EECE 2560:
Fundamentals of Engineering Algorithms

Abstract Data Types

Abstraction

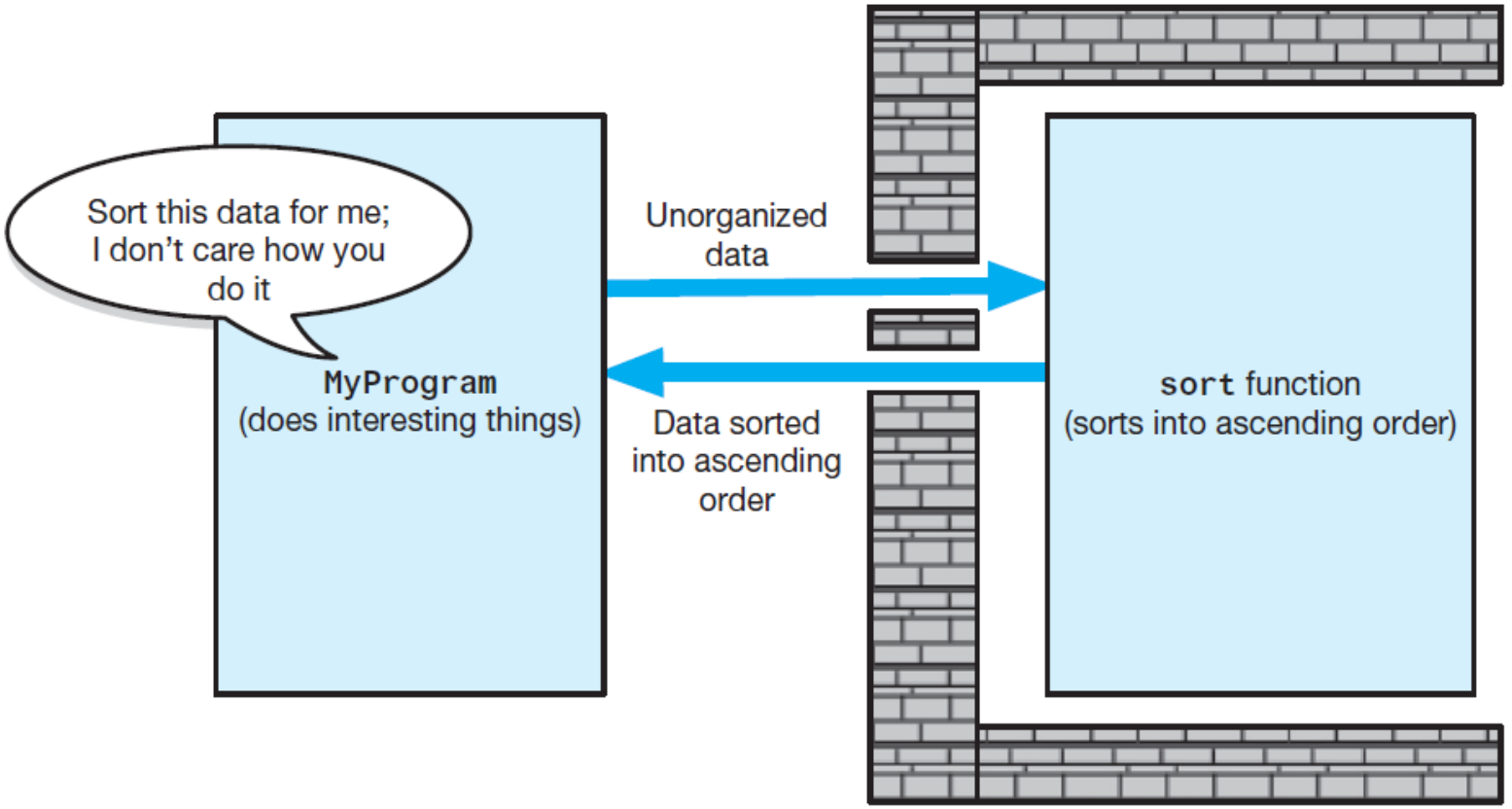
- Separate purpose of a module from its implementation
- Specifications do not indicate how to implement
  - Able to use without knowing implementation

Information Hiding (1 of 3)

- Abstraction helps identify details that should be hidden from public view
  - Ensured no other module can tamper with these hidden details.
- Isolation of the modules cannot be total, however
  - Client programmer must know what tasks can be done, how to initiate a task
- Interface for a class made up of publicly accessible methods and data that allow the client programmer to accomplish class's essential tasks.

