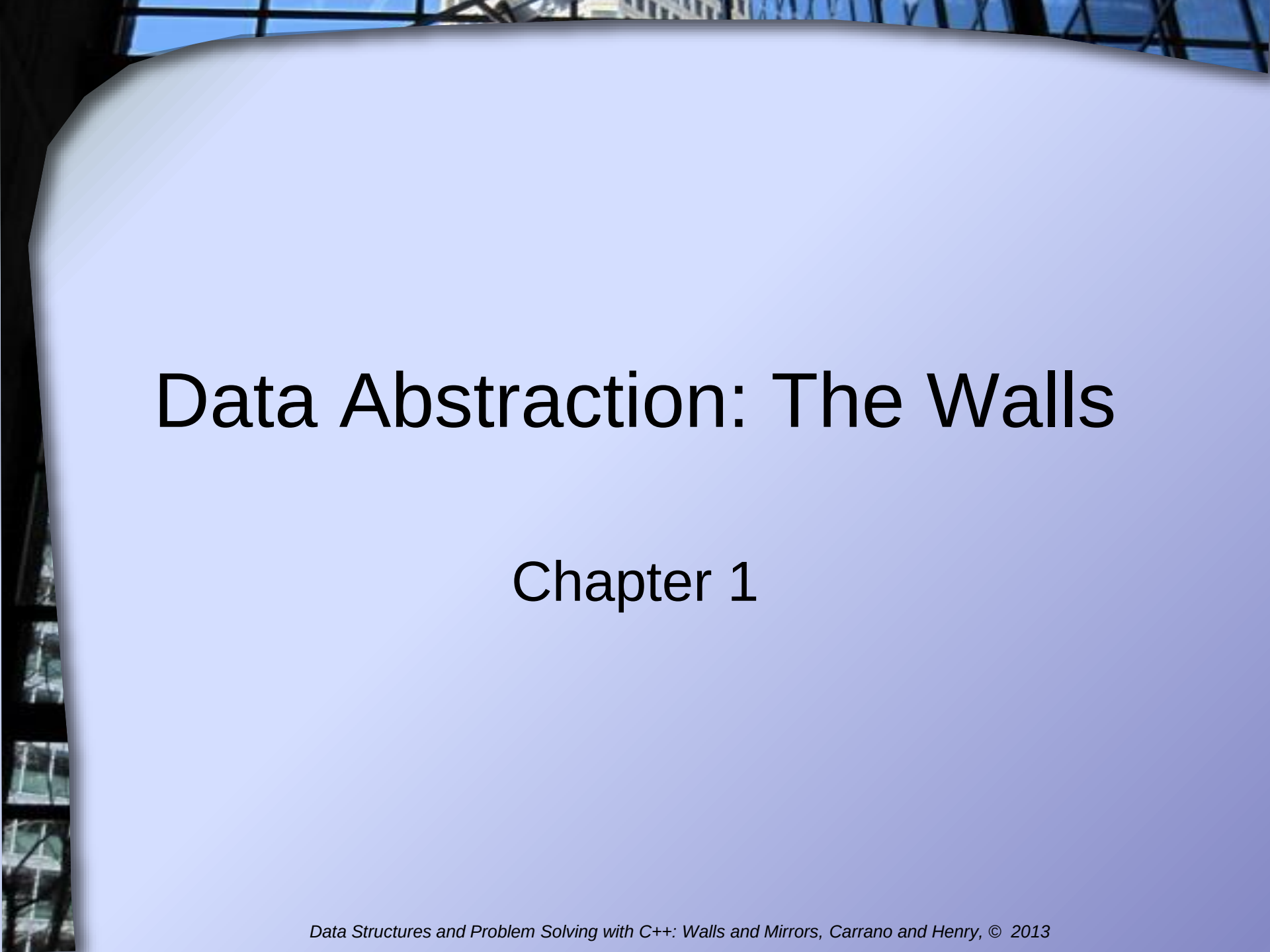


# Data Abstraction: The Walls

## Chapter 1

# Contents

- Object-Oriented Concepts
- Achieving a Better Solution
- Specifications
- Abstract Data Types
- The ADT Bag

