LEC 04

CSE 122

Stacks & Queues

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

What are your favorite/least favorite classes at UW so far?

Instructors Tristan Huber & Hunter Schafer

TAs

Ambika
Andrew
Audrey
Autumn
Ayush
Ben
Colton
Di
Eesha
Elizabeth
Evelyn
Jacob
Jaylyn
Jin
Joe
Kevin
Leon
Megana
Melissa
Mia
Poojitha
Rishi
Rucha
Shivani
Shreya
Steven
Suhani
Yijia
Ziao

Lecture Outline

- Announcements
- Review: ADTs, Stacks & Queues
- Queue Manipulation
- Stack Manipulation
  - Problem Solving

Announcements

- Quiz 0
  - Grades available in a couple days
  - Retake info on Friday
- Creative Project (C0) due tomorrow
- Programming Assignment 1 (P1) will be released Friday
  - It will be due next Thursday (4/20)

Lecture Outline

- Announcements
- **Review: Stacks & Queues**
- Queue Manipulation
- Stack Manipulation
  - Problem Solving

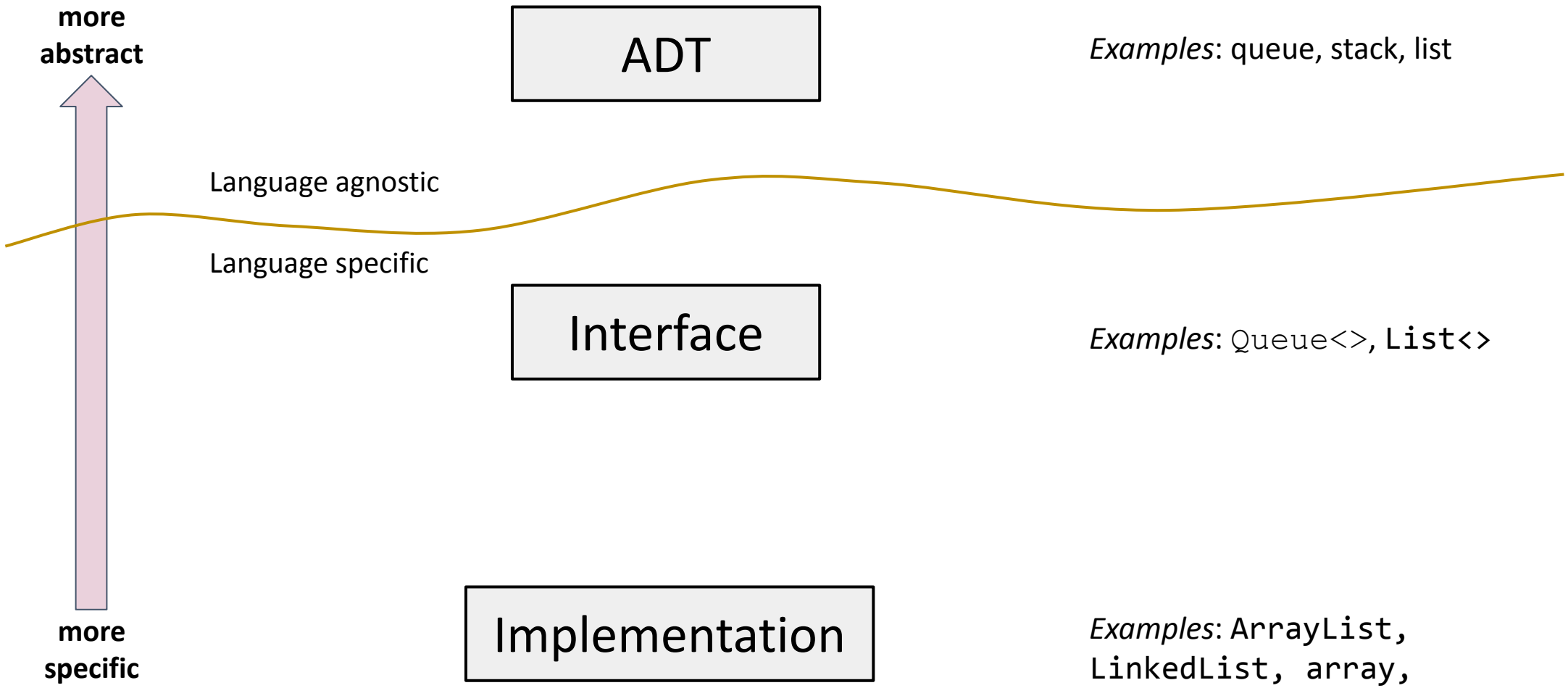
(PCM) Abstract Data Types