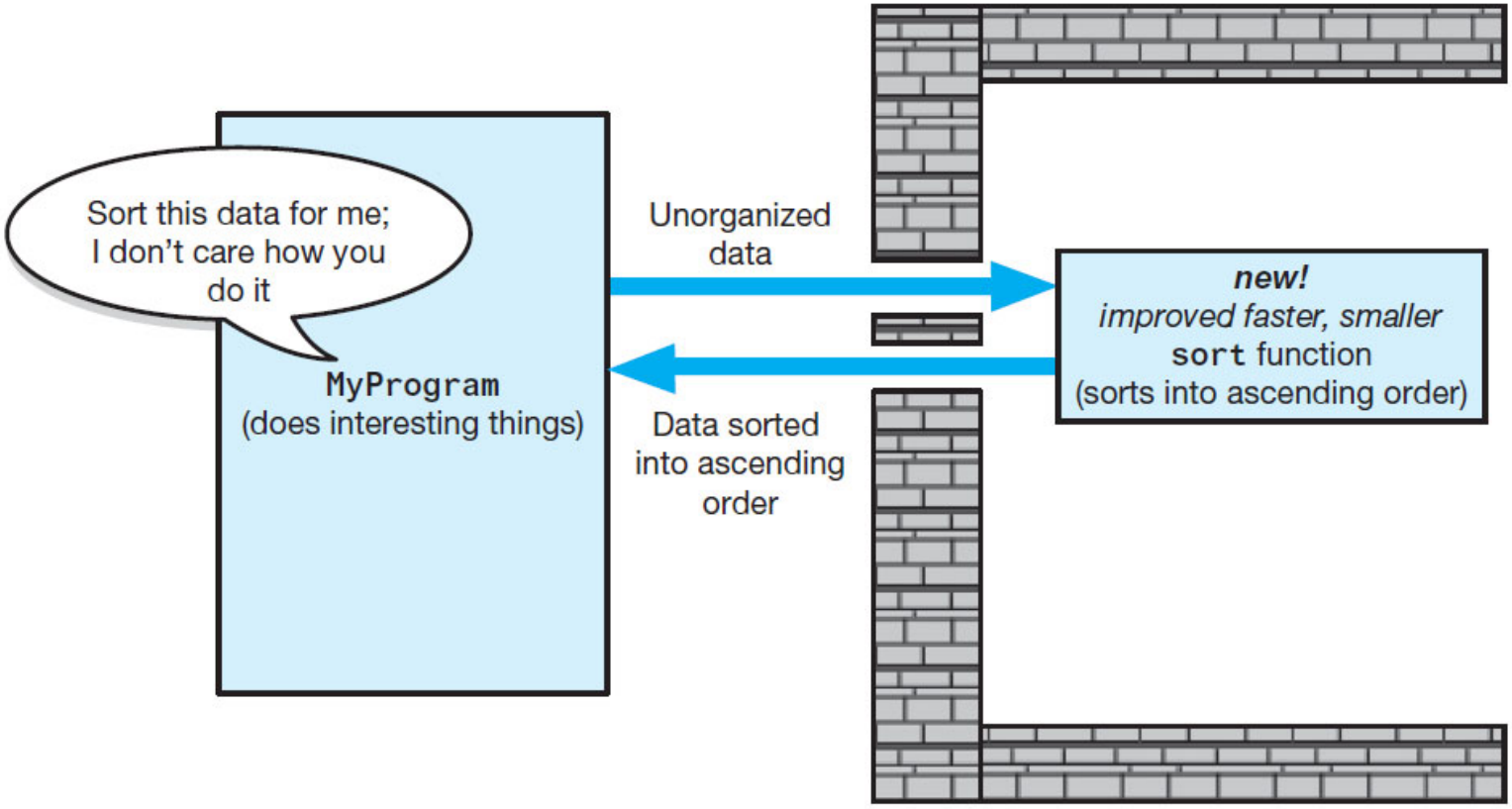
Information Hiding (2 of 3)

