CSE 373: Data Structures and Algorithms

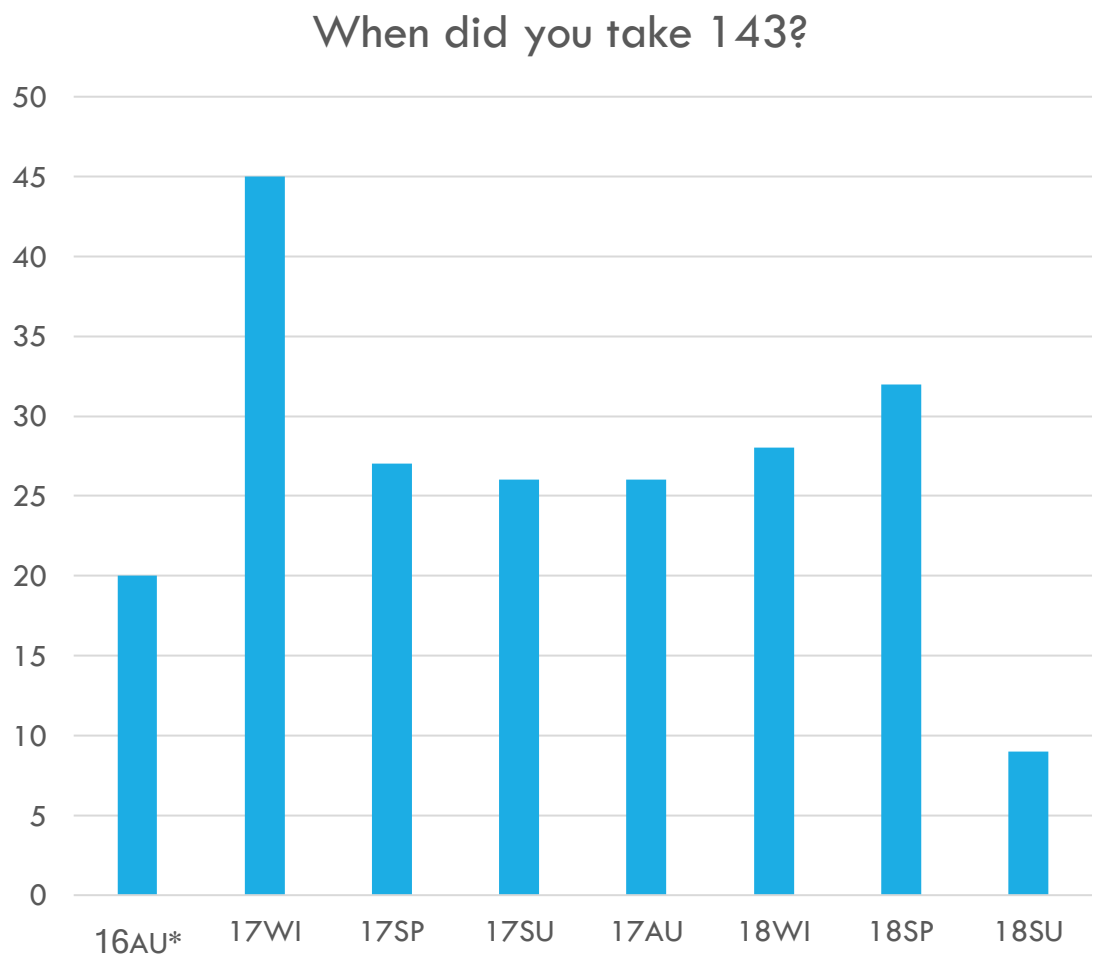
Maps and Iterators

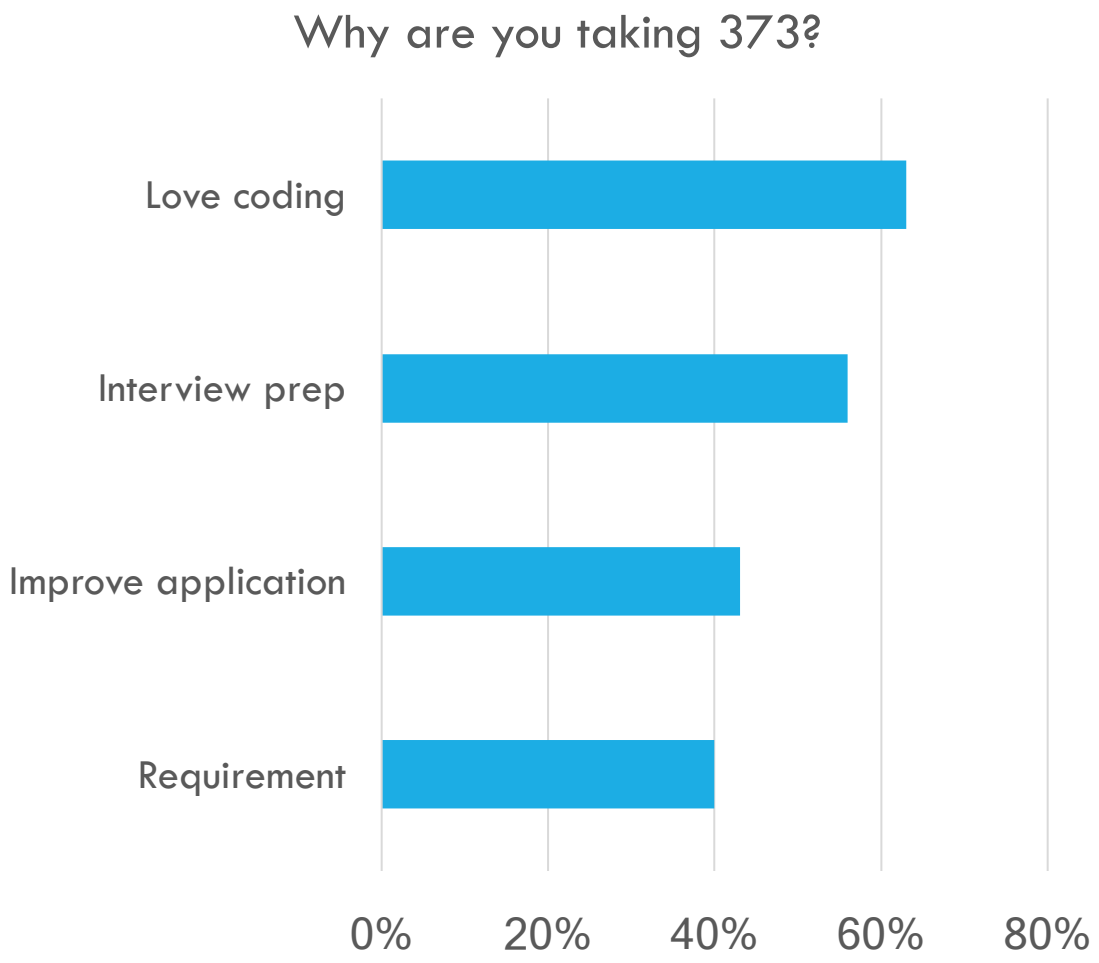
Autumn 2018

Shrirang (Shri) Mare
shri@cs.washington.edu

Thanks to Kasey Champion, Ben Jones, Adam Blank, Michael Lee, Evan McCarty, Whitaker Brand, Zora Fung, Stuart Reges, Justin Hsia, Ruth Anderson, and many others for sample slides and materials ...

Pre-course survey results





Administrivia

- Lecture recordings
- Changes to grading policy
- Materials from the lecture
- Required reading/exercise
- Homework 1 due this Friday (10/5)
- Project 1:
  - Partner selection forms will be out later today, due this Friday
  - Project 1 goes out this Friday
- Eclipse setup

Recap

From last lecture:

- Implementing List ADT with an Array
- Generics
- Implementing Stack ADT with an Array and a Linked List

Today's Goals:

