Sets, For-Each Loops, Iterators

Talk to your neighbors:

Did you eat breakfast today?

Music: 122 24wi Lecture Tunes 🎼

Instructors

Miya Natsuhara and Joe Spaniac

TAs

Lecture Outline

• Announcements

• Practice Problem

• Sets Review

• Tradeoffs with Different Data Structures

• For-Each Loop

• Iterators
Announcements

• Programming Assignment 1 (P1) due tomorrow!
  - Stacks, Queues, Exceptions

• Resubmission Cycle 1 was due yesterday
  - Resubmission Cycle 2 will open tomorrow

• Heads up: Quiz 1 scheduled for Tuesday, Feb 13
  - ArrayLists, Reference Semantics, Stacks and Queues, Sets, Maps

• **How to Use the IPL**

• Programming Assignment 2 released on Friday, Feb 2
  - Yes, two Programming Assignments in a row
  - BUT, you have *two weeks* to complete this assignment
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

"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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(value)</td>
<td>Adds the given value to the set, returns whether or not the given value was added successfully</td>
</tr>
<tr>
<td>contains(value)</td>
<td>Returns true if the given value is found in this set</td>
</tr>
<tr>
<td>remove(value)</td>
<td>Removes the given value from the set; returns true if the set contained the value, false if not</td>
</tr>
<tr>
<td>clear()</td>
<td>Removes all elements from the set</td>
</tr>
<tr>
<td>size()</td>
<td>Returns the number of elements in list</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>Returns true if the set’s size is 0; false otherwise</td>
</tr>
<tr>
<td>toString()</td>
<td>Returns a String representation of the set such as &quot;[3, 42, -7, 15]&quot;</td>
</tr>
</tbody>
</table>
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!

```java
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”
What output is produced by this code?

```java
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. -2 0 3 9
B. 3 9 3 -2 0
C. 9 3 0 -2
D. -2 0 3 3 9
E. ConcurrentModificationException
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. -2 0 3 9
B. 3 9 3 -2 0
C. 9 3 0 -2
D. -2 0 3 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

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasNext()</td>
<td>Returns true if there are more elements for the iterator to return</td>
</tr>
<tr>
<td>next()</td>
<td>Returns the next element in the iteration</td>
</tr>
<tr>
<td>remove()</td>
<td>Removes and returns the element that was last returned by next()</td>
</tr>
</tbody>
</table>

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