it is a common design principle that the interface of a component should be restricted to a set of methods. Data-items are primarily considered implementation details and access to data-items should therefore be limited. In object-oriented design as advocated in this book, an interface may also include local classes and/or intrinsic objects.

Information hiding, encapsulation and interface are technical terms of software design. They go hand inland with programming as modeling. One way of practicing information hiding s to make sure that only representative parts of a module are visible to clients and thus provides the interface. Non-representative parts correspond to implementation. Modules, classes and object-descriptors are language mechanisms for supporting encapsulation supplemented by access modifiers.

Information hiding serves as a criterion that can be used to decompose a program into modules. The principle is also useful for reducing coupling within a system.

A common use of information hiding is to hide the physical storage layout for data so that if it is changed, the change is restricted to a small subset of the total program. For example, if a three-dimensional point (x, y, z) is represented in a program with three floating-point scalar variables and later, the representation is changed to a single array variable of size three, a module designed with information hiding in mind would protect the remainder of the program from such a change.