- Tasks communicate through a slit in the wall:



Information Hiding (3 of 3)

- A revised implementation communicates through the same slit in the wall

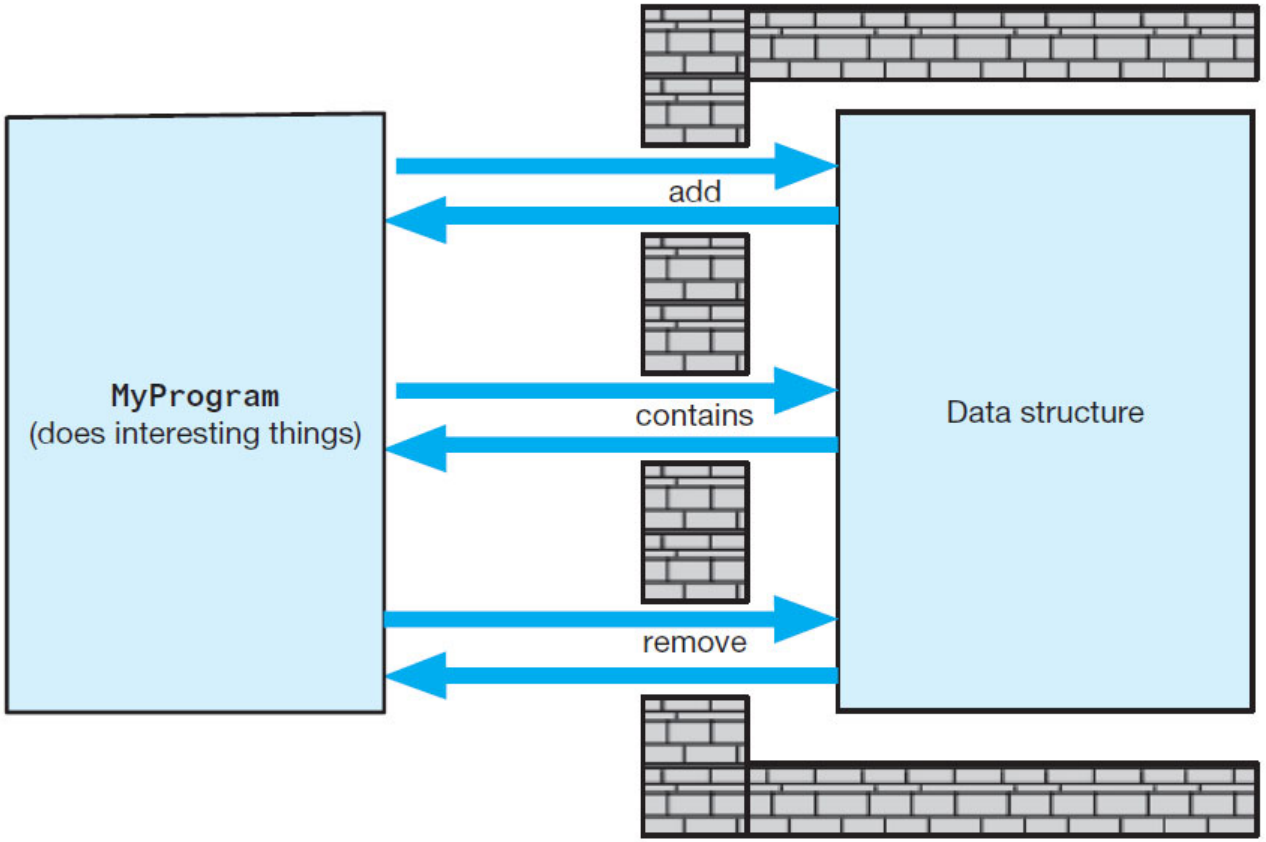


Abstract Data Types (ADT) (1 of 2)

- An ADT : a collection of data **and** a set of operations on data
- 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.
- A data structure : an implementation of an ADT within a programming language.