# Object-Oriented Concepts

- Code using a solution design
- Specify a system of interacting objects
- Object-oriented *analysis* specifies
  - What to do
  - Not how to do it
- Object-oriented *design* specifies
  - Models of how it might be done

# Object-Oriented Solution

- Create a good set of modules
  - Store, move, alter data
  - Communicate with one another
- Use classes of objects
  - Combines attributes and behaviors

# Principles of Object-Oriented Programming

- Encapsulation: Objects combine data and operations.
- Inheritance: Classes inherit properties from other classes.
- Polymorphism: Objects determine appropriate operations at execution.

# Achieving a Better Solution

- Cohesive modules perform single well-defined tasks
- Coupling – measure of dependence among modules
  - Loosely coupled modules desired

# Achieving a Better Solution

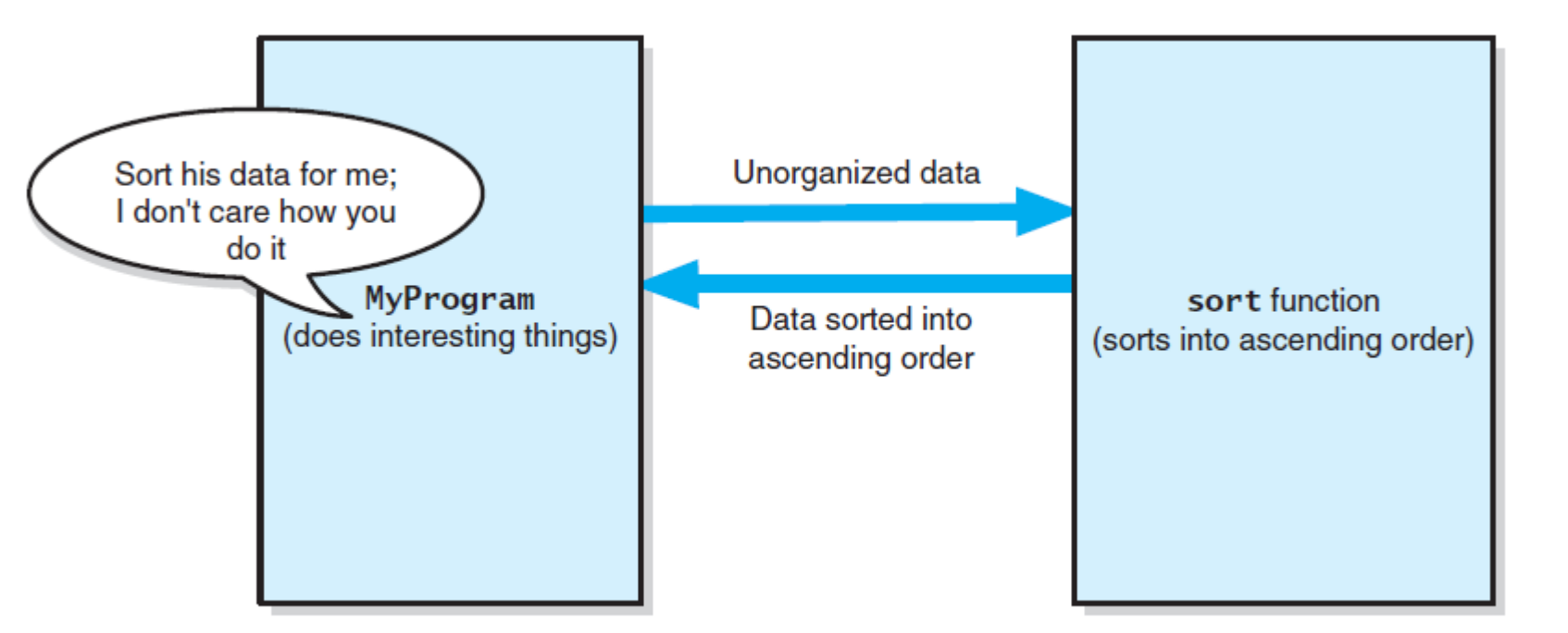



FIGURE 1-1 The task **`sort`** is a module separate from the **`MyProgram`** module

