SIDE BAR: Encapsulation in Ada

The Ada **package** construct was designed to support encapsulation via the use of information hiding. Information hiding is achieved by separating the description of the facilities that a module provides (its functional specification) from the description of how it provides them (its body or implementation). But even with such a well-designed construct, there remains a spectrum of possibilities on just how to use encapsulation in Ada.

Imagine an application that requires lines of text to be read from an input file and broken up into words. The words will be statistically analyzed based on properties such as their length and position. It is natural to think of *word* as being a candidate for encapsulation within a package. This is not just a binary choice; there are a variety of ways that the issue might be addressed in Ada.

- No encapsulation at all. Read, parse, and analyze the words in the context of the controlling (client) code.
- Hide the reading and parsing in a subprogram (GET_NEXT_WORD) that returns a STRING (array of characters) each time it is called.
- Define a new data type. Make WORD a new STRING type returned by GET_NEXT_WORD. Its structure is visible, but the compiler will detect operations on WORDs that should only be performed on STRINGs and vice versa.
- Encapsulate the functionality for dealing with words in a package. The package will make visible (export) a data type (WORD) and one or more subprograms for obtaining and processing them. Within this approach are further options.
  - Publish the representation for WORDs by using an existing Ada data type such as STRING.
  - Use the Ada **private** keyword to hide the representation of the new type but allow assignment and equality tests on WORDs. If an unconstrained array type is used for the representation, this

fact must be published in the public part of the package specification so that users of the package can indicate the length of the WORDs when they are declared.

  - Use the Ada **limited private** keywords to hide the representation and prevent even the use of assignment and equality tests.
  - Use an **access** type (pointer) to hide even the knowledge that an array is holding the words.

- Completely hide the data type. No type is exported. GET_NEXT_WORD reads and parses the next word but returns nothing. Other subprograms within the package assume the existence of a "current" word and perform their analysis functions on it.

The design of Ada provides considerable freedom along this range of design decisions. Most languages allow only a more restricted set of choices, either because they do not support encapsulation at all or because they include features intended to support a particular style of encapsulation to the exclusion of others.