Introduction: Abstract Data Types

Contents

Good Programs Use Abstraction

What makes a program ***good***?

1. It works (as specified!).
2. It is easy to understand and modify.
3. It is reasonably efficient.

One way to help achieve (2) (which helps with (1)) is to use **abstract data types**, or **ADTs**. The idea of an ADT is to separate the notions of **specification** (what kind of thing we're working with and what operations can be performed on it) and **implementation** (how the thing and its operations are actually implemented).

Benefits of using Abstract Data Types

- Code is easier to understand (e.g., it is easier to see "high-level" steps being performed, not obscured by low-level code).
- Implementations of ADTs can be changed (e.g., for efficiency) without requiring changes to the program that uses the ADTs.
- ADTs can be reused in future programs.

Fortunately for us, object-oriented programming languages (like Java) make it easy for programmers to use ADTs: each ADT corresponds to a **class** (or **Java interface** - more on this later) and the operations on the ADT are the class/interface's **public methods**. The user, or client, of the ADT only needs to know about the method **interfaces** (the names of the methods, the types of the parameters, what the methods do, and what, if any, values they return), not the actual implementation (how the methods are implemented, the private data members, private methods, etc.).

Abstract Data Types

There are two parts to each ADT:

1. The **public** or **external** part, which consists of:
    - the conceptual picture (the user's view of what the object looks like, how the structure is organized)
    - the conceptual operations (what the user can do to the ADT)
2. The **private** or **internal** part, which consists of:
    - the representation (how the structure is actually stored)
    - the implementation of the operations (the actual code)

In general, there are many possible operations that could be defined for each ADT; however, they often fall into these categories:

1. initialize
2. add data
3. access data
4. remove data

In this class, we will study a number of different abstract data types, different ways to implement them, and different ways to use them. Our first ADT (coming up in the next set of notes) is the List.

Back

Next: Continue with Lists