# Operation Contract

- Documents use and limitations of a method
- Specifies data flow
- Does not specify *how* module will perform its task
- Specifies pre- and post-conditions

# Unusual Conditions

Options

- Assume they never happen
- Ignore invalid situations
- Guess at client's intent
- Return value that signals a problem
- Throw an exception

# Abstraction

- Separates purpose of a module from its implementation
- Possible to use a module without knowing implementation
- Think “what” not “how”

# Information Hiding

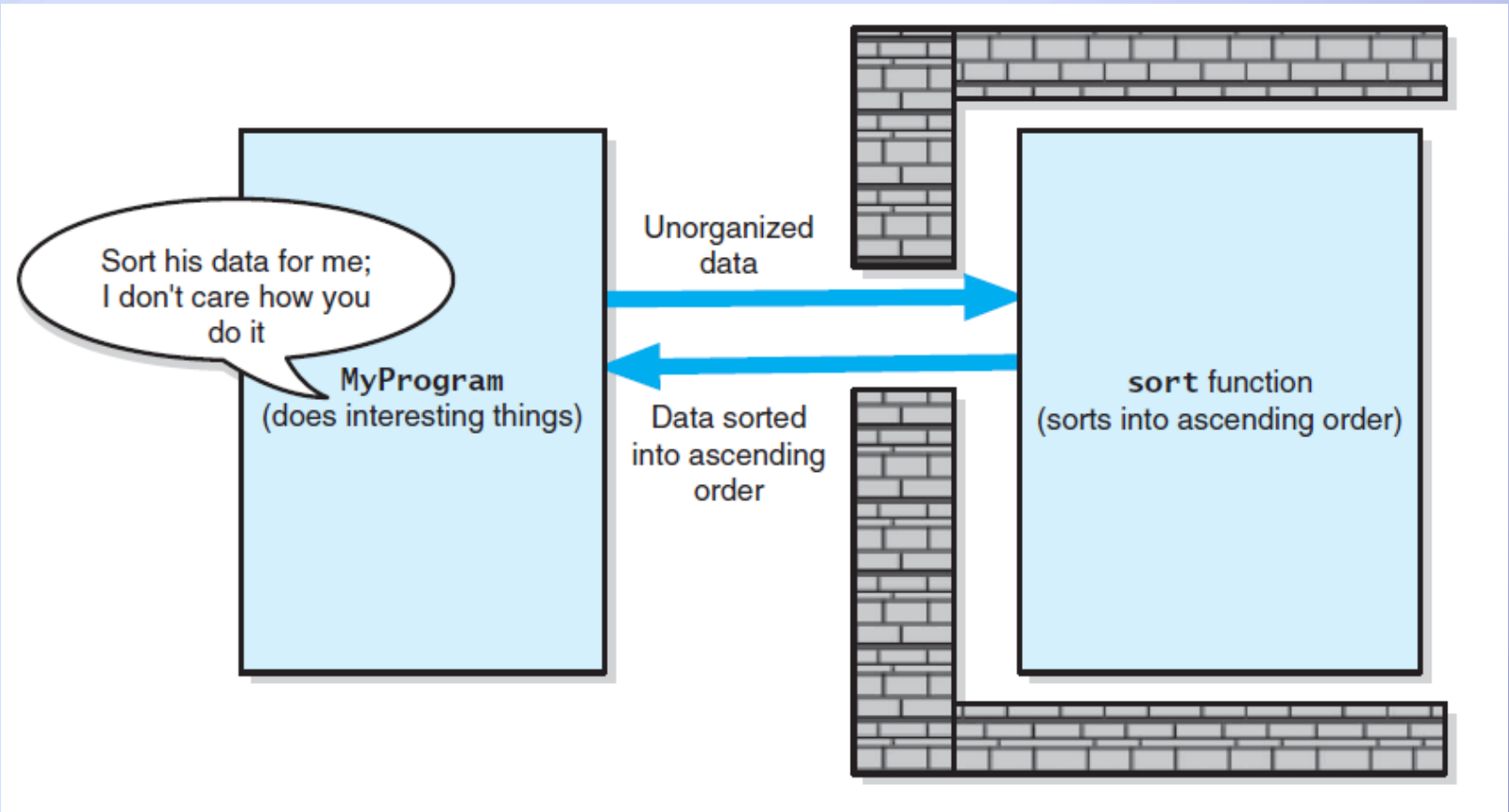



FIGURE 1-2 Tasks communicate through a slit in wall

