School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

COR-IS1702:
COMPUTATIONAL THINKING
WEEK 6: LINEAR DATA STRUCTURES

"Bad programmers worry about the code. Good programmers worry about data structures and their relationships." ~ Linus Torvalds

2020/21 Term 1

(06) Linear Data Structures Part 1: Abstract Data Types

Video (5mins): https://youtu.be/mLQbkVGdNKU

Road Map

Algorithm Design and Analysis

(Weeks 1 - 5)

Fundamental Data Structures

This week →

- Week 6: Linear data structures (stack, queue)
- Week 7: Hierarchical data structure (binary tree)
- Week 9: Networked data structure (graph)

Computational Intractability and Heuristic Reasoning

(Weeks 10 - 13)

Learning Outcomes

- Understand the operations of linear data structures, including stacks, queues, and priority queues

- Able to apply the appropriate linear data structure in different application contexts

Abstract Data Types (ADT)

Modeling data structures by what they do, not how they do it

- Abstraction
- Data Structures:
  - Linear
  - Hierarchical
  - Networked

Collections of Data Objects

- In everyday life we often encounter collections
  - course catalog – collection of course descriptions
  - car lot – collection of cars
- Mathematicians also work with collections
  - matrix
  - sequence (e.g. 1, 1, 2, 3, 5, 8, ...)
- In computer science we make a collection by defining a “data structure” that includes references to other objects

Abstract Data Type

Abstract data type is a data type whose representation is hidden from the client.

Robert Sedgewick, “Algorithms,” 4th edition, Pearson.

- Defined by the operations that it support
- Not by the specific implementations

