NTE Faculty Interview

Reese Hatfield

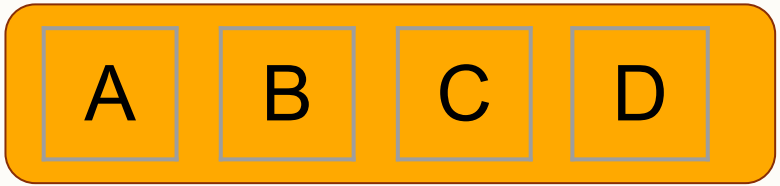
About Me

- Reese Hatfield
- Graduate Teaching Assistant
  - ~3 years
- Adjunct Instructor for this Summer and Fall
- Guest lecturer for introductory CS courses

ADTs, Stacks, and Queues

Abstract Data Types (ADTs)

- Everyone has used a List before
- What *actually* makes something a List
- How we can describe the idea of a “List” in more general terms



Abstract Data Types (ADTs)

- Define a series of *ways* to interact with the data
- Tell you *nothing* about how the data is stored

List ADT
+ add(Element)
+ contains(Element)
+ clear()
+ get(index)
+ remove(Element)

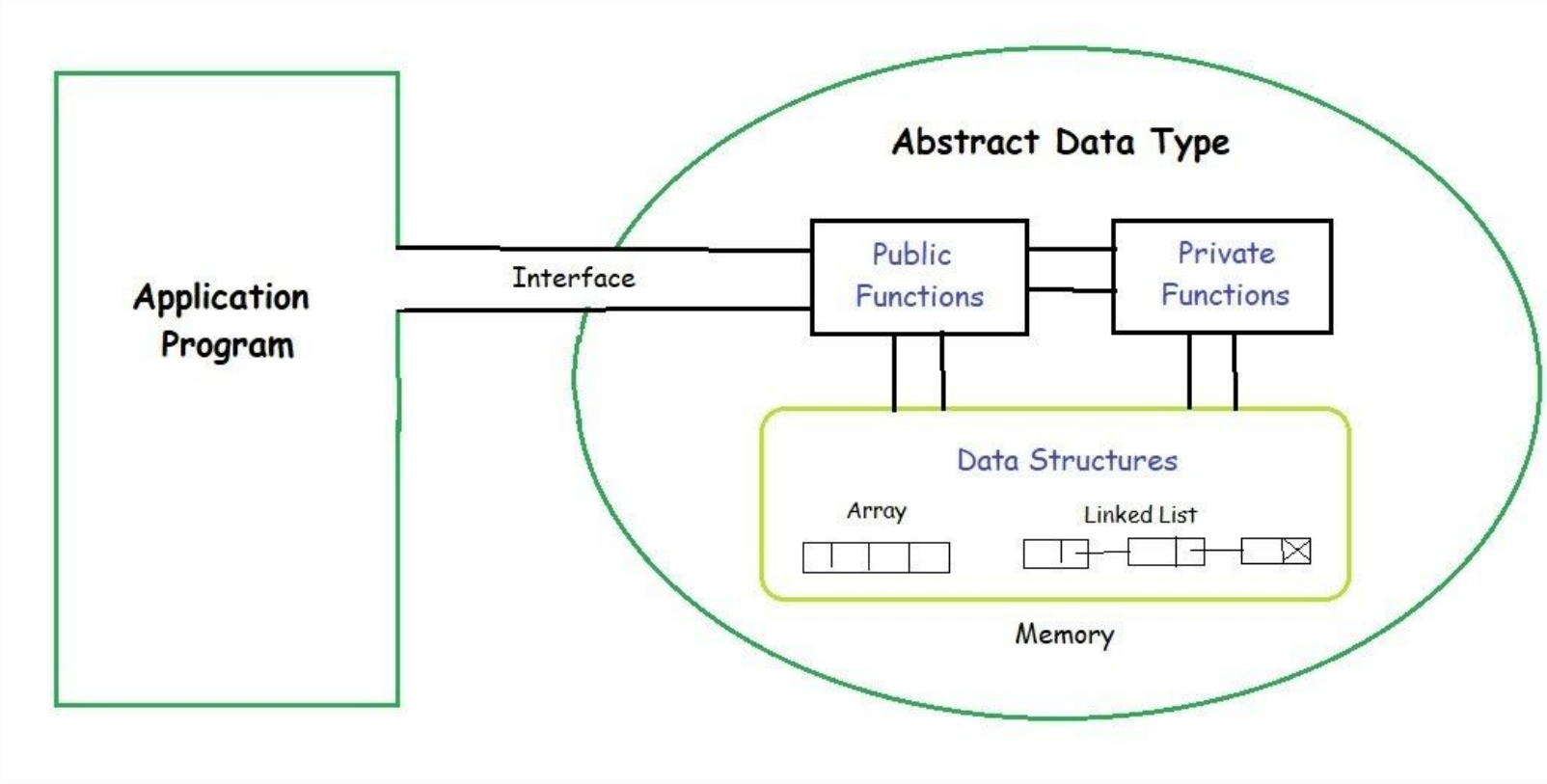
Abstract Data Types (ADTs)