# Minimal and Complete Interfaces

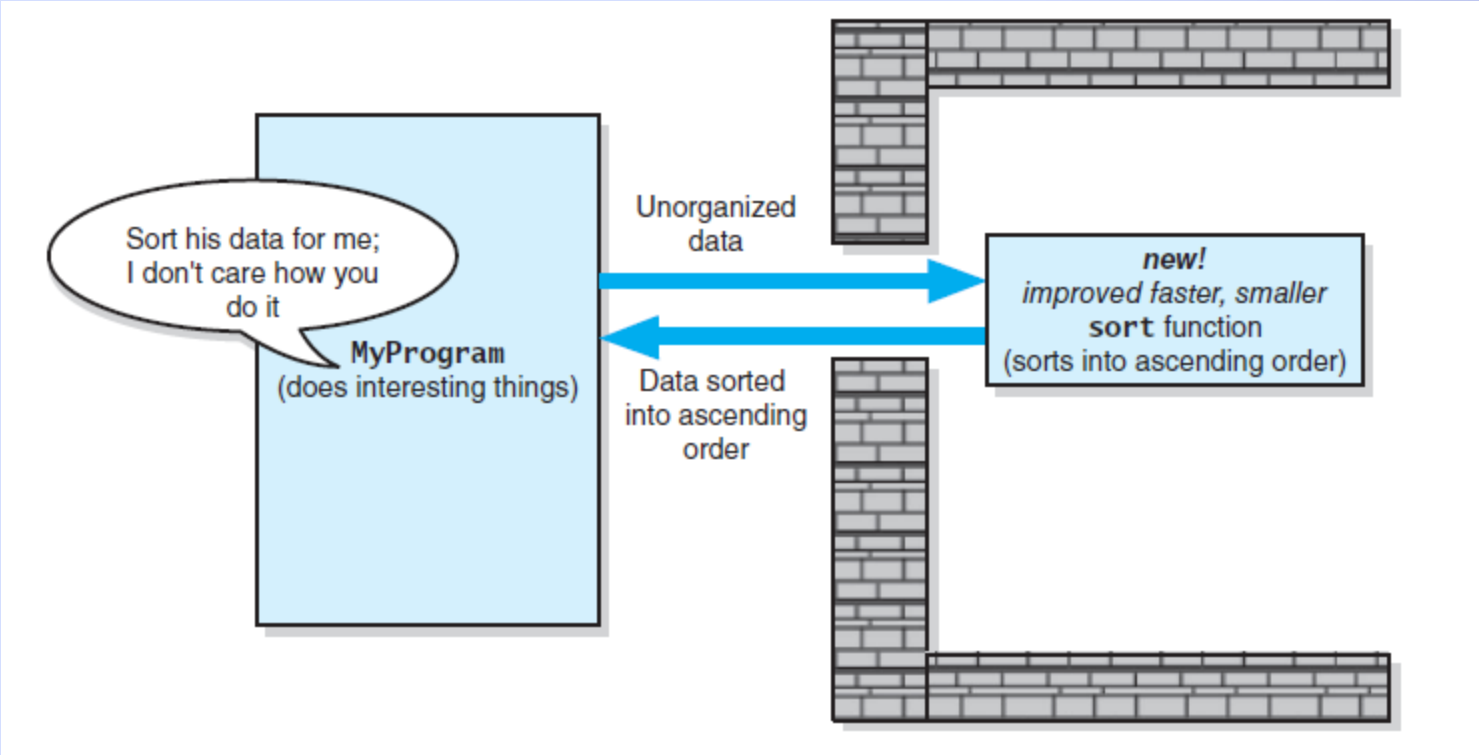


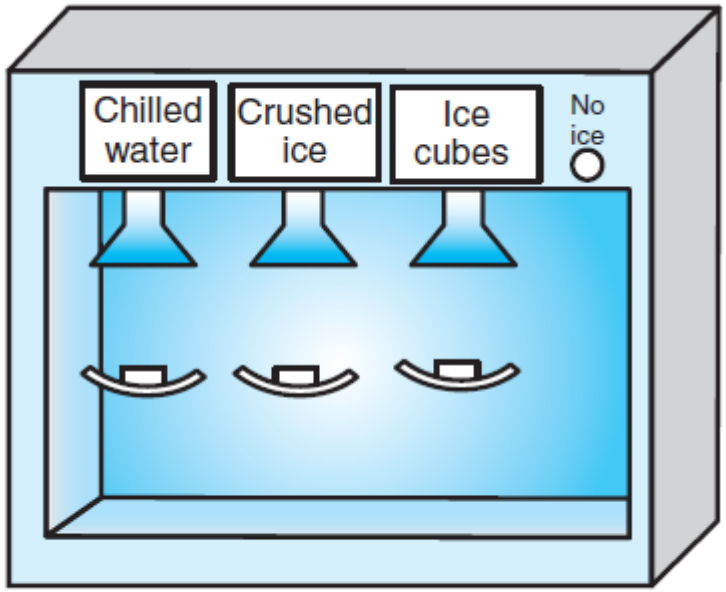

FIGURE 1-3 A revised implementation communicates through the same slit in the wall

# Abstract Data Type

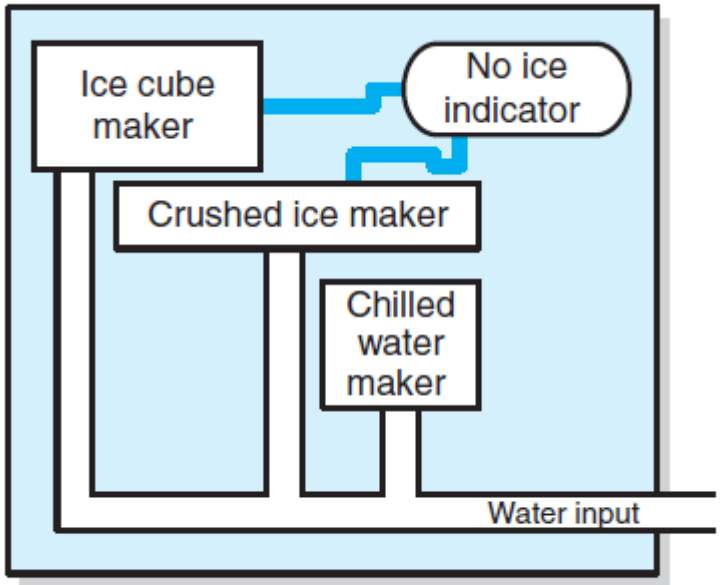
- A collection of data and
- A set of operations on the data.