- **Abstract Data Type (ADT)**: A specification of a collection of data and the operations that can be performed on it.
  - Describes *what* a collection does, not *how* it does it

- We don't know exactly how a stack or queue is implemented, and we don't need to.
  - Only need to understand high-level idea of what a collection does and its operations

  - **Stack:** retrieves elements in reverse order as added.
    Operations: push, pop, peek, …
  - **Queue:** retrieves elements in same order as added.
    Operations: add, remove, peek, …

(PCM) Abstract Data Types



🎉 NEW DATA STRUCTURE DAY!! 🎉

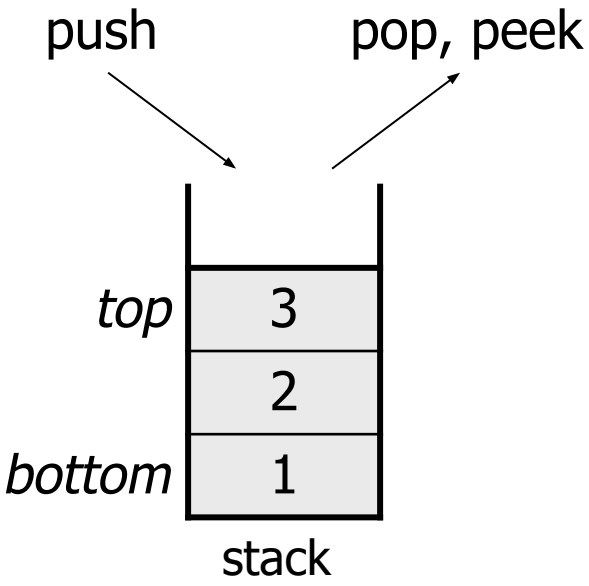
Stack - What is it good for?

What is it?

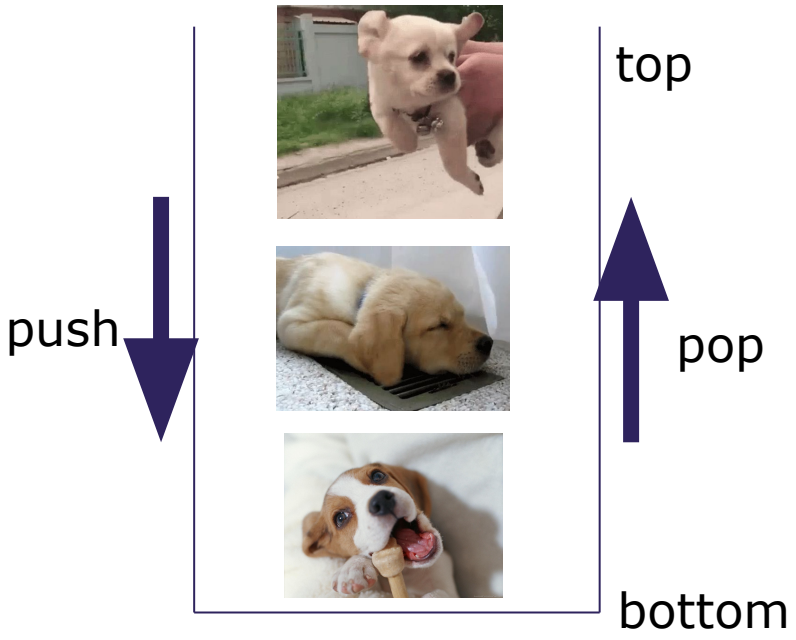
- A **Last**-in-First-out (LIFO) data structure
  - Elements are removed in the **reverse order** to how they were added
- All elements must be of same type*
- Dynamically sized

What is Stack particularly good at?

