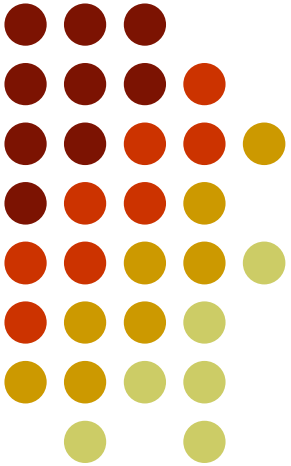
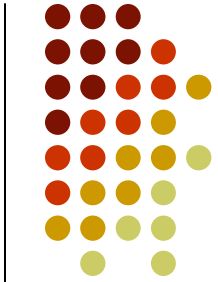
CMPT 225

Abstract Data Types

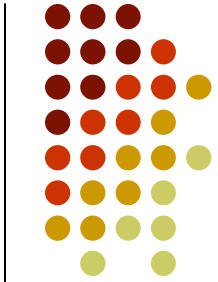


Abstract Data Types



- Typical operations on data
  - Add data to a data collection
  - Remove data from a data collection
  - Ask questions about the data in a data collection
- Data abstraction
  - Asks you to think *what* you can do to a collection of data independently of *how* you do it
  - Allows you to develop each data structure in relative isolation from the rest of the solution
  - A natural extension of procedural abstraction

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
    - A data structure is a construct that can be defined in a programming language to store a collection of data

