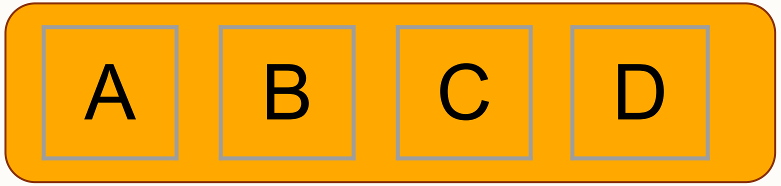
ADTs, Stacks, Q's, and Maps

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)

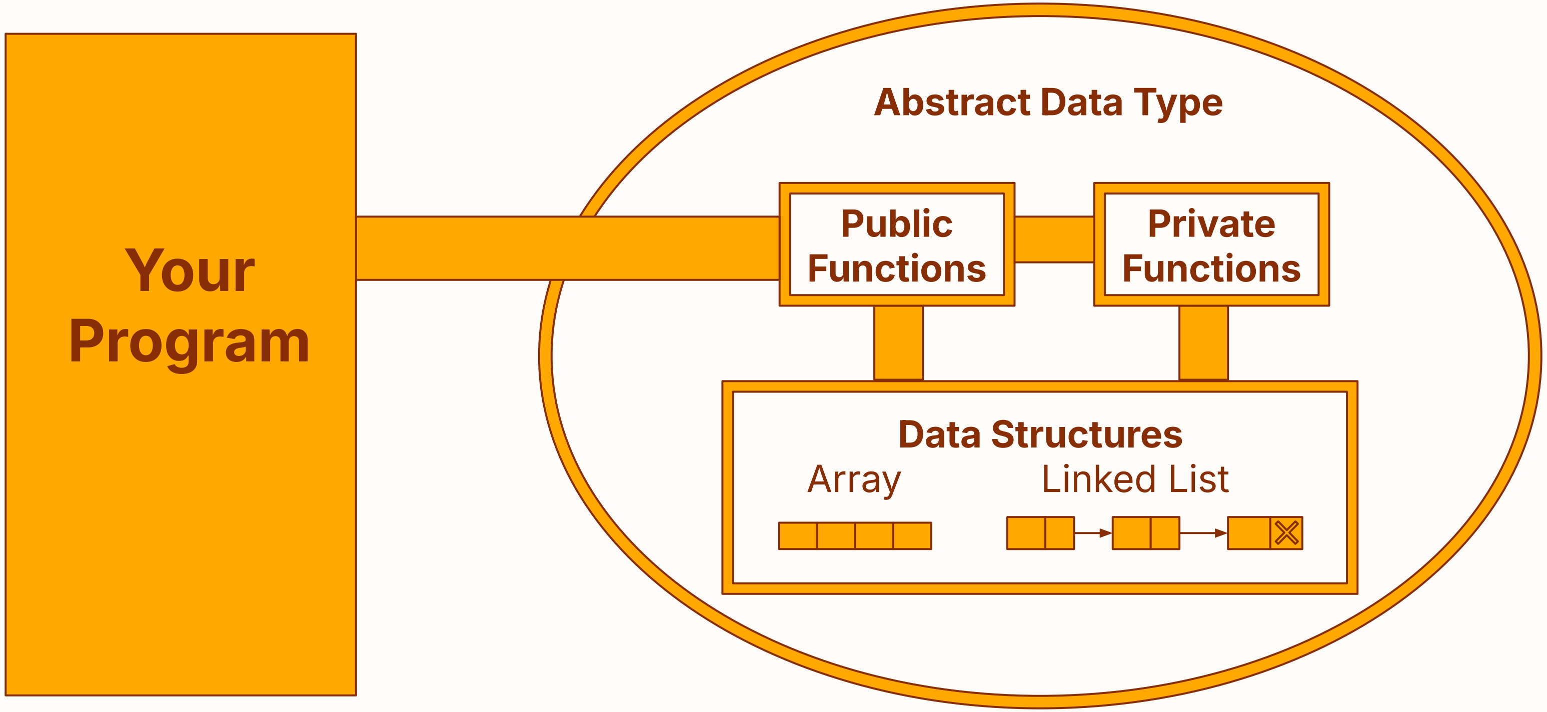
Abstract Data Types (ADTs)