ADTs *do*

- Define operations and methods
- What actions can be performed
- add(), get(), remove(), etc

ADTs do *not*

- Define implementation
- Structure or *type* of underlying data
- Specify performance



https://www.geeksforgeeks.org/abstract-data-types/

Applied ADTs

- ADTs enable you to focus on solving high-level problems
  - Power in abstraction

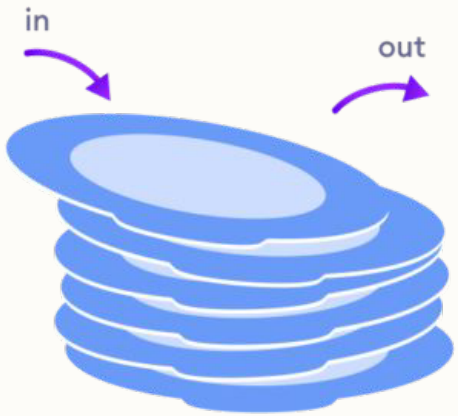
Applied ADTs

- ADTs enable you to focus on solving high-level problems
  - Power in abstraction

- What are some other ADTs we have probably heard of?

Stacks

- Last In → First out
- Only let you modify the thing on top
- Restricts any other operations
- Like a stack of plates



A pile of plates

https://studyalgorithms.com/theory/stack-data-structure/

Applied Stacks

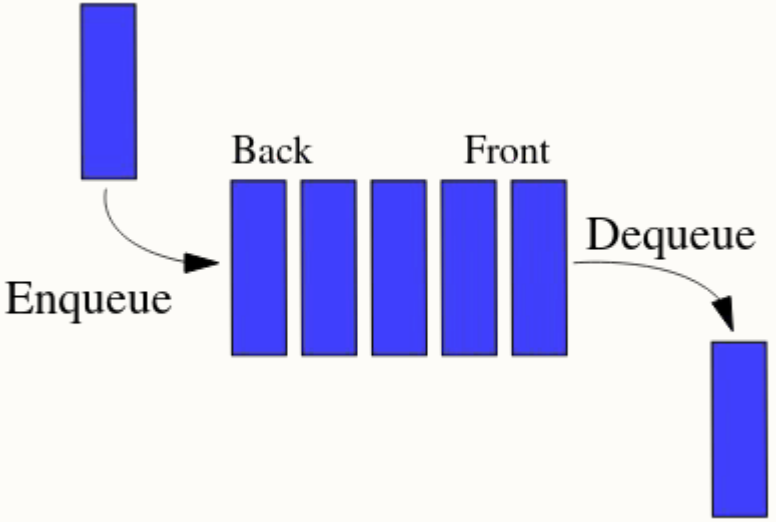
- Permitted operations
  - push(), pop(), peek()

- How should we implement a stack?
  - Linked data structure
  - Contiguous array structure

