Queues, Debugging, and Assignment 3

CS187 Data Structures
February 24, 2010
Chris Vigorito
Queues

- First-In-First-Out (FIFO) data structure
- Commonly used in producer/consumer scenarios, buffers, etc.
- Can be standard or prioritized by some key
- Many ways to implement
Queue

• Interface that provides standard queue functionality
• Defines two set of methods for insertion, removal, and examination
• One set throws exception if operation fails
• Other set returns a special value (null or false)
Queue

• **Insertion**: `add(e)` / `offer(e)` methods
  - Inserts element `e` (type defined through generics) at tail of queue if queue is not full

• **Removal**: `remove()` / `poll()`
  - Pulls off and returns element at head of queue if queue is not empty

• **Examination**: `element()` / `peek()`
  - Returns element at head of queue if queue is not empty but leaves element in queue
# Queue

<table>
<thead>
<tr>
<th>Action</th>
<th>Throws exception</th>
<th>Returns special value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert</strong></td>
<td><code>add(e)</code></td>
<td><code>offer(e) [false]</code></td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td><code>remove()</code></td>
<td><code>poll() [null]</code></td>
</tr>
<tr>
<td><strong>Examine</strong></td>
<td><code>element()</code></td>
<td><code>peek() [null]</code></td>
</tr>
</tbody>
</table>
Implementing Classes

- Multiple classes depending on functionality you want
- For simplest, standard queue, `LinkedList` seems most straightforward
- You’ll learn more about how they work later
- `PriorityQueue` implements prioritized queue
- Uses either natural ordering or a specified `Comparator` to prioritize
Iterating with Queues

- For standard queues (e.g., LinkedList), a ListIterator can be used to iterate through list in correct order
  - Retrieved by calling listIterator() method on list object

- For priority queues, standard Iterator can be used (retrieved via iterator() method)
  - Does not iterate through elements in any particular order
Example

```java
LinkedList<Integer> queue = new LinkedList<Integer>();
queue.add(4);
queue.offer(8);
queue.add(15);
System.out.println(queue.remove());  // 4
System.out.println(queue.poll());    // 8
System.out.println(queue.peek());    // 15
System.out.println(queue.element()); // 15
queue.add(16);
queue.add(23);
queue.add(42);
ListIterator<Integer> it = queue.listIterator();
while (it.hasNext())
    System.out.print(it.next() + " "); // 15 16 23 42
System.out.println();
System.out.println(queue.remove()); // 15
```
Assignment 3

• Finding all solutions:
  • Just treat finding a solution the same as reaching a “dead end” and backtrack
  • Only difference is that you print out the solution in those cases as well
  • Only stop when stack is empty and you’ve reached end of first row
Assignment 3

• Checking valid placements:
  • Iterate through current stack and check each element with potential placement
  • Use formula discussed in class (others are possible)
  • Same column is straightforward
  • Same diagonal: if difference in # of rows is same as difference in # of columns, then no good
Assignment 3

• Iterating through stack:
  • Can create a copy and pop elements off copy one at a time (acceptable but inefficient)
  • Can use a for loop and use stack’s get(i) method
• Use an iterator
Debugging

• Majority of time spent coding large programs
• Can use print statements to debug, but for large programs this get very messy and confusing
• Most IDEs come with debug tools
• Basics will be same for all IDEs - specifics and user interface will differ
Basics

- Breakpoints - points in program you want to pause execution at while running
- Stepping through program - step into, step over, step return
- Allow you to move through program at different rates
- Watching variables/expressions - like print statements that stick around and update
Tutorial Videos

- http://eclipsutorial.sourceforge.net/debugger.html
- 7 videos - first two cover basics and should be sufficient (each is 15 min)
- Windows UI - Mac shortcuts/menus will differ
- Sample project available for download