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


Your Program
Abstract Data Type
Public Functions
Private Functions
Data Structures
Array
Linked List

Applied ADTs

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

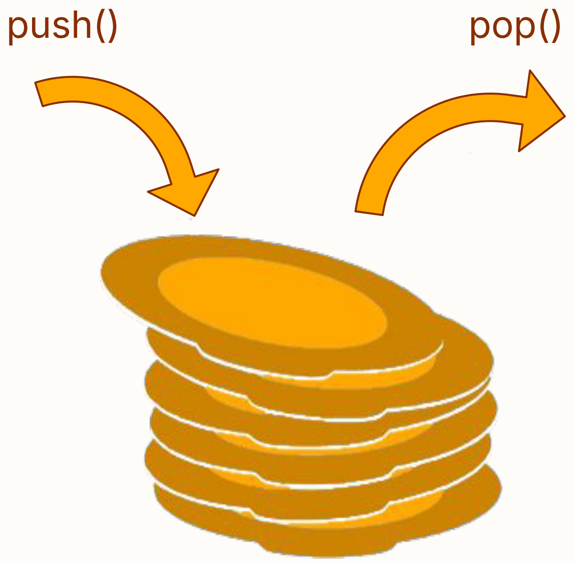
Applied ADTs

- 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

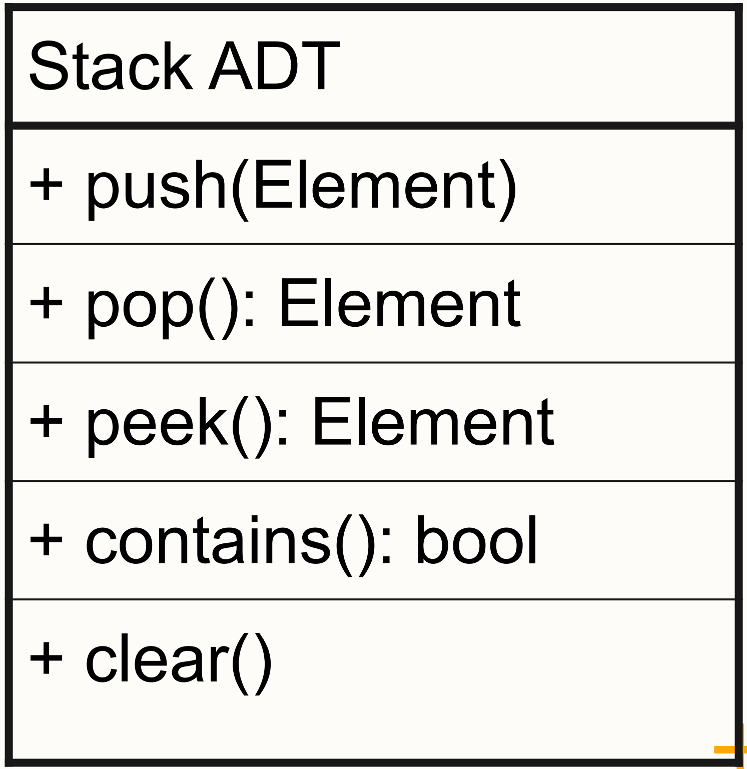


A stack of Plates

Applied Stacks

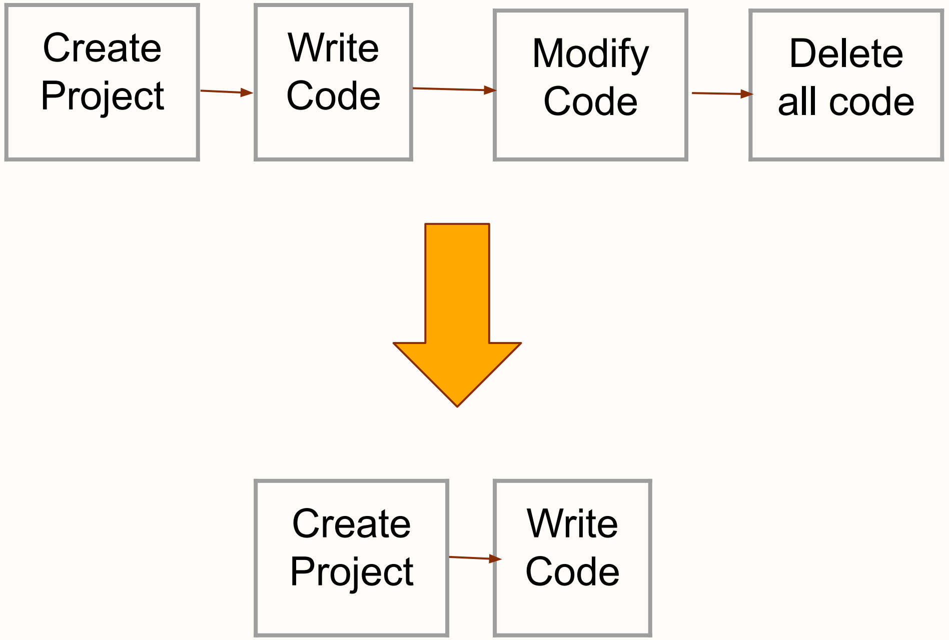
- Permitted operations
  - push(), pop(), peek()

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



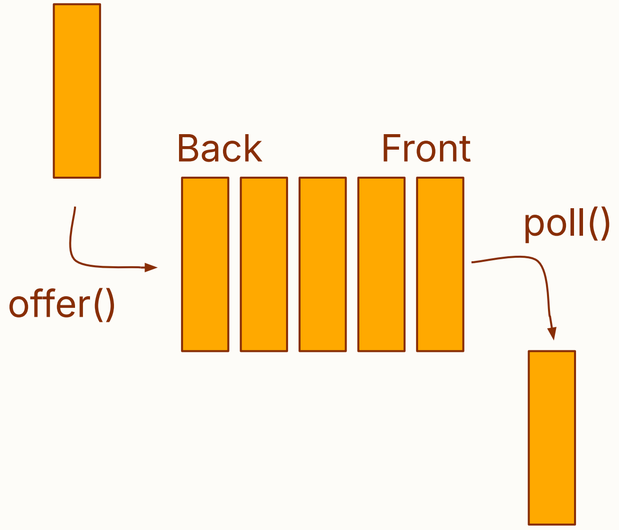
Problem #1

- Input: A sequence of operations.
- Output: The same operations, with the most recent two undone.