Stack ADT
+ push(Element)
+ pop(): Element
+ peek(): Element
+ contains(): bool
+ clear()

Queues

- First In → First out
- Only add to the front
- Remove from the back
- Restricts internal data manipulation
- Like a drive-thru line



https://javascripttoday.com/blog/queue-data-structure-in-javascript/

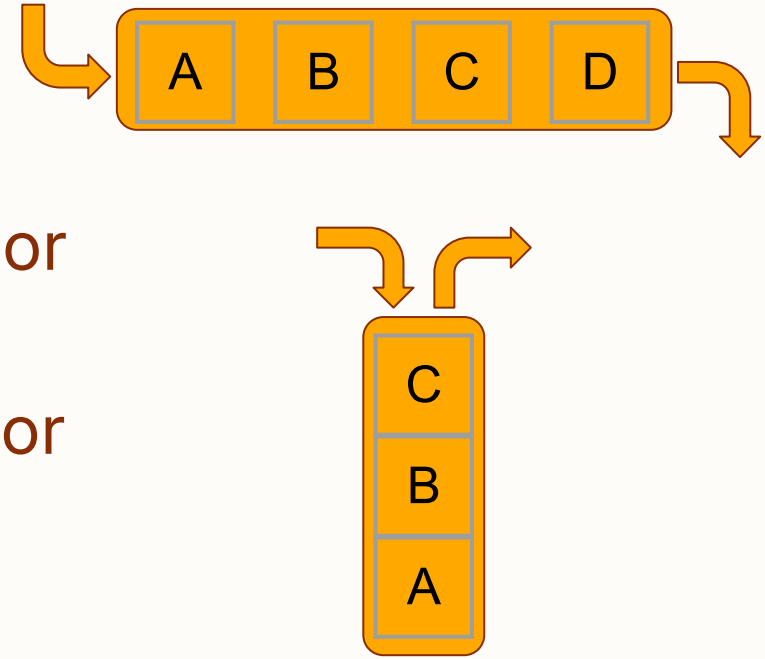
Applied Queues

- Permitted operations enqueue(), dequeue(), front(), etc.
- How should we implement a Queue?
  - Linked data structure
  - Contiguous array structure

Queue ADT
+ enqueue(Element)
+ dequeue(): Element
+ front(): Element
+ contains(): bool
+ clear()

Choosing the right tool

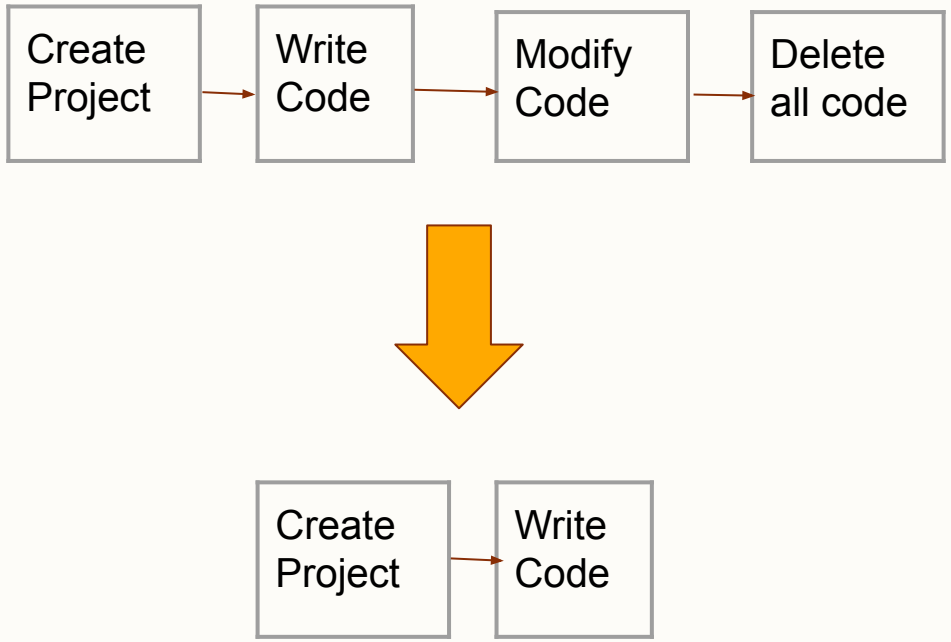
- Different problems call for different ADTs
- Queues excel at modeling scenarios with FIFO behavior
- Stacks excel at modeling scenarios with LIFO behavior
- Let’s do some examples