- Map ADT
- Iterators

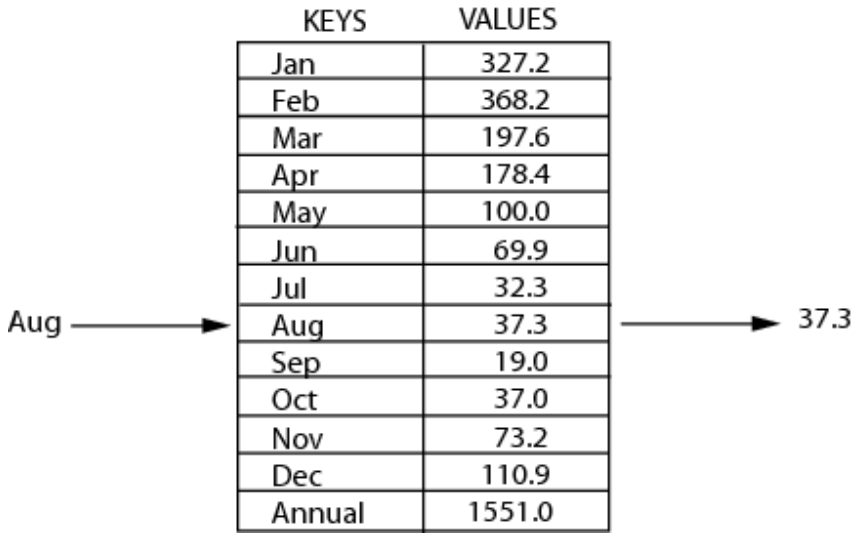
Review: Maps

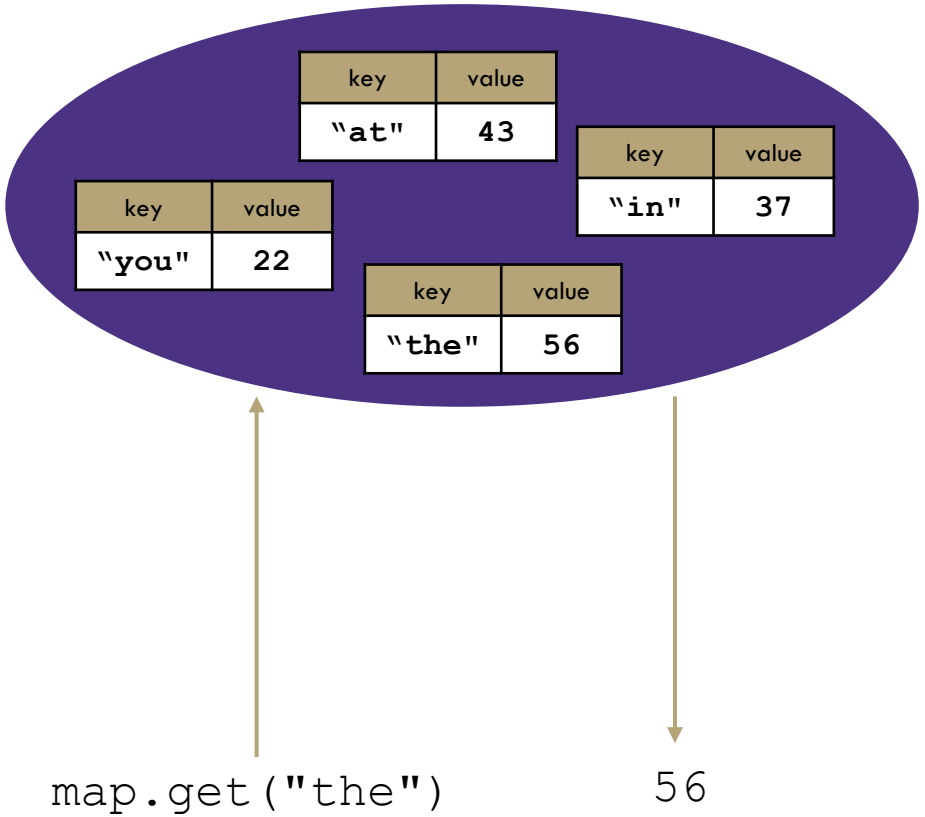
map: Holds a set of unique *keys* and a collection of *values*, where each key is associated with one value.

- a.k.a. "dictionary", "associative array", "hash"

operations:

- **put**(*key*, *value*): Adds a mapping from a key to a value.
- **get**(*key*): Retrieves the value mapped to the key.
- **remove**(*key*): Removes the given key and its mapped value.





Remember: Map ADT

- Keys must be unique
- Key and Value can be of different types
- Expectation: fast lookup, i.e., efficient **get(**key**)**
- Examples:
  - Postal service
  - Database lookups

How would you implement a Map with…

1. Array

2. Linked List

Big-O for Map operations, if implemented with…

Data structure	put	get	remove
Unsorted Array			
Unsorted Linked List			
Sorted Array			
Sorted Linked List			

Case Study: The List ADT

list: stores an ordered sequence of information.

- Each item is accessible by an index.
- Lists have a variable size as items can be added and removed

Supported Operations:

- **get(index):** returns the item at the given index
- **set(value, index):** sets the item at the given index to the given value
- **append(value):** adds the given item to the end of the list
- **insert(value, index):** insert the given item at the given index maintaining order
- **delete(index):** removes the item at the given index maintaining order
- **size():** returns the number of elements in the list

Question

How do we print out all the elements inside a list?

One idea:

```
for (int i = 0; i < myList.size(); i++) {
    System.out.println(myList.get(i));
}
```

How efficient is this if `myList` is:

- An array list:

- A linked list:

Case Study: The List ADT

list: stores an ordered sequence of information.

- Each item is accessible by an index.
- Lists have a variable size as items can be added and removed

Supported Operations:

- **get(index):** returns the item at the given index
- **set(value, index):** sets the item at the given index to the given value
- **append(value):** adds the given item to the end of the list
- **insert(value, index):** insert the given item at the given index maintaining order
- **delete(index):** removes the item at the given index maintaining order
- **size():** returns the number of elements in the list
- **Iterator()**: returns an iterator over the list

The Iterator ADT

An Iterator “wraps” some sequence.

It yields each subsequent element one by one on request.

An iterator “remembers” what it needs to yield next.

Supported operations:

- **hasNext():** returns ‘true’ if there is another element left to yield and ‘false‘ otherwise
- **next():** returns the next element (if there is one)

Question

Why we need an Iterator?

Implementing an iterator

TODO list

- Homework 1 – due this Friday (10/5)
- Find a partner for Project 1