Lists

Chapter 12

Data Structures and Abstractions with Java, 4e, Global Edition
Frank Carrano

Lists

- A way to organize data
- Examples
  - To-do list
  - Gift lists
  - Grocery Lists
- Items in list have position
  - May or may not be important
- Items may be added anywhere

Lists

FIGURE 12-1 A to-do list

Specifications for the ADT List

```
add (newEntry)
add (newPosition,
   newEntry)
remove(givenPosition)
clear()
replace(
   givenPosition,
   newEntry)
```

```
getEntry(
   givenPosition)
toArray()
contains(anEntry)
getLength()
isEmpty()
```

Specifications for the ADT List

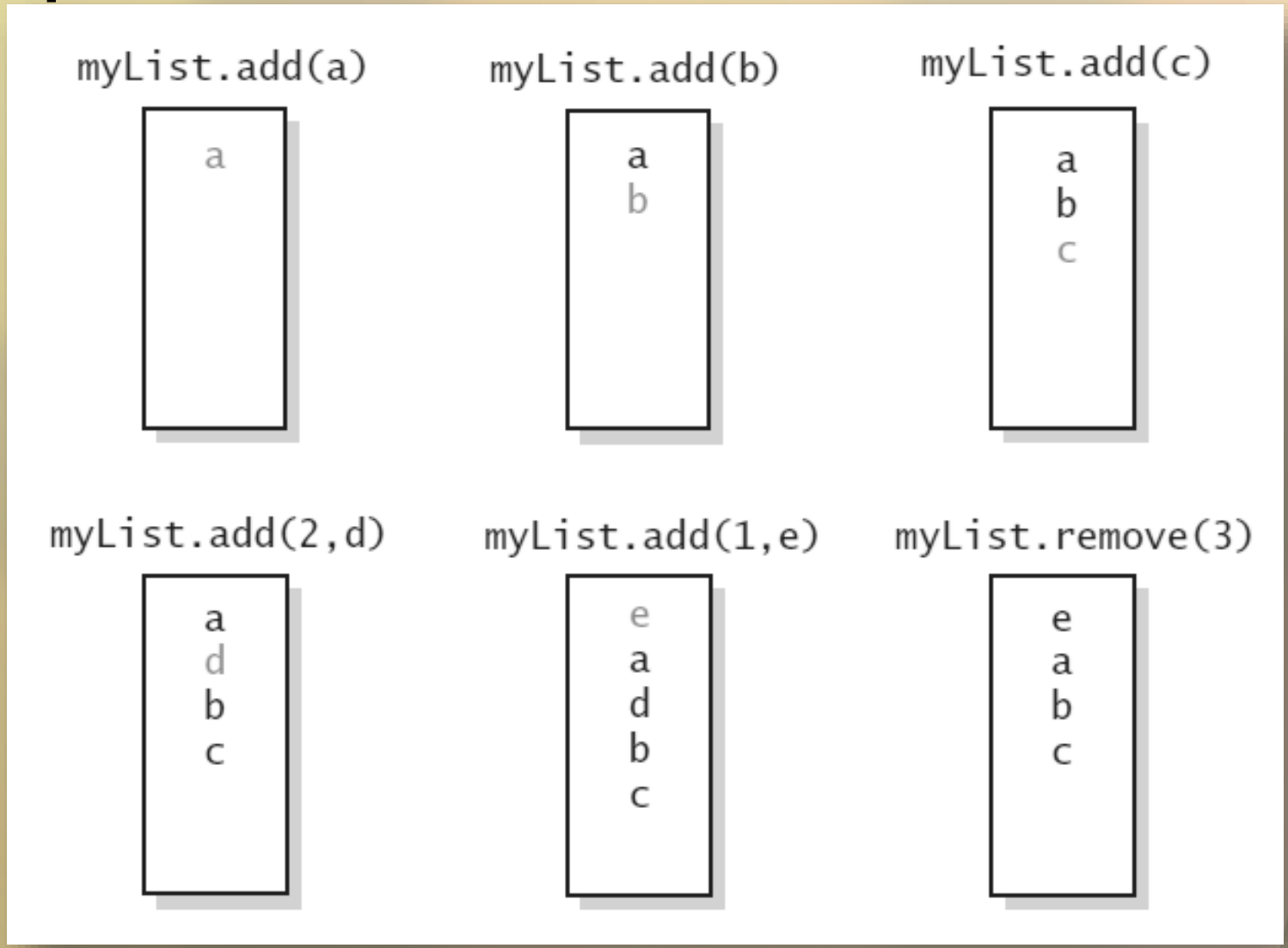

FIGURE 12-2 The effect of ADT list operations on an initially empty list

Specifications for the ADT List

Abstract Data Type: List

Data

- A collection of objects in a specific order and having the same data type
- The number of objects in the collection

