LEC 07

CSE 122

Sets, For-Each Loops, Iterators

Questions during Class?

Raise hand or send here

sli.do #cse122

BEFORE WE START

Talk to your neighbors:

Do you eat breakfast today?

Music: Hunter/Miya's Playlist

Instructor Hunter Schafer / **Miya Natsuhara**

TAs

Ajay	Gaurav	Melissa
Andrew	Hilal	Noa
Anson	Hitesh	Parker
Anthony	Jake	Poojitha
Audrey	Jin	Samuel
Chloe	Joe	Sara
Colton	Joe	Simon
Connor	Karen	Sravani
Elizabeth	Kyler	Tan
Evelyn	Leon	Vivek

Lecture Outline

- Announcements
- Practice Problem
- Sets Review
- Tradeoffs with Different Data Structures
- For-Each Loop
- Iterators

Announcements

- Programming Assignment 1 was due yesterday (Thurs, Oct 20)
- Creative Project 1 released later today
  - Focused on 2D arrays and Images!
- Quiz 1 on Tuesday (Oct 25) in your quiz section
- First opportunity for quiz retakes on Tuesday (Oct 25)

Lecture Outline

- Announcements
- Practice Problem
- Sets Review
- Tradeoffs with Different Data Structures
- For-Each Loop
- Iterators

Practice Problem:

Write a program that, given a `Scanner` over a large text file (e.g., *Moby Dick* or the King James Bible), counts the number of *unique words* in the text.

Lecture Outline

- Announcements
- Practice Problem
- **Sets Review** 
- Tradeoffs with Different Data Structures
- For-Each Loop
- Iterators

(PCM) Sets (ADT)

- A collection of unique values (no duplicates allowed) that can perform the following operations *efficiently*:
  - add
  - remove
  - search (contains)

- We don't think of a set as having indices; we just add things to the set in general and don't worry about order

(PCM) Sets in Java

- `Set` is an interface in Java
  - In `java.util`

- `HashSet` and `TreeSet` are classes that implement the `Set` interface in Java
  - `HashSet`: Very fast! Implemented using a "hash table" array
    - *Elements are stored in an unpredictable order*
  - `TreeSet`: Pretty fast! Implemented using a "binary search tree"
    - *Elements are stored in sorted order*

Set Methods

Method	Description
`add(value)`	Adds the given value to the set
`contains(value)`	Returns `true` if the given value is found in this set
`remove(value)`	Removes the given value from the set; returns `true` if the set contained the value, `false` if not
`clear()`	Removes all elements from the set
`size()`	Returns the number of elements in list
`isEmpty()`	Returns `true` if the set's size is 0; `false` otherwise
`toString()`	Returns a `String` representation of the set such as `"[3, 42, -7, 15]"`

Lecture Outline

- Announcements
- Practice Problem
- Sets Review
- **Tradeoffs with Different Data Structures**
- For-Each Loop
- Iterators

Choosing a Data Structure: Tradeoffs

- You got a bit of practice with this in your quiz sections on Tuesday!
  - Solving the same problem with an `ArrayList`, a `Stack`, and a `Queue`

- Things to consider:
  - Functionality
    - If you need duplicates or indexing, `Sets` are not for you!
  - Efficiency
    - Different data structures are “good at” different things!

Lecture Outline

- Announcements
- Practice Problem
- Sets Review
- Tradeoffs with Different Data Structures
- **For-Each Loop** 
- Iterators

For-Each Loop

- A new kind of loop!

```
Set<String> words = new HashSet<>();
for (String s : words) {
    System.out.println(s);
}
```

- BUT, you cannot *modify* the data structure inside a for-each loop
  - You will get a `ConcurrentModificationException`
  - They are “read-only”

Lecture Outline

- Announcements
- Practice Problem
- Sets Review
- Tradeoffs with Different Data Structures
- For-Each Loop
- **Iterators** 

Iterators

A new object that has access to all of the elements of a given structure and can give them to you, one at a time.

Iterators

- Returned by the `iterator()` method

Methods	Description
`hasNext()`	Returns `true` if there are more elements for the iterator to return
`next()`	Returns the next element in the iteration
`remove()`	Removes and returns the element that was last returned by `next()`

- You must use the iterator's `remove()` method to remove things from what you're iterating over – otherwise you will get a `ConcurrentModificationException`