- `push` - add element to top
- `pop` - remove element from top
- Supported operations are few but *very efficient*



(PCM) Stacks



Stacks in Computer Science

- Programming languages and compilers:
  - method calls are placed onto a stack (*call=push, return=pop*)
  - compilers use stacks to evaluate expressions
- Matching up related pairs of things:
  - find out whether a string is a palindrome
  - examine a file to see if its braces { } match
  - convert "infix" expressions to pre/postfix
- Sophisticated algorithms:
  - searching through a maze with "backtracking”
  - many programs use an "undo stack" of previous operations

(PCM) Programming with Stacks

`Stack<E>()`	constructs a new stack with elements of type **E**
`push(value)`	places given value on top of stack
`pop()`	removes top value from stack and returns it; throws `EmptyStackException` if stack is empty
`peek()`	returns top value from stack without removing it; throws `EmptyStackException` if stack is empty
`size()`	returns number of elements in stack
`isEmpty()`	returns `true` if stack has no elements

```
Stack<String> s = new Stack<String>();
s.push("a");
s.push("b");
s.push("c");                    // bottom ["a", "b", "c"] top
System.out.println(s.pop()); // "c"
```

- `Stack` has other methods that we will ask you not to use

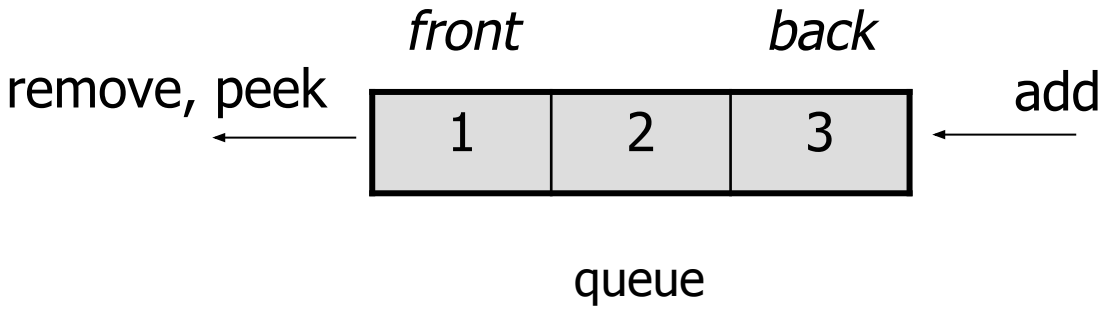
Queue - What is it good for?

What is it?

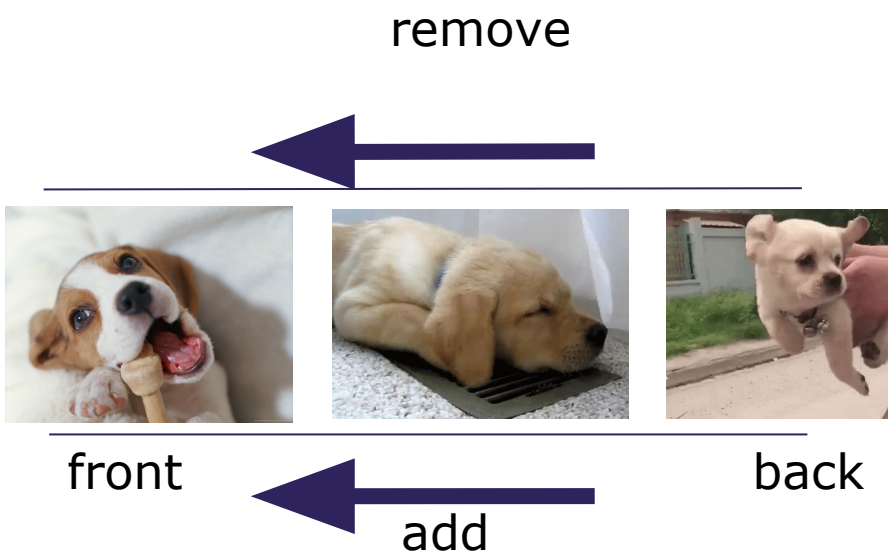
- A **First**-in-First-out (FIFO) data structure
  - Elements are removed in the **same order** to how they were added
- All elements must be of same type*
- Dynamically sized

What is Queue particularly good at?