Operations

Pseudocode	UML	Description
`add(newEntry)`	`+add(newEntry: T): void`	Task: Adds `newEntry` to the end of the list. Input: `newEntry` is an object. Output: None.
`add(newPosition, newEntry)`	`+add(newPosition: integer, newEntry: T): void`	Task: Adds `newEntry` at position `newPosition` within the list. Position 1 indicates the first entry in the list. Input: `newPosition` is an integer, `newEntry` is an object. Output: Throws an exception if `newPosition` is invalid for this list before the operation.

Description of ADT List

Specifications for the ADT List

		this list before the operation.
`remove(givenPosition)`	`+remove(givenPosition: integer): T`	Task: Removes and returns the entry at position `givenPosition`. Input: `givenPosition` is an integer. Output: Either returns the removed entry or throws an exception if `givenPosition` is invalid for this list. Note that any value of `givenPosition` is invalid if the list is empty before the operation.

Description of ADT List

Specifications for the ADT List

`clear()`	`+clear(): void`	Task: Removes all entries from the list. Input: None. Output: None.
`replace(givenPosition, newEntry)`	`+replace(givenPosition: integer, newEntry: T): T`	Task: Replaces the entry at position `givenPosition` with `newEntry`. Input: `givenPosition` is an integer, `newEntry` is an object. Output: Either returns the replaced entry or throws an exception if `givenPosition` is invalid for this list. Note that any value of `givenPosition` is invalid if the list is empty before the operation.

Description of ADT List

Specifications for the ADT List

`getEntry (givenPosition)`	`+getEntry(givenPosition: integer): T`	Task: Retrieves the entry at position `givenPosition`. Input: `givenPosition` is an integer. Output: Either returns the entry at position givenPosition or throws an exception if `givenPosition` is invalid for this list. Note that any value of `givenPosition` is invalid if the list is empty before the operation.
`toArray()`	`+toArray: T[]`	Task: Retrieves all entries that are in the list in the order in which they occur. Input: None. Output: Returns a new array of the entries currently in the list.

Description of ADT List

Specifications for the ADT List

`contains(anEntry)`	`+contains(anEntry: T): boolean`	Task: Sees whether the list contains `anEntry`. Input: `anEntry` is an object. Output: Returns true if `anEntry` is in the list, or false if not.
`getLength()`	`+getLength(): integer`	Task: Gets the number of entries currently in the list. Input: None. Output: Returns the number of entries currently in the list.
`isEmpty()`	`+isEmpty(): boolean`	Task: Sees whether the list is empty. Input: None. Output: Returns true if the list is empty, or false if not.

Description of ADT List

Specifications for the ADT List

```
1  /** An interface for the ADT list.
2      Entries in a list have positions that begin with 1.
3      @author Frank M. Carrano
4  */
5  public interface ListInterface<T>
6  {
7      /** Adds a new entry to the end of this list.
8          Entries currently in the list are unaffected.
9          The list's size is increased by 1.
10         @param newEntry  The object to be added as a new entry. */
11     public void add(T newEntry);
12
13     /** Adds a new entry at a specified position within this list.
14         Entries originally at and above the specified position
15         are at the next higher position within the list.
16         The list's size is increased by 1.
17         @param newPosition  An integer that specifies the desired
18                             position of the new entry.
19         @param newEntry     The object to be added as a new entry.
20         @throws  IndexOutOfBoundsException if either
21                  newPosition < 1 or newPosition > getLength() + 1. */
22     public void add(int newPosition, T newEntry);
23
```