Abstract Data Types

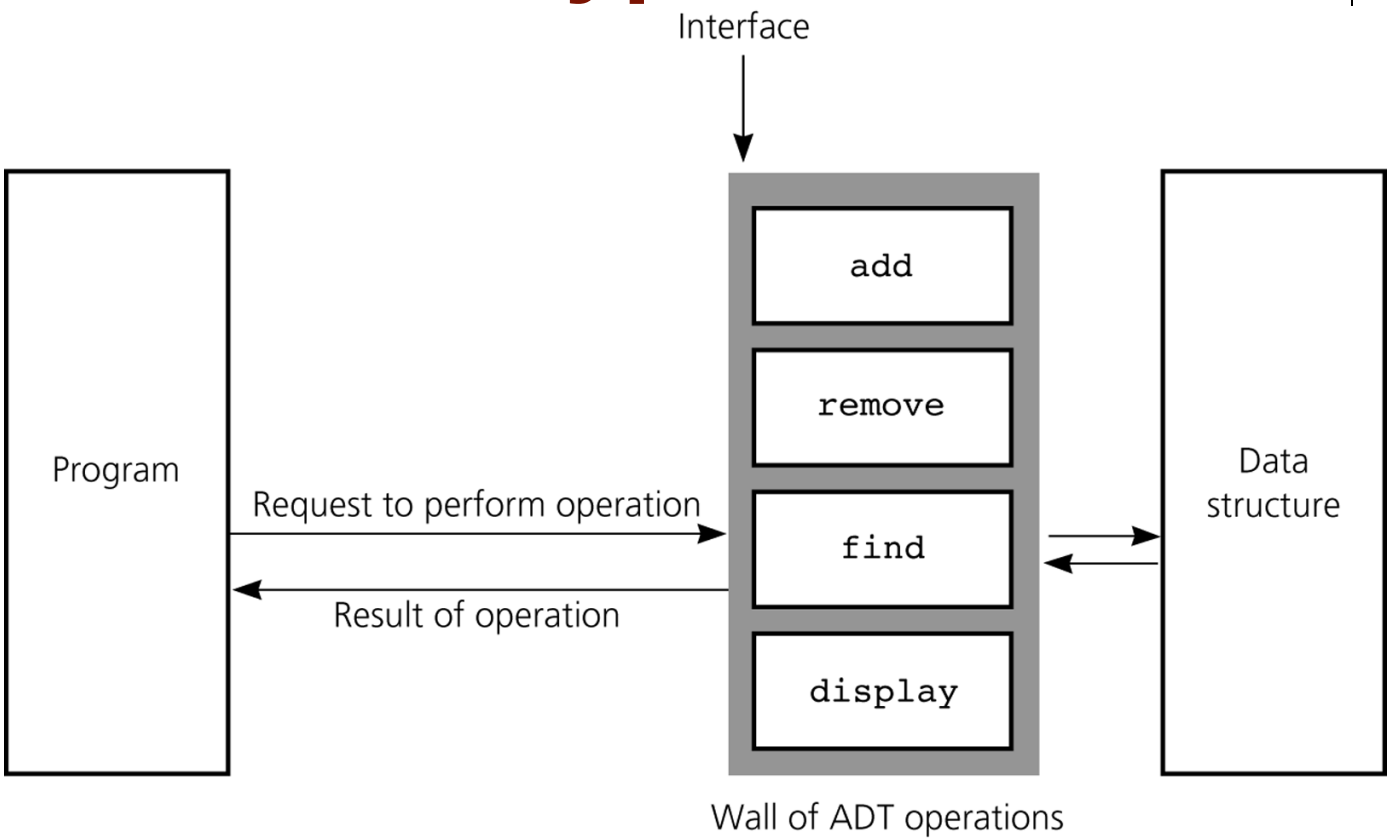


Figure 4-4

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

Violating the wall of ADT

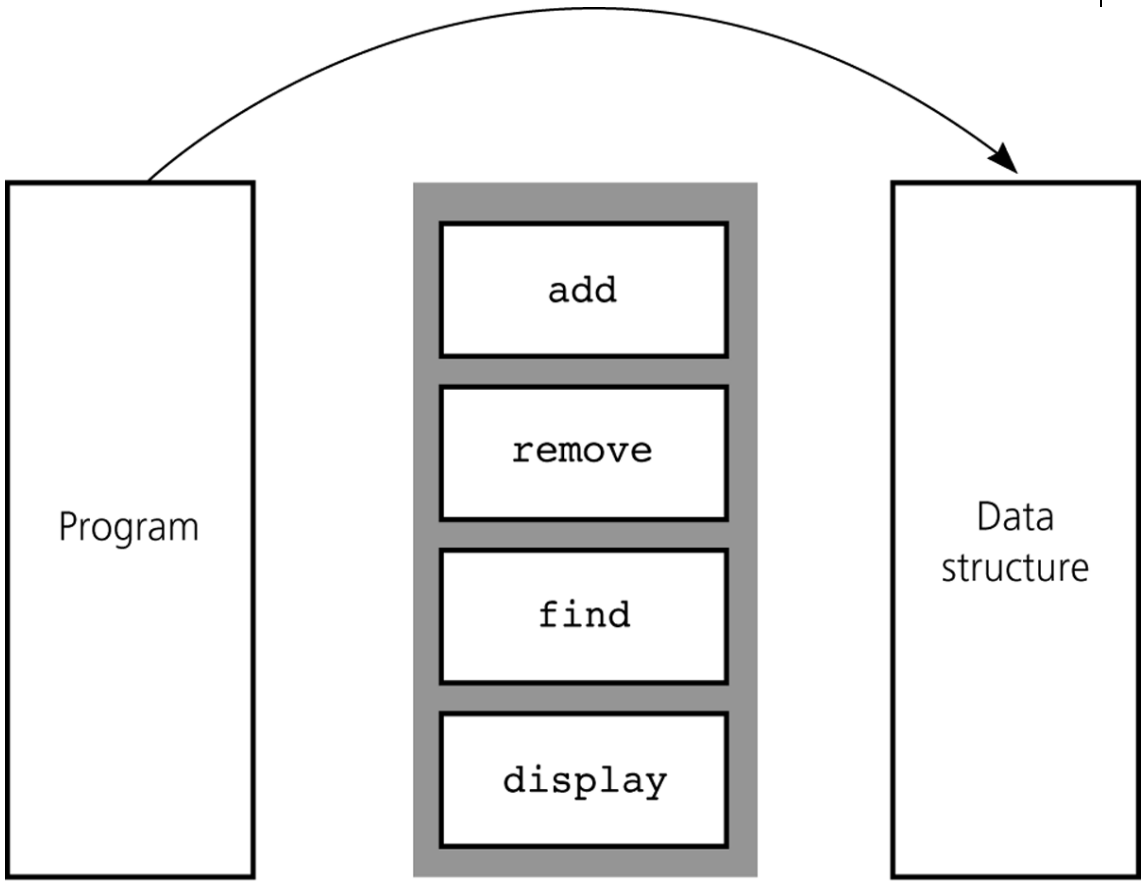


Figure 4-9

Violating the wall of ADT operations

Specifying ADTs

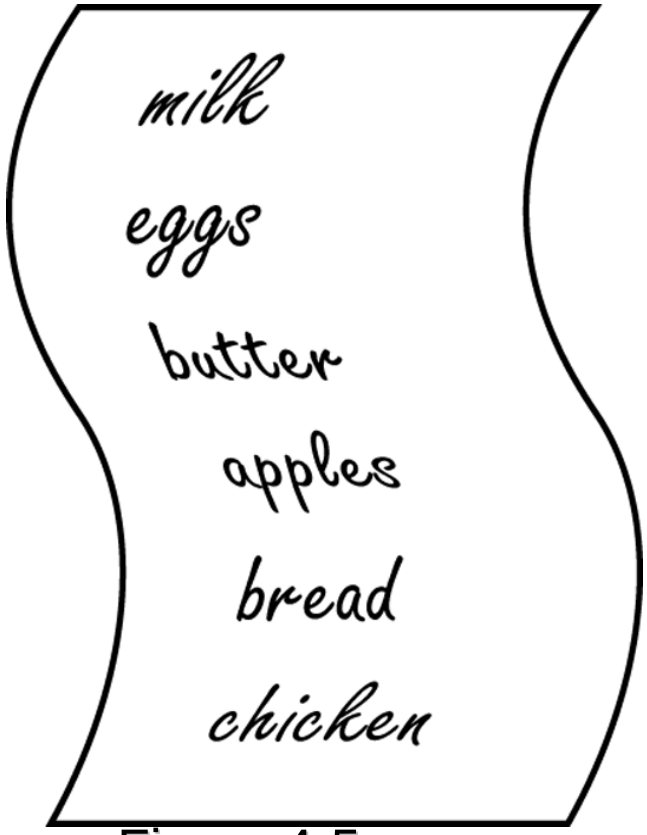
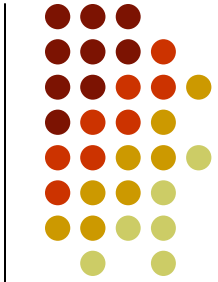


Figure 4-5

list A grocery

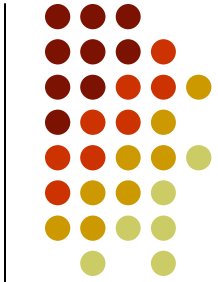
- list:
  - Except for the first and last items, each item has
    - A unique predecessor
    - A unique successor
  - Head or front
    - Does not have a predecessor
  - Tail or end
    - Does not have a successor

Designing ADT List



- Items are referenced by their position within the list
- ADT List operations
  - Create an empty list
  - Determine whether a list is empty
  - Determine the number of items in a list
  - Add an item at a given position in the list
  - Remove the item at a given position in the list
  - Remove all the items from the list
  - Retrieve (get) the item at a given position in the list

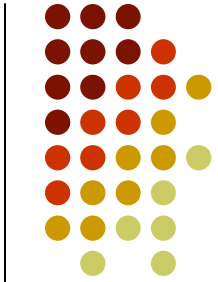
The ADT List



- Java provide convenient tool for specifying ADT **interface**