Queues

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



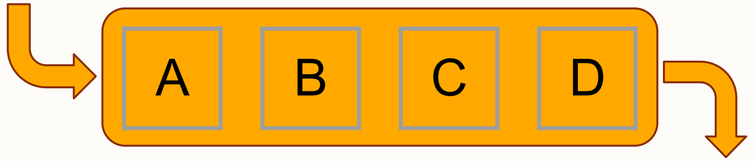
Applied Queues

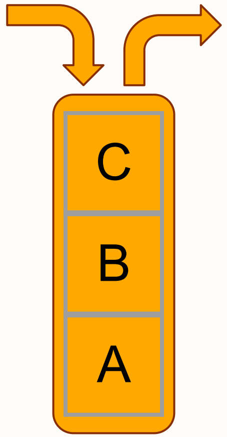
- Permitted operations offer(), poll(), peek(), etc.
- How should we implement a Queue?
  - Linked data structure
  - Contiguous array structure

Queue ADT
+ offer(Element)
+ poll(): Element
+ peek(): Element
+ contains(): bool
+ clear()

Choosing the right tool

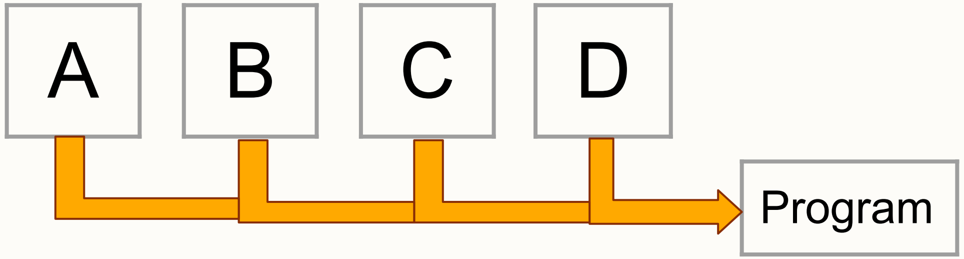
- Different problems call for different ADTs
- Queues excel at modeling scenarios with FIFO behavior
- Stacks excel at modeling scenarios with LIFO behavior
- Let's do some examples