- Carefully specify an ADT's operations before you implement them

# Abstract Data Type




User view from specifications




Technician view

FIGURE 1-4 A dispenser of chilled water, crushed ice, and ice cubes

# Abstract Data Type

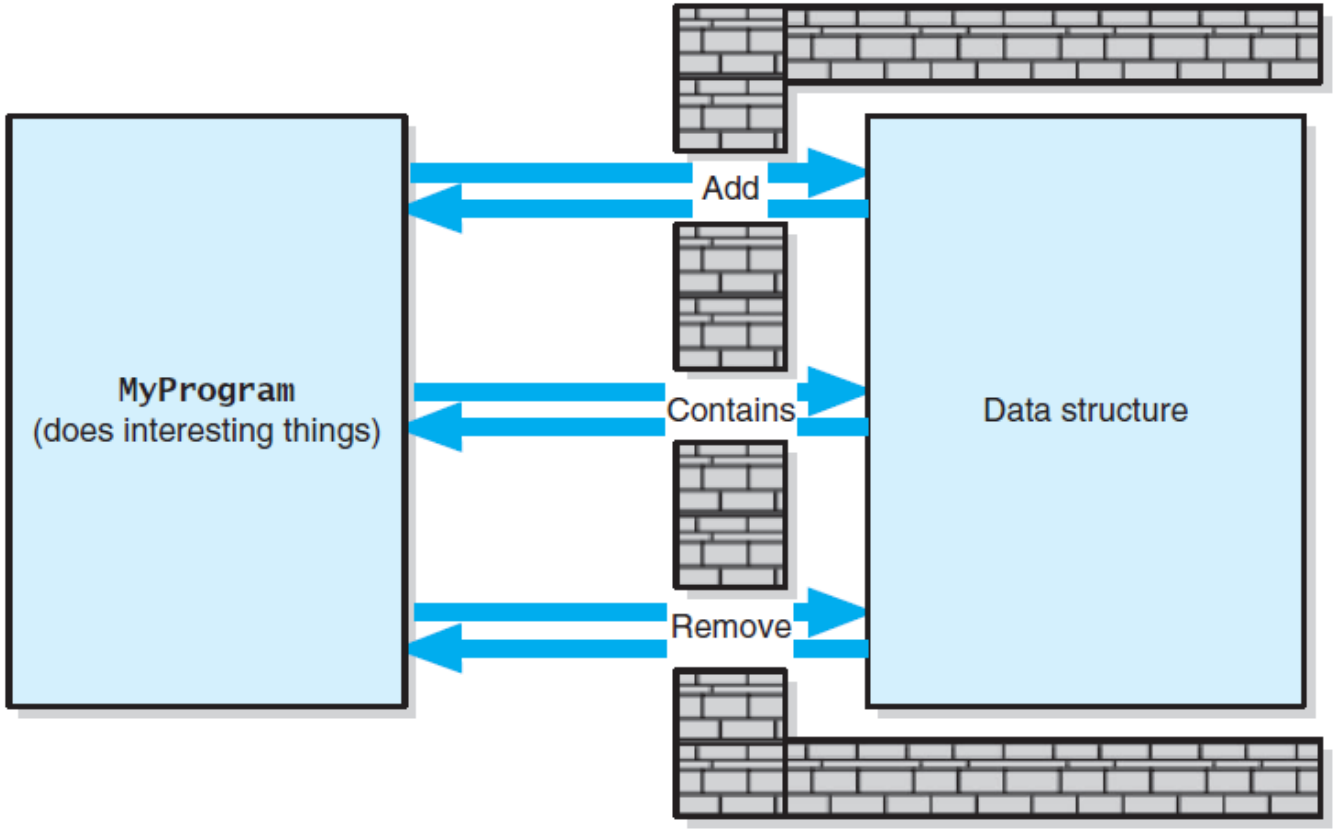



FIGURE 1-5 A wall of ADT operations isolates a data structure from the program that uses it

# Designing an ADT

Ask the questions

- What data does the problem require?
  - Names
  - IDs
  - Numerical data
- What operations will be done on that data?
  - Initialize
  - Display
  - Calculations

# The ADT Bag

- A bag is a container
  - Contains finite number of data objects
  - All objects of same type
  - Objects in no particular order
  - Objects may be duplicated

# Identifying Behaviors

- Get the number of items currently in the bag.
- See whether the bag is empty.
- Add a given object to bag.
- Remove occurrence of specific object from bag
- Remove all objects from bag.