ADT (2 of 2)

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



Designing an ADT

- Evolves naturally during the problem-solving process
  - What data does a problem require?
  - What operations does a problem require?

- ADTs typically have initialization and destruction operations
  - Assumed but not specified at this stage

The ADT Bag

- Consider the bag to be an abstract data type.
  - We are specifying an abstraction inspired by an actual physical bag
  - Doesn't do much more than contain its items
  - Can unordered and possibly duplicate objects
  - We insist objects be of same or similar types

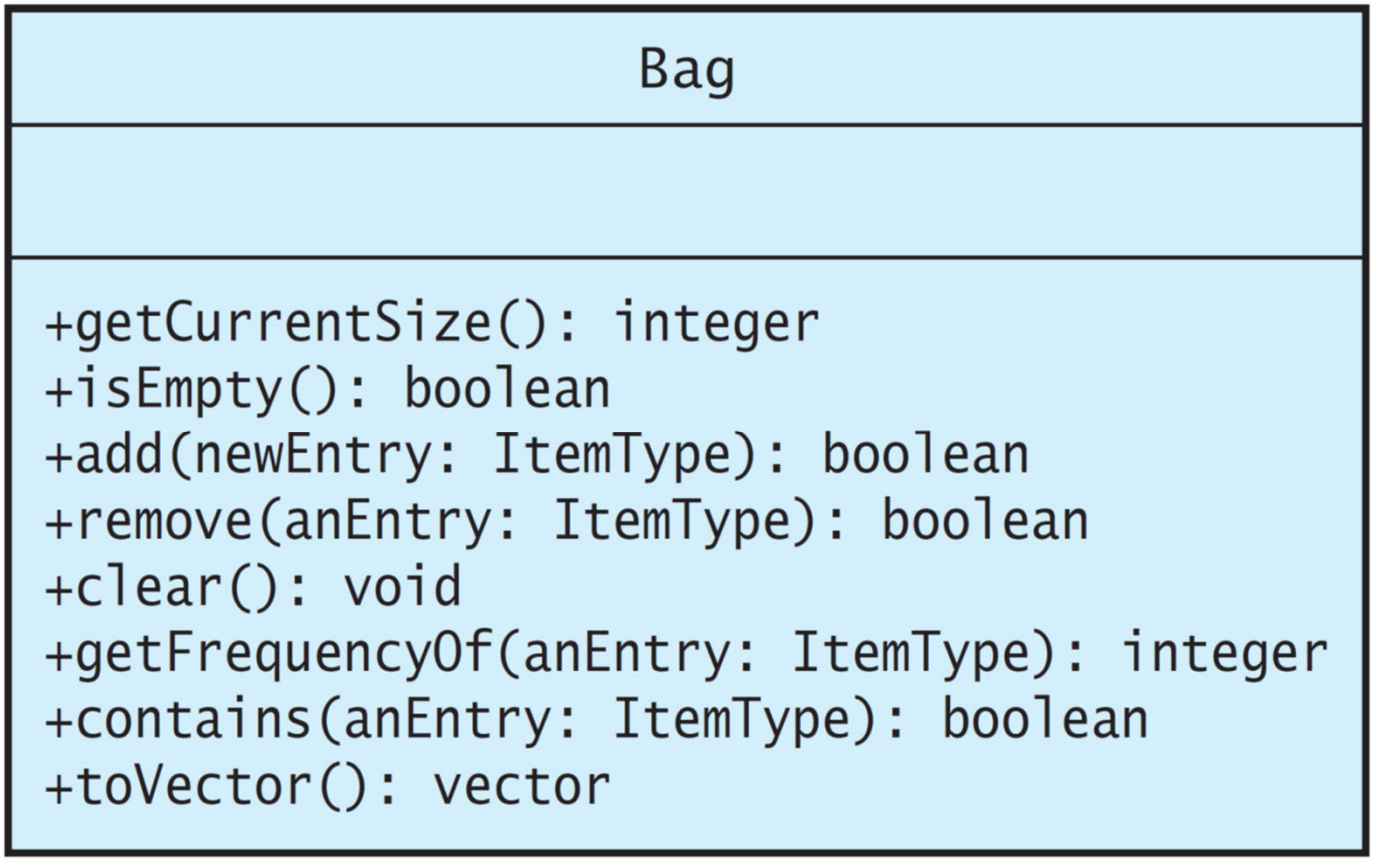
- Knowing just its interface
  - Can use ADT bag in a program

