Lecture 4: Stacks and Queues

06/29/22
A1: LetterInventory

- Due Thursday 6/30 @ 11:59pm

- To be making **satisfactory** progress in the course, your homework should pass all the test cases on Ed.
Abstract Data Type

Abstract Data Type (ADT)

• Composed of:
  • A collection of data
  • The operations that can be performed on that Data

• Describes what a collection does, not how it does it

• Not specific to Java!
Interface

• Java’s way of representing an Abstract Data Type
• Describes all the methods a class must have in order to be that data type
• Doesn’t implement the methods
  • A class with all the guts ripped out
2 New Abstract Data Types!

• queue
  • line at a grocery store

• stack
  • stack of cafeteria trays
Queue Example

front

remove

add

back
Queue ADT

• **Queue**: First-In, First-Out ("FIFO")
  • No indices

• basic queue operations:
  • **add** (enqueue): Add an element to the back.
  • **remove** (dequeue): Remove the front element.
  • **peek**: Examine the front element.
Queues in Computer Science

• Operating systems:
  • queue of print jobs to send to the printer
  • queue of programs / processes to be run

• Programming:
  • modeling a line of customers or clients
  • storing a queue of computations to be performed in order

• Real world examples:
  • people on an escalator or waiting in a line
  • cars at a gas station (or on an assembly line)
Queue Interface in Java

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add(value)</code></td>
<td>places given value at back of queue</td>
</tr>
<tr>
<td><code>remove()</code></td>
<td>removes value from front of queue and returns it;</td>
</tr>
<tr>
<td></td>
<td>throws a <code>NoSuchElementException</code> if queue is empty</td>
</tr>
<tr>
<td><code>peek()</code></td>
<td>returns front value from queue without removing it;</td>
</tr>
<tr>
<td></td>
<td>returns <code>null</code> if queue is empty</td>
</tr>
<tr>
<td><code>size()</code></td>
<td>returns number of elements in queue</td>
</tr>
<tr>
<td><code>isEmpty()</code></td>
<td>returns <code>true</code> if queue has no elements</td>
</tr>
</tbody>
</table>

Queue has other methods that are off-limits (not efficient)

```java
Queue<String> q = new LinkedList<>();
```

`LinkedList` implements the Queue interface!
Stack Example

drawings of characters showing push and pop operations.
Stack ADT

• **Stack**: Last-In, First-Out ("LIFO")
  • No indices

• basic stack operations:
  • **push**: Add an element to the top.
  • **pop**: Remove the top element.
  • **peek**: Examine the top element.
Stacks in Computer Science

• Programming languages and compilers:
  • method calls are placed onto a stack (*call*=push, *return*=pop)
  • compilers use stacks to evaluate expressions

• Matching up related pairs of things:
  • examine a file to see if its braces `{ }` match
  • convert "infix" expressions to pre/postfix

• Sophisticated algorithms:
  • searching through a maze with "backtracking"
  • many programs use an "undo stack" of previous operations
**Stack Class in Java**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack&lt;E&gt;()</td>
<td>constructs a new stack with elements of type E</td>
</tr>
<tr>
<td>push(value)</td>
<td>places given value on top of stack</td>
</tr>
<tr>
<td>pop()</td>
<td>removes top value from stack and returns it; throws EmptyStackException if stack is empty</td>
</tr>
<tr>
<td>peek()</td>
<td>returns top value from stack without removing it; throws EmptyStackException if stack is empty</td>
</tr>
<tr>
<td>size()</td>
<td>returns number of elements in stack</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>returns true if stack has no elements</td>
</tr>
</tbody>
</table>

Stack has other methods that are off-limits (not efficient)

```java
Stack<String> s = new Stack<>();
```

Java messed up, there is no Stack interface 😞
Misc. Notes

• Lecture and section problems are brainteasers, not great applications of stacks and queues
  • Practice problem solving!

• (Reminder: Exam problems are exactly like section problems! We’re not trying to surprise you)
  • peek() method isn’t allowed on exam/section questions