

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:
What is your favorite study spot on or
near campus?*

Instructor

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

Announcements

- Programming Assignment 1 was due yesterday (Thurs, Apr 20)
- Creative Project 1 released today
 - Focused on 2D arrays and Images!
- Resubmission and Retake forms for next week posted


Lecture Outline

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

Count Unique Words


Write a program that, given a Scanner over a large text file (e.g., *Moby Dick*), 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



```
"hi" "hola"  
"bonjour" "hello"  
"konichiwa"
```

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

Count Unique Words (version 2)

- Write a program that, given a Scanner over a large text file (e.g., *Moby Dick*), 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

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”



Practice : Think

[sli.do](#)[#cse-122](#)

What output is produced by this code?

```
Set<Integer> nums = new TreeSet<>();  
nums.add(3);  
nums.add(9);  
nums.add(3);  
nums.add(-2);  
nums.add(0);  
  
for (int n : nums) {  
    System.out.print(n + " ");  
}
```

A) 3 9 3 -2 0

B) -2 0 3 3 9

C) 9 3 0 -2

D) -2 0 3 9

E) ConcurrentModificationException



Practice : Pair

sli.do

#cse-122

What output is produced by this code?

```
Set<Integer> nums = new TreeSet<>();  
nums.add(3);  
nums.add(9);  
nums.add(3);  
nums.add(-2);  
nums.add(0);  
  
for (int n : nums) {  
    System.out.print(n + " ");  
}
```

A) 3 9 3 -2 0


B) -2 0 3 3 9

C) 9 3 0 -2

D) -2 0 3 9

E) ConcurrentModificationException

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 |
|------------------------|---|
| <code>hasNext()</code> | Returns true if there are more elements for the iterator to return |
| <code>next()</code> | Returns the next element in the iteration |
| <code>remove()</code> | Removes and returns the element that was last returned by <code>next()</code> |

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