Advantage: Encapsulation

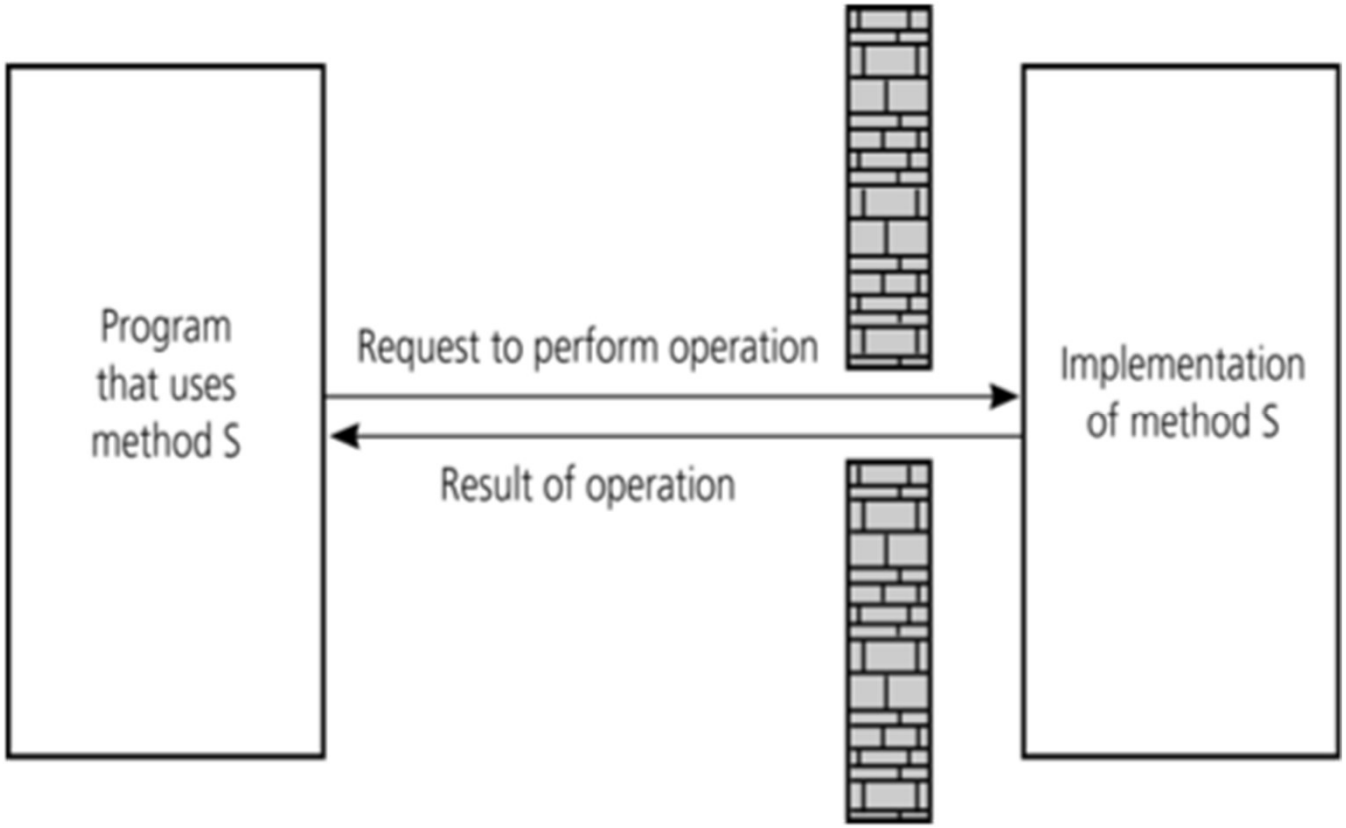


Figure 4-2 in Prichard and Carrano, "Data Abstraction & Problem Solving with Java", 3rd edition, Pearson.

The implementation may be complex, but clients do not need to know.

Advantage: Localization

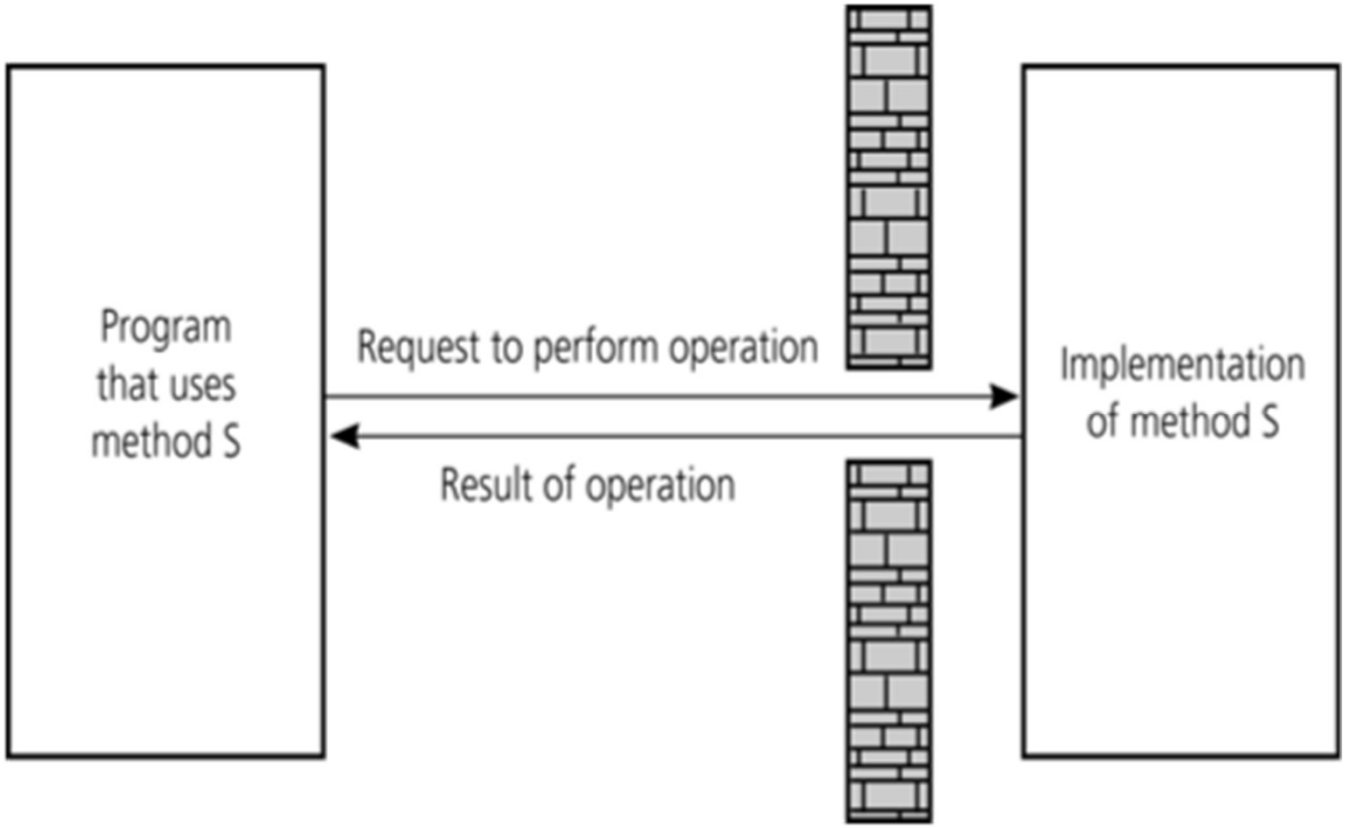


Figure 4-2 in Prichard and Carrano, "Data Abstraction & Problem Solving with Java", 3rd edition, Pearson.

