Sequences and Iterators

Sequence ADT (§ 5.3)
- The Sequence ADT is the union of the Vector and List ADTs
- Elements accessed by
  - Rank, or
  - Position
- Generic methods:
  - size(), isEmpty()
- Vector-based methods:
  - elemAtRank(r), replaceAtRank(r, o), insertAtRank(r, o), removeAtRank(r)
- List-based methods:
  - first(), last(), prev(p), next(p), replace(p, o), insertBefore(p, o), insertAfter(p, o), insertFirst(o), insertLast(o), remove(p)
- Bridge methods:
  - atRank(r), rankOf(p)

Applications of Sequences
- The Sequence ADT is a basic, general-purpose, data structure for storing an ordered collection of elements
- Direct applications:
  - Generic replacement for stack, queue, vector, or list
  - small database (e.g., address book)
- Indirect applications:
  - Building block of more complex data structures

Linked List Implementation
- A doubly linked list provides a reasonable implementation of the Sequence ADT
- Nodes implement Position and store:
  - element
  - link to the previous node
  - link to the next node
- Special trailer and header nodes
- Position-based methods run in constant time
- Rank-based methods require searching from header or trailer while keeping track of ranks; hence, run in linear time
Array-based Implementation

- We use a circular array storing positions.
- A position object stores:
  - `Element`
  - `Rank`
- Indices `f` and `l` keep track of first and last positions.

Sequence Implementations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Array</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>size, isEmpty</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><code>atRank</code>, <code>rankOf</code>, <code>elemAtRank</code></td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td>first, last, prev, next</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>replace</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>replaceAtRank</td>
<td>1</td>
<td>n</td>
</tr>
<tr>
<td><code>insertAtRank</code>, <code>removeAtRank</code></td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td><code>insertFirst</code>, <code>insertLast</code></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><code>insertAfter</code>, <code>insertBefore</code></td>
<td>n</td>
<td>1</td>
</tr>
<tr>
<td>remove</td>
<td>n</td>
<td>1</td>
</tr>
</tbody>
</table>

Iterators (§ 5.4)

- An iterator abstracts the process of scanning through a collection of elements.
- Methods of the `ObjectIterator` ADT:
  - `object object()`
  - `boolean hasNext()`
  - `object next(Object)`
  - `reset()`
- Extends the concept of Position by adding a traversal capability.
- Implementation with an array or singly linked list.

An iterator is typically associated with another data structure.
We can augment the `Stack`, `Queue`, `Vector`, `List` and `Sequence` ADTs with method:
- `ObjectIterator elements()`

Two notions of iterator:
- `snapshot`: freezes the contents of the data structure at a given time
- `dynamic`: follows changes to the data structure.