LISTING 12-1 The interface **`ListInterface`**

Specifications for the ADT List

```
24      /** Removes the entry at a given position from this list.
25          Entries originally at positions higher than the given
26          position are at the next lower position within the list,
27          and the list's size is decreased by 1.
28          @param givenPosition  An integer that indicates the position of
29                                the entry to be removed.
30          @return  A reference to the removed entry.
31          @throws  IndexOutOfBoundsException if either
32                   givenPosition < 1 or givenPosition > getLength(). */
33      public T remove(int givenPosition);
34
35      /** Removes all entries from this list. */
36      public void clear();
37
38      /** Replaces the entry at a given position in this list.
39          @param givenPosition  An integer that indicates the position of
40                                the entry to be replaced.
41          @param newEntry  The object that will replace the entry at the
42                           position givenPosition.
43          @return  The original entry that was replaced.
44          @throws  IndexOutOfBoundsException if either
45                   givenPosition < 1 or givenPosition > getLength(). */
46      public T replace(int givenPosition, T newEntry);
```


LISTING 12-1 The interface **`ListInterface`**

Specifications for the ADT List

```
/** Retrieves the entry at a given position in this list.
    @param givenPosition  An integer that indicates the position of
                          the desired entry.
    @return  A reference to the indicated entry.
    @throws  IndexOutOfBoundsException if either
             givenPosition < 1 or givenPosition > getLength(). */
public T getEntry(int givenPosition);

/** Retrieves all entries that are in this list in the order in which
    they occur in the list.
    @return  A newly allocated array of all the entries in the list.
             If the list is empty, the returned array is empty. */
public T[] toArray();
```


LISTING 12-1 The interface **ListInterface**

Specifications for the ADT List

```
    /** Retrieves all entries that are in this list in the order in which
        they occur in the list.
        @return  A newly allocated array of all the entries in the list.
                 If the list is empty, the returned array is empty. */
    public T[] toArray();

    /** Sees whether this list contains a given entry.
        @param anEntry  The object that is the desired entry.
        @return  True if the list contains anEntry, or false if not. */
    public boolean contains(T anEntry);

    /** Gets the length of this list.
        @return  The integer number of entries currently in the list. */
    public int getLength();

    /** Sees whether this list is empty.
        @return  True if the list is empty, or false if not. */
    public booleanisEmpty();
} // end ListInterface
```