- `add` - add element to back
- `remove` - remove element from front
- Supported operations are few but *very efficient*



(PCM) Queue



Queues in Computer Science

- Operating systems:
  - queue of print jobs to send to the printer
  - queue of programs / processes to be run
  - queue of network data packets to send

- Programming:
  - modeling a line of customers or clients
  - storing a queue of computations to be performed in order

- Real world examples:
  - people on an escalator or waiting in a line
  - cars at a gas station (or on an assembly line)

(PCM) Programming with Queues

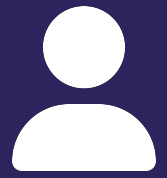
`add(`**value**`)`	places given value at back of queue
`remove()`	removes value from front of queue and returns it; throws a `NoSuchElementException` if queue is empty
`peek()`	returns front value from queue without removing it; returns `null` if queue is empty
`size()`	returns number of elements in queue
`isEmpty()`	returns `true` if queue has no elements

```
Queue<Integer> q = new LinkedList<Integer>();
q.add(42);
q.add(-3);
q.add(17);          // front [42, -3, 17] back
System.out.println(q.remove());    // 42
```

- **IMPORTANT**: When constructing a queue you must use a new `LinkedList` object instead of a new `Queue` object.

Lecture Outline

- Announcements
- Review: Stacks & Queues
- **Queue Manipulation**
- Stack Manipulation
  - Problem Solving



Practice : Think

sli.do #cse122

What is the return of this method?

```
// numbers: top [5, 4, 3, 2, 1] bottom
public static int sum(Stack<Integer> numbers) {
    Queue<Integer> q = new LinkedList<>();

    int total = 0;
    for (int i = 0; i < numbers.size(); i++) {
        int number = numbers.pop();
        total += number;

        q.add(number);
    }

    // Still need to move back to the stack!
    return total;
}
```

A) 0
B) 1
C) 5
D) 12
E) 15
F) Throws an error

Practice : Pair

sli.do #cse122

What is the return of this method?

```
// numbers: top [5, 4, 3, 2, 1] bottom
public static int sum(Stack<Integer> numbers) {
    Queue<Integer> q = new LinkedList<>();

    int total = 0;
    for (int i = 0; i < numbers.size(); i++) {
        int number = numbers.pop();
        total += number;

        q.add(number);
    }

    // Still need to move back to the stack!
    return total;
}
```

A) 0
B) 1
C) 5
D) 12
E) 15
F) Throws an error

Stack Sum bug

```
// numbers: top [5, 4, 3, 2, 1] bottom
public static int sum(Stack<Integer> numbers) {
    Queue<Integer> q = new LinkedList<>();

    int total = 0;
    for (int i = 0; i < numbers.size(); i++) {
        int number = numbers.pop();
        total += number;

        q.add(number);
    }

    // Still need to move back to the stack!
    return total;
}
```

Loop Table

i	total	numbers	numbers.size()
0	5	[4, 3, 2, 1]	
1	9	[3, 2, 1]	4
2	12	[2, 1]	3
3			2

Exit the loop!!