Any change to the implementation should not affect existing clients.

Advantage: Flexibility

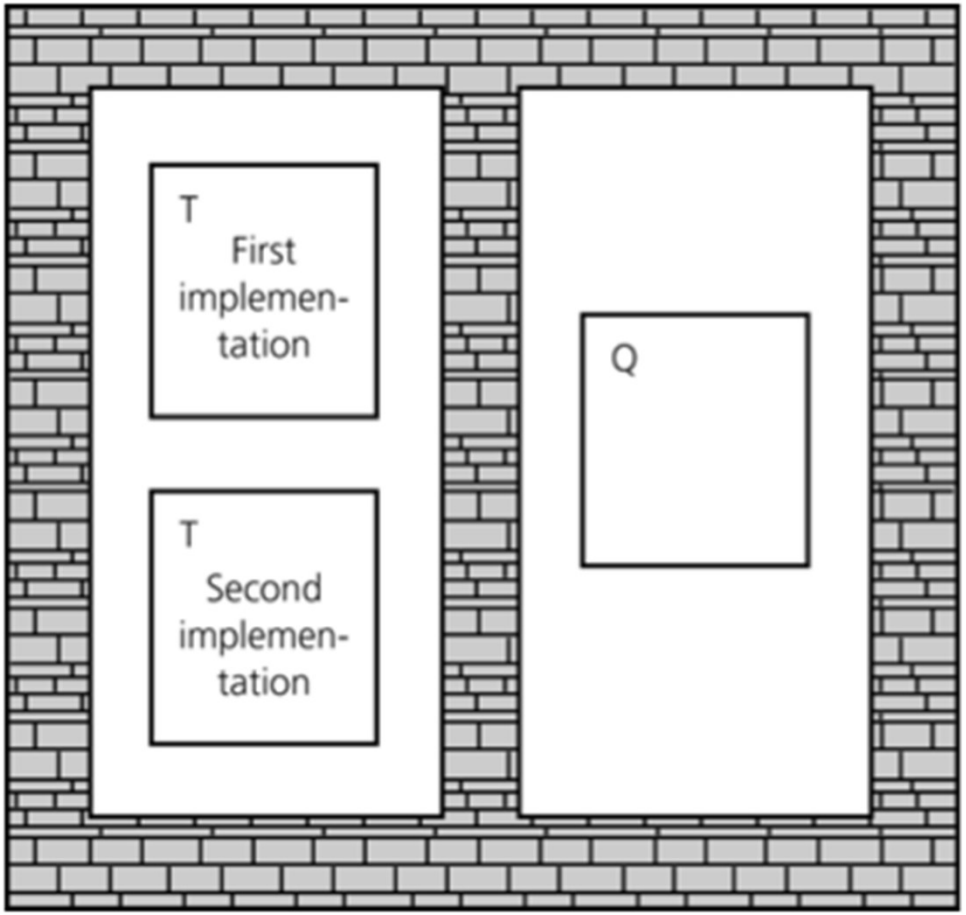
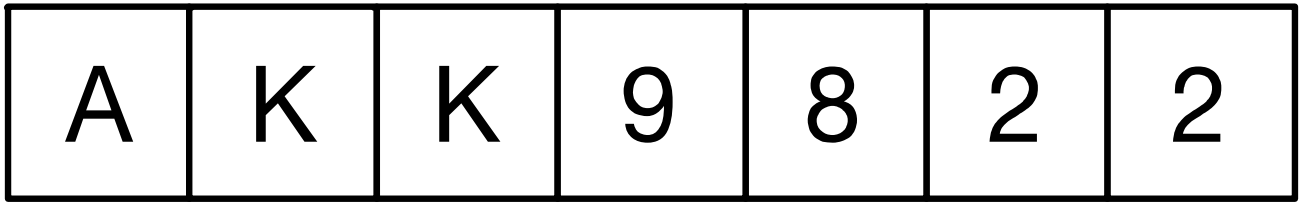


Figure 4-2 in Prichard and Carrano, "Data Abstraction & Problem Solving with Java", 3rd edition, Pearson.

We can swap different implementations of the same ADT without affecting the client.