Problem #2

- Input: A sequence of customers

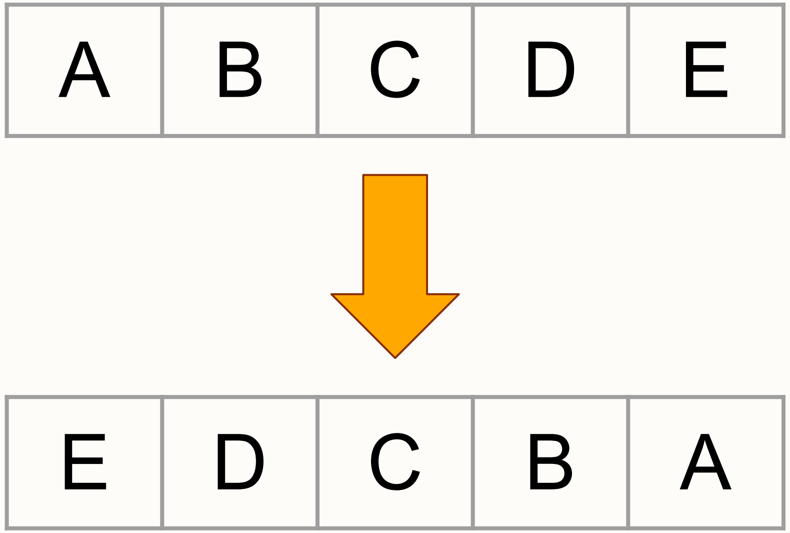


- Output: A log detailing the order of customer arrivals

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

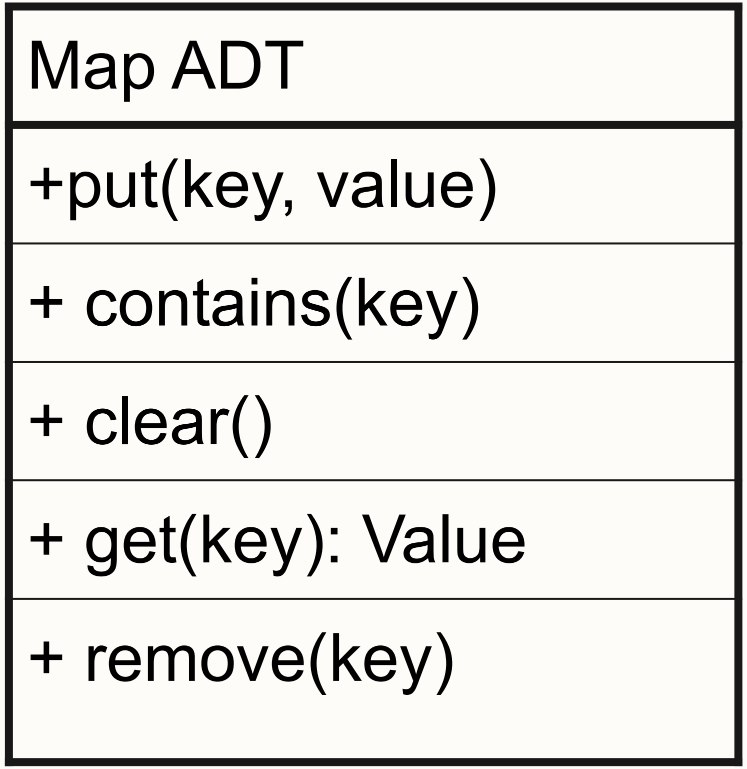


Overview

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

Map ADT

- Key Value Pairs
- Associate one value to another
- Really fast get() operations



Map ADT

+put(key, value)

+ contains(key)

+ clear()

+ get(key): Value

+ remove(key)

Map ADT

- Maps show up *all* the time
- People → Favorite Color
- Countries → Capitals
- Let’s practice using them a bit more
  - Letter occurrence counter

countLetters(targetLetter, String)

Questions?

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