# Identifying Behaviors

- Count the number of times certain object occurs in bag.
- Test whether bag contains particular object.
- Look at all objects in bag.

# Identifying Behaviors

| Bag |
| --- |
| Responsibilities |
| Get the number of items currently in the bag |
| See whether the bag is empty |
| Add a given object to the bag |
| Remove an occurrence of a specific object from the bag, if possible |
| Remove all objects from the bag |
| Count the number of times a certain object occurs in the bag |
| Test whether the bag contains a particular object |
| Look at all objects that are in the bag |
| |
| Collaborations |
| The class of objects that the bag can contain |
| |
| |

FIGURE 1-6 A CRC card for a class **`Bag`**

# Specifying Data and Operations

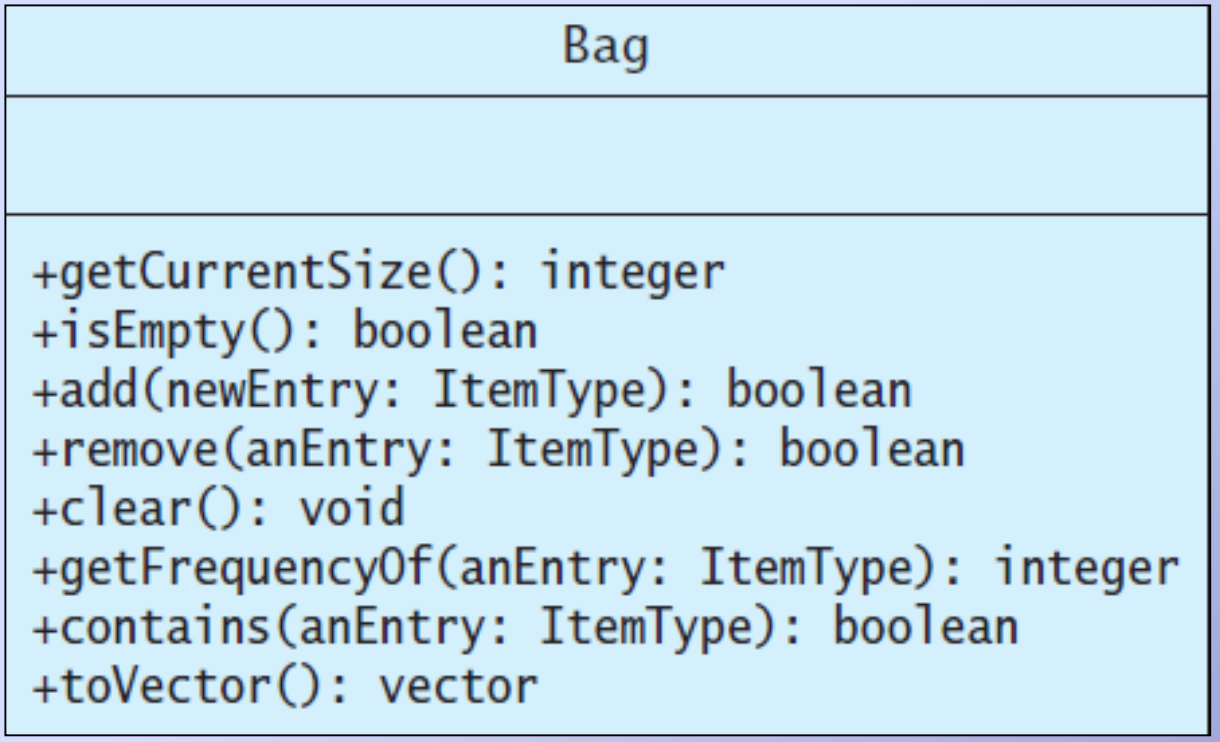
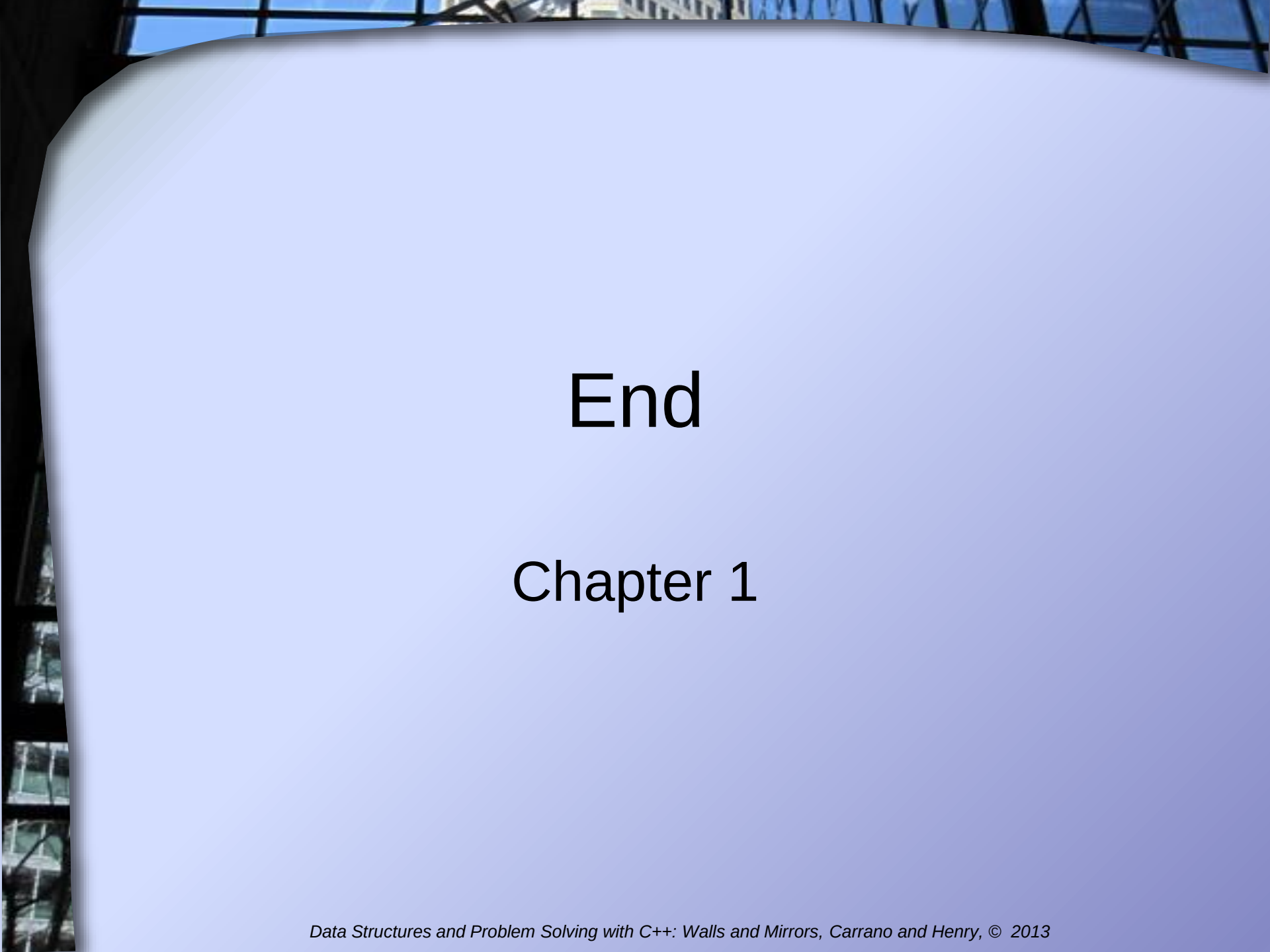



FIGURE 1-7 UML notation for the class **`Bag`**

# The ADT Bag

- View code listing for Bag interface,
  - Listing 1-1
- Note use of ADT Bag, Program for Card Guessing
  - Listing 1-2

.htm code listing files must be in the same folder as the .ppt files for these links to work



# End

## Chapter 1