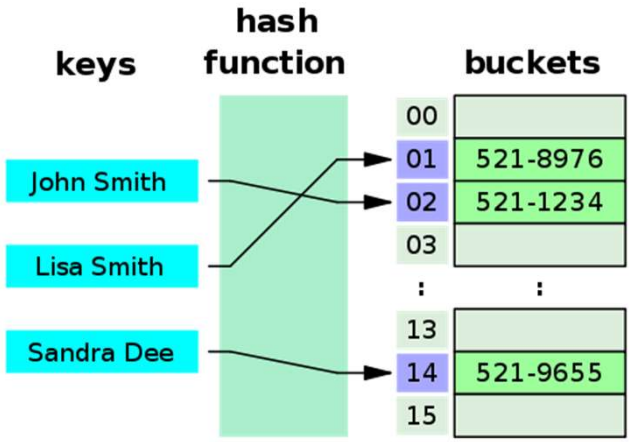
Example: List



- Items in a list is *ordered*.
  - Each item except the last has a successor.
- Operations:
  - creating a new list
  - inserting a new item into the list (at a specific position)
  - retrieving an item at a specific position in the list
  - removing an item at a specific position in the list
  - counting the number of items in the list

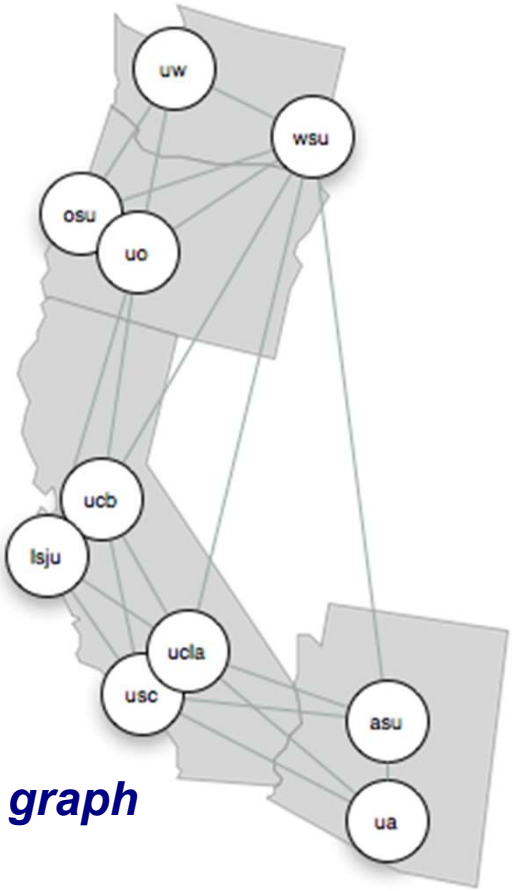
List in Python

- We have been using lists in Python so far
- Operations:
  - creating a new list `a`
    - `a = []`
  - inserting a new item into the list `a`
    - `a.append("item")`
  - inserting a new item into the list `a` at a specific position `i`
    - `a.insert(i, "item2")`
  - retrieving an item at a specific position `i` in the list `a`
    - `a[i]`
  - removing an item at a specific position `i` in the list `a`
    - `del a[i]`
  - counting the number of items in the list
    - `len(a)`

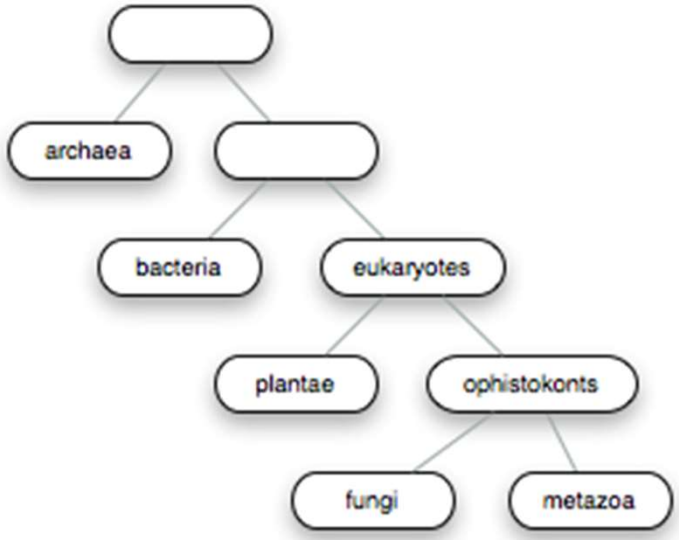


hash table

queue



graph



tree

stack

What We Will Cover in the Second Module

Fundamental Data Structures

- Week 6: Linear data structures (stack, queue, priority queue)
- Week 7: Hierarchical data structure (binary tree)
- Week 9: Networked data structure (graph)
- Week 10: Graph Algorithms

References

- Indexed data structures: *hash tables*
  - Not covered in the course, but you are encouraged to read

- Handout prepared by instruction team
  - Linear data structures: *stacks, queues, priority queues*
  - Hierarchical data structure: *binary trees*
  - Networked data structure: *graphs*