# Chapter 4

# Data Abstraction: The Walls

## Abstract Data Types

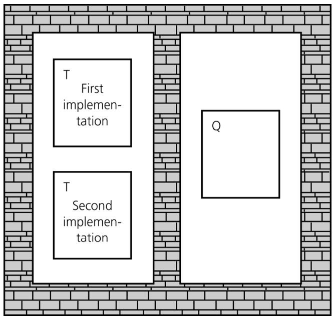


Figure 4-1

Isolated tasks: the implementation of task *T* does not affect task *Q*

## Abstract Data Types

- The isolation of modules is not total
  - Methods' specifications, or contracts, govern how they interact with each other

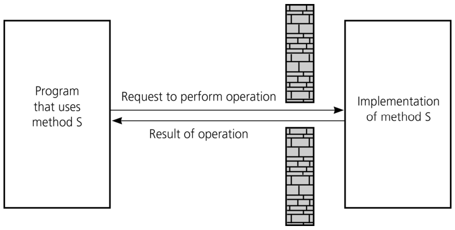

Program that uses method S
Request to perform operation
Result of operation
Implementation of method S

Figure 4-2

A slit in the wall

## Abstract Data Types

- Abstract data type (ADT)
  - An ADT is composed of
    - A collection of data
    - A set of operations on that data
  - Specifications of an ADT indicate
    - What the ADT operations do, not how to implement them
  - Implementation of an ADT
    - Includes choosing a particular data structure

## Abstract Data Types

- Data structure
  - A construct that is defined within a programming language to store a collection of data
  - Example: arrays
- ADTs and data structures are not the same
- Data abstraction
  - Results in a wall of ADT operations between data structures and the program that accesses the data within these data structures

## Abstract Data Types

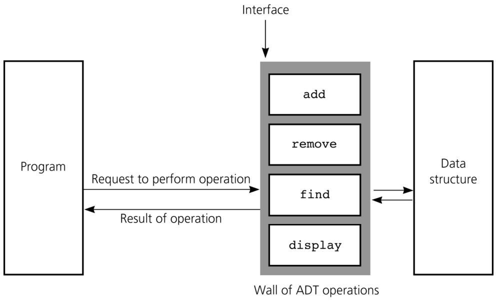
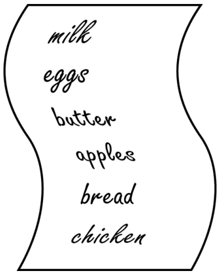


Figure 4-4

A wall of ADT operations isolates a data structure from the program that uses it

## Specifying ADTs



- In a list
  - Except for the first and last items, each item has
    - A unique predecessor
    - A unique successor
  - Head or front
    - Does not have a predecessor
  - Tail or end
    - Does not have a successor

Figure 4-5

A grocery list

## Implementing ADTs

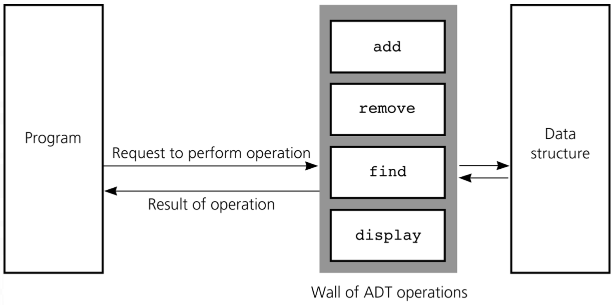


Figure 4-8

ADT operations provide access to a data structure

## Implementing ADTs

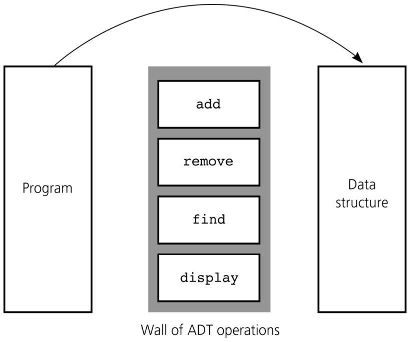


Figure 4-9

Violating the wall of ADT operations

## Java Classes Revisited

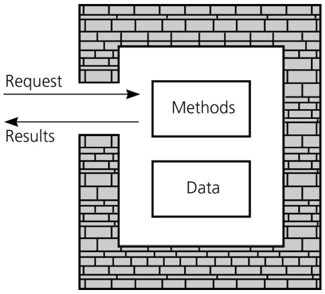


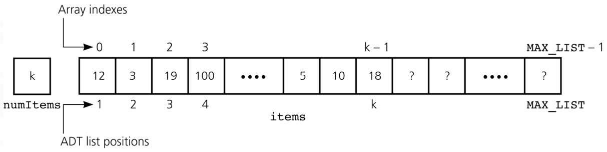
Figure 4-10

An object's data and methods are encapsulated

## An Array-Based Implementation of the ADT List

- An array-based implementation
  - A list's items are stored in an array `items`
  - A natural choice
    - Both an array and a list identify their items by number
  - A list's k$^{th}$ item will be stored in `items[k-1]`

## An Array-Based Implementation of the ADT List



| Array indexes | 0 | 1 | 2 | 3 | | | | k – 1 | | | | MAX_LIST – 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| items | 12 | 3 | 19 | 100 | •••• | 5 | 10 | 18 | ? | ? | •••• | ? |
| ADT list positions | 1 | 2 | 3 | 4 | | | | k | | | | MAX_LIST |

Figure 4-11

An array-based implementation of the ADT list