  Example: ListInterface.java

- Specifications of the ADT operations
  - Define the contract for the ADT list
  - Do not specify how to store the list or how to perform the operations

Client of ADT List

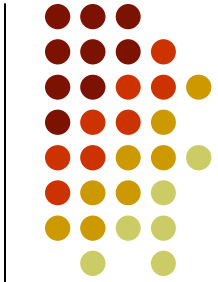


- ADT operations can be used in an application without the knowledge of how the operations will be implemented

- Example: write an algorithm swap(L,i,j), swapping elements at positions i and j in L

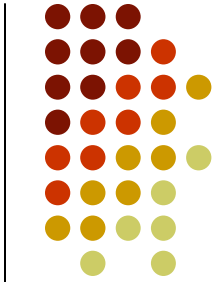
Recall operations of ADT List: createList, isEmpty, size, removeAll, add, get, remove

Axioms



- For complex abstract data types, the behavior of the operations must be specified using axioms
  - Axiom: A mathematical rule – invariant for ADT operations

Axioms



- Axioms for the ADT List L
  1. L.createList().size() = 0
  2. L.add(i, x).size() = L.size() + 1
  3. L.remove(i).size() = L.size() – 1
  4. L.createList().isEmpty() = true
  5. L.add(i, item).isEmpty() = false
  6. L.createList().remove(i) = error
  7. L.add(i, x).remove(i) = L
  8. L.createList().get(i) = error
  9. L.add(i, x).get(i) = x
  10. L.get(i) = L.add(i, x).get(i+1)
  11. L.get(i+1) = L.remove(i).get(i)