Advanced Programming

Introduction to Lists
Textbook
Lists
Topics

• Specifications for the ADT List
  ▪ Redefining the Specifications
• Using the ADT List
• Using a List Is Like Using a Vending Machine
• Java Class Library: The Interface List
Terms

• Abstract Data Type (ADT)
  ▪ Represents collection of data having the same type and operations on that data
    • Only describes data
    • Specifies operations
  ▪ NOT how to store data & implement operations

• Data Structure
  ▪ An implementation of an ADT within a programming language
Specifications for the ADT List

• A list provides a way to organize data

I have so much to do this weekend — I should make a list.

To Do
1. Read Chapter 4
2. Call home
3. Buy card for Sue
Specifications for the ADT List

- Operations on lists
  - **Add** new entry – at end, or anywhere
  - **Remove** an item
  - **Remove all** items
  - **Replace** an entry
  - **Look at** any entry
  - **Look for** an entry of a specific value
  - **Count** how many entries
  - Check if list is **empty**, **full**
  - **Display** all the entries
Specifications for the ADT List

• To specify an ADT list
  ▪ Describe its data
  ▪ Specify the operations

• ADT list must be considered in general
  ▪ Not necessarily a list of strings

• View specifications
Example

The effect of ADT list operations on an initially empty list
Potential Problem Operations

- add, remove, replace, `getEntry` work OK when valid position given
- `remove`, `replace` and `getEntry` not meaningful on empty lists
- A list could become full, what happens to `add`?
Possible Solutions

- Assume the invalid situations will not occur
  - Preconditions to which clients must adhere
- Ignore the invalid situations
  - Methods do nothing in such situations
- Make reasonable assumptions, act in predictable way
  - Remove item from 3\textsuperscript{rd} position (remove 6\textsuperscript{th})
- Return a value that signals a problem
  - An unusual value (null)
- Return boolean value indicating success or failure of the operation
- Throw an exception
Redefining Specifications

• A first draft of an ADT specifications may ignore potential problems
  ▪ Simplifies the first draft

• Concentrate on details after major portions of specifications written
  ▪ Makes the specifications complete

• After writing specifications, implementations
  ▪ Write Java statements to use the ADT
  ▪ Checks understanding, suitability of the specifications
Interface for ADT

ListInterface

• View source code
• Note
  ▪ Interface has no data fields, constructors
  ▪ Methods must be public
  ▪ Strategy for add, remove, replace, getEntry is to have them return a value
  ▪ Use of return of a reference in remove and getEntry
Using the ADT List

A list of numbers that identify runners in the order in which they finish a race
Using the ADT List

• Consider the scoring of a running race
• We wish to note the order in which the runners finish
• We add each runner's (unique) number to the end of the list
• When done we display the whole list
• View sample program
A List is Like a Vending Machine

A vending machine.
A List is Like a Vending Machine

Observations about vending machines
• Can perform only tasks shown on interface
• Must understand the tasks
• Cannot see inside the machine
• Can use the machine even though don’t know what happens inside
• If inside of machine replaced with new improved version
  ▪ Interface remains unchanged
  ▪ Customer uses machine in same way as before
A List is Like a Vending Machine

Observations about clients and List ADT

- Client can perform only operations from the ADT List
- Client must adhere to specifications
- Client cannot access data without an ADT operation
- Client can use the list – even though unable to access entries directly
- If implementation is changed, client still uses list in same way as before
Java Class Library: The Interface List

- The standard package contains a list interface – called List
- Methods provided:

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

- Data Structures and Abstractions with Java, by Carrano, Second Edition.