LISTING 12-1 The interface **ListInterface**

Using the ADT List

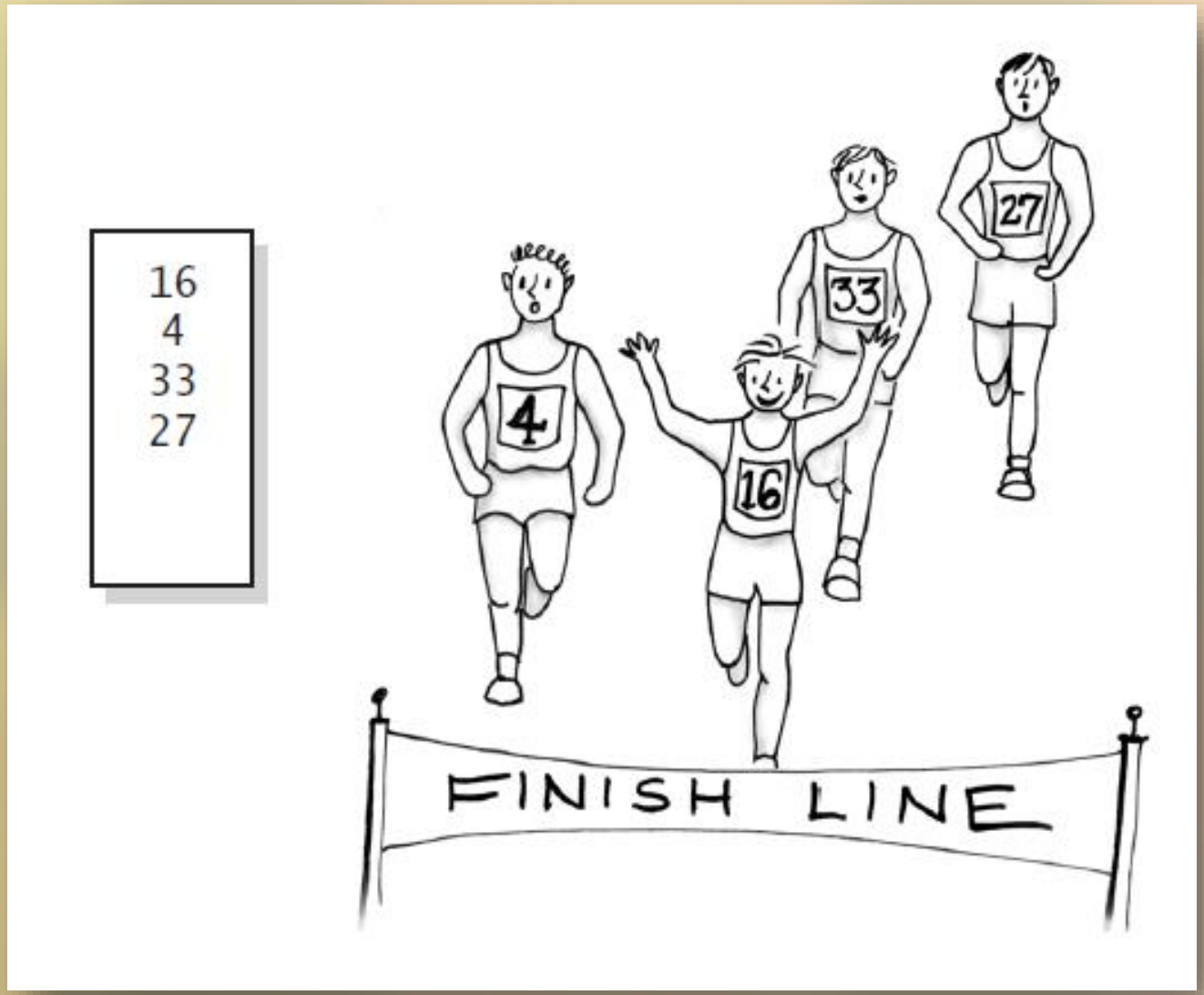

FIGURE 12-3 A list of numbers that identify runners in the order in which they finished a race

Using the ADT List

```
public class ListClient
{
   public static void main(String[] args)
   {
      testList();
   } // end main

   public static void testList()
   {
      ListInterface<String> runnerList = new AList<>();
//  runnerList has only methods in ListInterface

      runnerList.add("16"); // Winner
      runnerList.add(" 4"); // Second place
      runnerList.add("33"); // Third place
      runnerList.add("27"); // Fourth place
      displayList(runnerList);
   } // end testList

   public static void displayList(ListInterface<String> list)
```


LISTING 12-2 A client of a class that implements `ListInterface`

Using the ADT List

```
   public static void displayList(ListInterface<String> list)
   {
      int numberOfEntries = list.getLength();
      System.out.println("The list contains " + numberOfEntries +
                         " entries, as follows:");

      for (int position = 1; position <= numberOfEntries; position++)
         System.out.println(list.getEntry(position) +
                            " is entry " + position);

      System.out.println();
   } // end displayList
} // end ListClient
```

Output

```
The list contains 4 entries, as follows:
16 is entry 1
 4 is entry 2
33 is entry 3
27 is entry 4
```


LISTING 12-2 A client of a class that implements `ListInterface`

Using the ADT List

```
// Make an alphabetical list of names as students enter a room
ListInterface<String> alphaList = new AList<>();
alphaList.add(1, "Amy");    // Amy
alphaList.add(2, "Ellen");  // Amy Ellen
alphaList.add(2, "Bob");    // Amy Bob Ellen
alphaList.add(3, "Drew");   // Amy Bob Drew Ellen
alphaList.add(1, "Aaron");  // Aaron Amy Bob Drew Ellen
alphaList.add(4, "Carol");  // Aaron Amy Bob Carol Drew Ellen
```

Example

Using the ADT List

```
// Make a list of names as you think of them
ListInterface<Name> nameList = new AList<>();
Name amy = new Name("Amy", "Smith");
nameList.add(amy);
nameList.add(new Name("Tina", "Drexel"));
nameList.add(new Name("Robert", "Jones"));
```

A list of **`Name`** objects, rather than **`String`**

Java Class Library: The Interface `List`

```
public boolean add(T newEntry)
public void add(int index, T newEntry)
public T remove(int index)
public void clear()
public T set(int index, T anEntry) // Like replace
public T get(int index)            // Like getEntry
public boolean contains(Object anEntry)
public int size()                  // Like getLength
public boolean isEmpty()
```

Method headers from the interface `List`

Java Class Library: The Class `ArrayList`

- Available constructors
  - `public ArrayList()`
  - `public ArrayList(int initialCapacity)`
- Similar to java.util.vector
  - Can use either `ArrayList` or `Vector` as an implementation of the interface `List`.

End

Chapter 12