CMPT 225
Data Collection as ADT – List –
Array-based implementation of List ADT
Part 3
Last Lecture

- The 4 Steps of the Software Development Process
  - Step 2 – OO Design
- Introduced List data collection
Learning Outcomes

- At the end of this lecture, a student will be able to … :
  - Describe what we do in each of the 4 steps of software development process
  - Apply the steps of this process while solving a problem using a data collection designed as an abstract data type (ADT)
  - Describe our first data collection -> List
Today’s Menu

- i-clicker Session 1
- The 4 Steps of the Software Development Process
  - Finish Step 2
  - Step 3 – Implementation
  - Step 4 – Testing
Step 2 – How to design an ADT
2. Data Structure Design

- Design of the data structure used to implement our List data collection ADT class
  - Array-based implementation
Step 3 - How to implement a data collection ADT class in C++

- C++ Header file
  1. Visible part -> Specification (Public Interface)
     - Public methods headers
  2. Hidden part -> Implementation
     - Private data members such as the data structure
     - Private methods
  3. Documentation
Step 3 - How to implement a data collection ADT class in C++

- C++ Implementation file
  1. The remaining of the hidden part
     - Implementation of all methods (public and private)
  2. Documentation
Step 3 - Documentation

- Header comment block
  - Class/file name
  - Class description
  - Class invariant (if any)
  - Author
  - Creation/modification date
- For each public and private methods
  - Description
  - Precondition
  - Postcondition
Step 3 – Solution Code

- Posted on our course web site
Step 4 - Testing

- Test case
  1. Test data
  2. Expected results
  3. Actual results

- Example:
Testing classes

- Each class must be tested using a test driver program
- This test driver program must call each of the class’ public methods at least once
Testing our List ADT class

- Which test plans would we consider?

- Cover all possible cases at least once
  - For example:
    - inserting into/removing from empty/full List
    - inserting into/removing from List that has some elements
    - etc…
Advantages of data collection ADT’s

1. Application easy to understand, design, implement, maintain, debug, test because each class is assigned only one responsibility
   - FriendsBook -> interface with the user
   - Data collection List ADT -> manage data (friends objects)
   - Profile -> represents a member’s profile
Advantages of data collection ADT’s

2. Ease s/w development team work
   ◦ Software developer A writes the data collection class as an ADT
     1. visible (public) section
        • Public Interface
     2. hidden (private) section:
        • Underlying data structure
        • Implementation of the ADT’s operations
   ◦ Software developer B writes the client code, i.e., classes using this ADT class, simply by using the ADT’s Public Interface (Specifications)
Advantages of data collection ADT’s

3. Ease modification
   ◦ Software developer A can change the ADT’s hidden (private) section
     • Underlying data structure
     • Implementation of the ADT’s operations
   without affecting the ADT’s Public Interface and therefore without affecting Software developer B’s client code
Advantages of data collection ADT’s

4. Ease reusability
   ◦ Can be application-independent, hence reusable in another application
     • Data collection List ADT can be reused in another application whenever a List is needed without having to modify, recompile and retest the code

Note:
• Right now, we cannot fully achieve this advantage because we need to modify the code, i.e., changing the data type of the data we store in our data collection, recompile and retest our List ADT class for every application
• However, we shall soon see “template”, which is a mechanism that will allow us to completely achieve the advantage described above
Summary

- i-clicker Session 1
- The 4 Steps of the Software Development Process
  - Finish Step 2
  - Step 3 – Implementation
  - Step 4 – Testing
Next Lecture -> Wed. May 24

- Link-based implementation of List ADT class
- Side trip -> pointers and linked lists