Lecture Outline

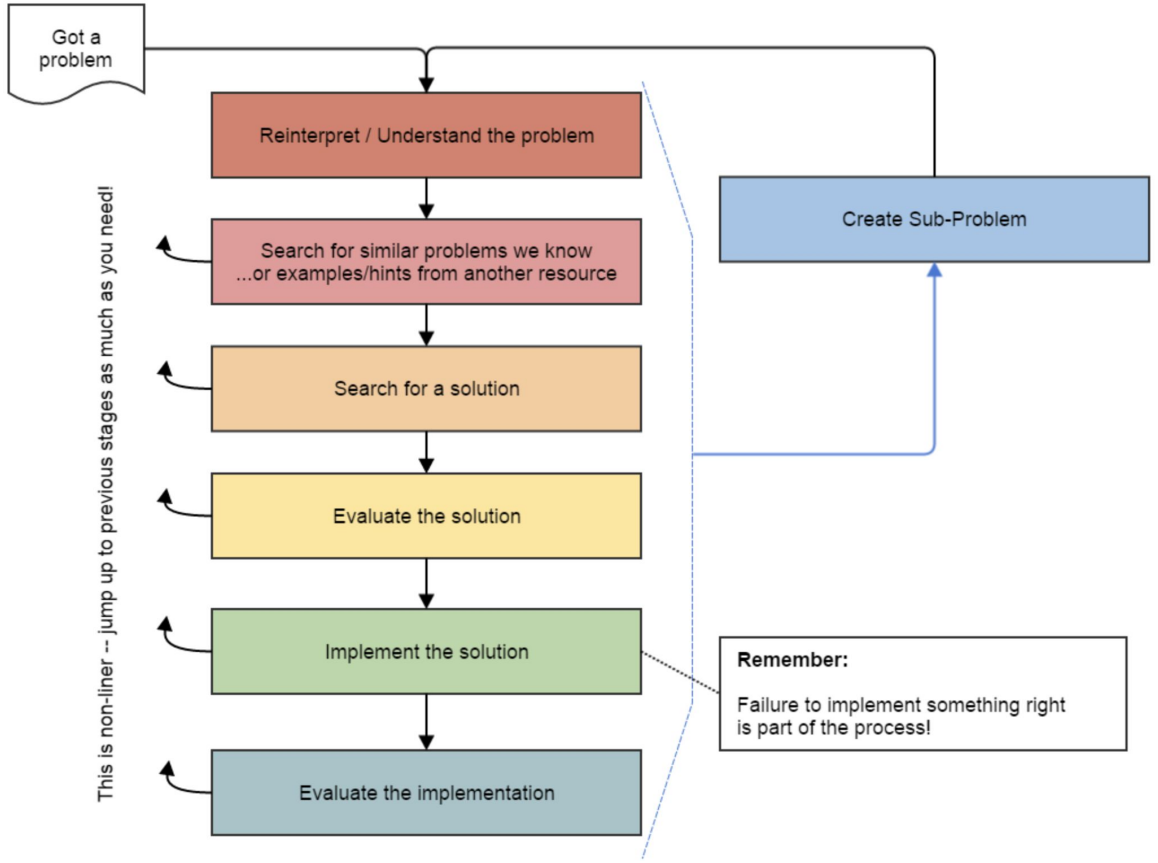
- Announcements
- Review: Stacks & Queues
- Queue Manipulation
- **Stack Manipulation**
  - Problem Solving

Lecture Outline

- Announcements
- Review: Stacks & Queues
- Queue Manipulation
- Stack Manipulation
  - Problem Solving

Problem Solving

- On their own, Stacks & Queues are quite simple with practice (few methods, simple model)
- Some of the problems we ask are complex *because* the tools you have to solve them are restrictive
  - sum(Stack) is hard with a Queue as the auxiliary structure
- We challenge you on purpose here to practice **problem solving**



Source: Oleson, Ko (2016) - Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance

Common Problem-Solving Strategies

- **Analogy** – Is this similar to a problem you've seen?
  - sum(Stack) is probably a lot like sum(Queue), start there!
- **Brainstorming** – Consider steps to solve problem before writing code
  - Try to do an example "by hand" → outline steps
- **Solve Sub-Problems** – Is there a smaller part of the problem to solve?
  - Move to queue first
- **Debugging** – Does your solution behave correctly on the example input.
  - Test on input from specification
  - Test edge cases ("What if the Stack is empty?")
- **Iterative Development** – Can we start by solving a different problem that is easier?
  - Just looping over a queue and printing elements

Common Stack & Queue Patterns

- Stack → Queue and Queue → Stack
  - We give you helper methods for this on problems
- Reverse a Stack with a S→Q + Q→S
- "Cycling" a queue: Inspect each element by repeatedly removing and adding to back `size` times
  - Careful: Watch your loop bounds when queue's size changes
- A "splitting" loop that moves some values to the Stack and others to the Queue

See you Friday!

- Practice with Stacks & Queues in Section
- Challenge problem in lecture on Friday
- P1, released Friday, will use Stacks & Queues.