A Book Chapter

By Narendra Kumar Chahar

on **Abstract Data Types and Information Hiding**

Index:

Chapter	Chapter Name	Page No.
10.	Abstract Data Types and Information Hiding	2 - 8

Chapter 10

Abstract Data Type

- An *abstraction* is a view or representation of an entity that includes only the most significant attributes.
- The concept of *abstraction* is fundamental in programming (and computer science)
- Nearly all programming languages support *process abstraction* with subprograms
- Nearly all programming languages designed since 1980 support *data abstraction*

Introduction to Data Abstraction

- An abstract data type is a user-defined data type that satisfies the following two conditions:

  –The representation of, and operations on, objects of the type are defined in a single syntactic unit.

  –The representation of objects of the type is hidden from the program units that use these objects, so the only operations possible are those provided in the type's definition.

Advantages of Data Abstraction

- Advantage of the first condition

  –Program organization, modifiability (everything associated with a data structure is together), and separate compilation.

- Advantage the second condition

  –Reliability--by hiding the data representations, user code cannot directly access objects of the type or depend on the representation, allowing the representation to be changed without affecting user code.

Language Requirements for ADTs

- A syntactic unit in which to encapsulate the type definition
- A method of making type names and subprogram headers visible to clients, while hiding actual definitions.
- Some primitive operations must be built into the language processor .

  Design Issues
- Can abstract types be parameterized?
- What access controls are provided?

Language Examples: Ada

- The encapsulation construct is called a package

  –Specification package (the interface)

  –Body package (implementation of the entities named in the specification)
- Information Hiding

  –The spec package has two parts, public and private

  –The name of the abstract type appears in the public part of the specification package. This part may also include representations of unhidden types.

  –The representation of the abstract type appears in a part of the specification called the private part.
- More restricted form with limited private types

  Private types have built-in operations for assignment and comparison

  Limited private types have NO built-in operations
- Reasons for the public/private spec package:

1. The compiler must be able to see the representation after seeing only the spec package (it cannot see the private part).
2. Clients must see the type name, but not the representation (they also cannot see the private part).

- Having part of the implementation details (the representation) in the spec package and part (the method bodies) in the body package is not good.

Parameterized Abstract Data Types

- Parameterized ADTs allow designing an ADT that can store any type elements (among other things).
- Also known as generic classes.
- C++, Ada, Java 5.0, and C# 2005 provide support for parameterized ADTs Parameterized ADTs in Ada.
- Ada Generic Packages

  –Make the stack type more flexible by making the element type and the size of the stack generic.

```
generic
Max_Size: Positive;
type Elem_Type is private;
package Generic_Stack is
Type Stack_Type is limited private;
function Top(Stk: in out StackType) return Elem_type;
…
end Generic_Stack;
Package Integer_Stack is new Generic_Stack(100,Integer);
Package Float_Stack is new Generic_Stack(100,Float);
```

Parameterized ADTs in C++

•Classes can be somewhat generic by writing parameterized constructor functions

```
class stack {
…
stack (int size) {
stk_ptr = new int [size];
max_len = size - 1;
top = -1;
```

```
};
…
}
stack stk(100);
```

- The stack element type can be parameterized by making the class a templated class

```
template <class Type>
class stack {
private:
Type *stackPtr;
const int maxLen;
int topPtr;
public:
stack() {
stackPtr = new Type[100];
maxLen = 99;
topPtr = -1;
}
…
}
```

Parameterized Classes in Java 5.0

- Generic parameters must be classes
- Most common generic types are the collection types, such as LinkedList and ArrayList
- Eliminate the need to cast objects that are removed
- Eliminate the problem of having multiple types in a structure Parameterized Classes in C# 2005.

- Similar to those of Java 5.0
- Elements of parameterized structures can be accessed through indexing

Summary

- The concept of ADTs and their use in program design was a milestone in the development of languages.
- Two primary features of ADTs are the packaging of data with their associated operations and information hiding.
- Ada provides packages that simulate ADTs
- C++ data abstraction is provided by classes
- Java‘s data abstraction is similar to C++
- Ada, C++, Java 5.0, and C# 2005 support parameterized ADTs.

Information Hiding

Information hiding occurs when a language allows some program entities to be invisible to parts of a program. When applied to data within an object or module, we usually use the more specific term encapsulation. Information hiding is a good thing

since it can make it harder to write certain types of malicious code, and make it harder to accidentally ruin parts of a system you didn't mean to touch.

Here are some different language forms for information hiding.

Explicit information hiding

A direct approach is to use a modifier — a keyword or symbol that defines an access level for an entity. Here are various possible forms:

```
private var x;
protected double x;
private protected var x;
package protected boolean x;
package local var x;
public array of int x;
x: public string;
-int x;
+int x;
-(int) x;
+(int) x;
```

These modifiers can be used to hide a class within a package, a member within a class, or just about anything within a module. Visibility of fields for example, can be restricted to

- The object in which it appears
- All objects of the same class in which it appears
- The package in which it appears
- Its class and all its class's subclasses
- Its class and subclasses and the same package

...and it can have no restrictions at all (this generally uses the modifier public).

For languages that favor symbols over words, you could use something like

- - for most-restricted (private)
- for semi-restricted or (protected or package-private or package-protected)
- + for least-restricted or (public)

A good case can be made for + and - ... not sure about others.

References:

[1] Loeckx, Jacques, Hans-Dieter Ehrich, and Markus Wolf. *Specification of abstract data types*. John Wiley & Sons, Inc., 1997.

[2] Kutzler, Bernhard, and Franz Lichtenberger. *Bibliography on abstract data types*. Vol. 68. Springer Science & Business Media, 2012.

[3] Preguiça, Nuno, Carlos Baquero, and Marc Shapiro. "Conflict-free replicated data types (CRDTs)." *arXiv preprint arXiv:1805.06358* (2018).

[4] Jindian, Su, and Yu Shanshan. "A Bialgebraic Perspective on Abstract Data Types." *International Information Institute (Tokyo). Information* 16.9 (2013): 6549.

[5] Almeida, Paulo Sérgio, Ali Shoker, and Carlos Baquero. "Delta state replicated data types." *Journal of Parallel and Distributed Computing* 111 (2018): 162-173.

[6] Gulisano, Vincenzo, et al. "Efficient data streaming multiway aggregation through concurrent algorithmic designs and new abstract data types." *ACM Transactions on Parallel Computing (TOPC)* 4.2 (2017): 1-28.

[7] Parah, Shabir A., et al. "Information hiding in edges: A high capacity information hiding technique using hybrid edge detection." *Multimedia Tools and Applications* 77.1 (2018): 185-207.

[8] Sumathi, C. P., T. Santanam, and G. Umamaheswari. "A study of various steganographic techniques used for information hiding." *arXiv preprint arXiv:1401.5561* (2014).

[9] Parah, Shabir A., et al. "Information hiding in edges: A high capacity information hiding technique using hybrid edge detection." *Multimedia Tools and Applications* 77.1 (2018): 185-207.