Problem #1

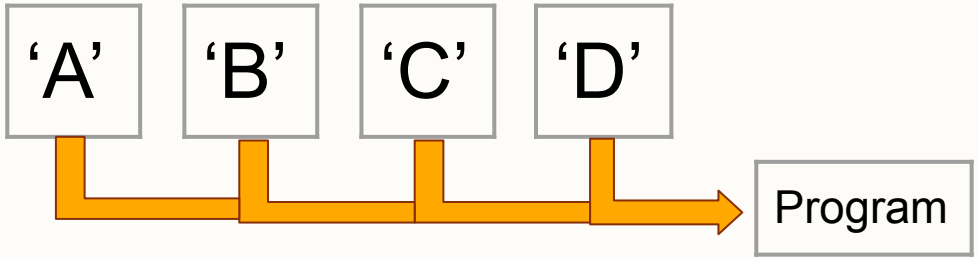
- Input: A sequence of operations.

- Output: The same operations, with the most recent two undone.



Problem #2

- Input: A sequence of customers

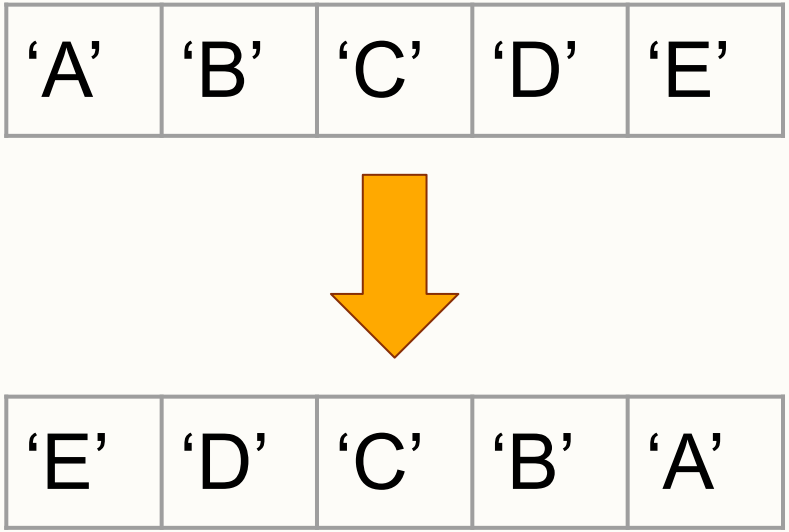


- Output: A log detailing the order of customer arrivals

Arrival Time	1	2	3	4
Customer	‘A’	‘B’	‘C’	‘D’

Problem #3

- Input: a series of elements
- Output: the series of elements in reversed order



Overview

Feature	Stack	Queue
Access Order	LIFO	FIFO
Element availability	Only the top	Only front and back
Common methods	push(), pop(), peek()	enqueue(), dequeue(), front()
Analogy	Stack of Plates	Drive-thru line

Questions?

Questions?

How would we implement a queue that gives some elements special priority?

Teaching Philosophy

Communication

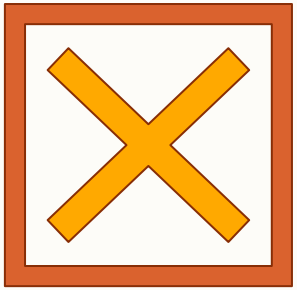
- Lived student experiences
- Growing together
- Bi-directional expectations

Diversity

- Students come from all over the world
- Diverse background
- Diversity of teaching styles

Engagement

- Low risk environment
- Make frequent mistakes
- Consistent, honest feedback



Open floor Q/A