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

Specifying Data and Operations

- UML (Unified Modeling Language) notation for the class Bag



ADT Bag Header File (1 of 2)

```
template<class ItemType>
class Bag
{
 private:
   static const int DEFAULT_BAG_SIZE = 6;
   ItemType items[DEFAULT_BAG_SIZE]; // array of bag items
   int itemCount;                    // current count of bag items
   int maxItems;                     // max capacity of the bag

   // Returns either the index of the element in the array items that
   // contains the given target or -1, if the array does not contain
   // the target.
   int getIndexOf(const ItemType& target) const;
```

ADT Bag Header File (2 of 2)

```
public:
        Bag();
        int getCurrentSize() const;
        bool isEmpty() const;
        bool add(const ItemType& newEntry);
        bool remove(const ItemType& anEntry);
        void clear();
        bool contains(const ItemType& anEntry) const;
        int getFrequencyOf(const ItemType& anEntry) const;
    std::vector<ItemType> toVector() const;
};  // end Bag
```

Class Static and Const Members

- A class member (attribute or method) that is defined as `static` is shared across all the objects instances of the class. This means for all objects of the class there is only one copy of the static member.
  - Example of accessing the static member:
    <class name>::<static member name> = 50; or
    <object name>.<static member name> = 50;
  - A static, method scope variable only gets initialized once.
- Applying the `const` operator to a class attribute members makes the attribute effectively read only and it can be initialized once at its definition line.
- Applying `const` to a class method prevents the method from modifying any class member attributes.

Using the ADT Bag (1 of 2)

```
#include <iostream> // For cout and cin
#include <string>   // For string objects
#include "Bag.h"    // For ADT bag

int main()
{
    std::string clubs[] = { "Joker", "Ace", "Two", "Three", "Four",
                            "Five", "Six", "Seven", "Eight", "Nine",
                            "Ten", "Jack", "Queen", "King" };

   // Create our bag to hold cards.
   Bag<std::string> grabBag;

   // Place six cards in the bag.
   grabBag.add(clubs[1]);
   grabBag.add(clubs[2]);
   grabBag.add(clubs[4]);
   grabBag.add(clubs[8]);
   grabBag.add(clubs[10]);
   grabBag.add(clubs[12]);
```

Using the ADT Bag (2 of 2)

```
   // Get friends guess and check it.
    int guess = 0;
    while (!grabBag.isEmpty())
    {
       std::cout << "What is your guess? (1 for Ace to 13 for King):";
       std::cin >> guess;

       // Is card in the bag?
       if (grabBag.contains(clubs[guess])) {
          // Good guess - remove card from the bag.
          std::cout << "You get the card!\n";
          grabBag.remove(clubs[guess]);
       }
       else {
          std::cout << "Sorry, card was not in the bag.\n";
       }  // end if
    }  // end while

    std::cout << "No more cards in the bag. Game over!\n";
    return 0;
}; // end main
```