Sequence
Sequence

- The **Sequence** component family allows you to manipulate strings of entries of any (arbitrary) type through **direct access** by position, similar to an array
  - Another generic type like **Queue** and **Set**
  - One possible **best practice** alternative to the built-in Java array, from the OSU CSE components
Interfaces and Classes

Standard

Sequence-Kernel

Sequence

Sequence1L implements

Sequence2L implements

Sequence3 implements
Interfaces and Classes

Standard

Sequence

Sequence1L

Sequence2L

Sequence3

Sequence extends Standard

Standard has contracts for three methods: clear, newInstance, transferFrom

Sequence-2-Kernel extends Sequence
Interfaces and Classes

**Standard**

Sequence-Kernel extends Standard

Sequence-Kernel has contracts for three methods:
- add
- remove
- length

Sequence-Kernel extends

Sequence2L implements Sequence

Sequence2L implements

Sequence3 implements Sequence2L
Sequence

has contracts for six other methods:

- entry
- replaceEntry
- append
- flip
- insert
- extract

Sequence

extends

Standard

Sequence-

kernel

extends

Sequence

implements

Sequence1L

implements

Sequence2L

implements

Sequence3
Mathematical Model

• The value of a Sequence variable is modeled as a string of entries of type $T$

• Formally:

```
type Sequence is modeled by string of $T$
```
No-argument Constructor

• Ensures:

\[ \text{this} = < > \]
Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sequence&lt;Integer&gt; si = new Sequence1L&lt;&gt;();</code></td>
<td></td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sequence&lt;Integer&gt; si = new Sequence1L&lt;&gt;();</code></td>
<td><code>si = &lt; &gt;</code></td>
</tr>
</tbody>
</table>
add

```c
void add(int pos, T x)
```

- Adds `x` at position `pos` of `this`.
- Aliases: reference `x`
- Updates: `this`
- Requires:
  
  \[
  0 \leq pos \quad \text{and} \quad pos \leq |this|
  \]
- Ensures:
  
  \[
  this = this[0, pos) \times <x> \times this[pos, |this|)
  \]
**Example**

<table>
<thead>
<tr>
<th><strong>Code</strong></th>
<th><strong>State</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>si.add(1, z);</td>
<td>$si = \langle 49, 3 \rangle$</td>
</tr>
<tr>
<td></td>
<td>$z = 70$</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>si.add(1, z);</code></td>
<td><code>si = &lt; 49, 3 &gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>z = 70</code></td>
</tr>
<tr>
<td></td>
<td><code>si = &lt; 49, 70, 3 &gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>z = 70</code></td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_i = &lt;49, 3&gt;$</td>
</tr>
<tr>
<td>$z = 70$</td>
</tr>
</tbody>
</table>

Note the alias created here, which you cannot see in the tracing table; you should be able to draw the appropriate diagram showing it.
T remove(int pos)
• Removes and returns the entry at position pos of this.
• Updates: this
• Requires:
  \[ 0 \leq pos \quad \text{and} \quad pos < |this| \]
• Ensures:
  \[
  this = this[0, pos) *
  this[pos+1, |this|) \quad \text{and} \quad <\text{remove}> = this[pos, pos+1]
  \]
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
</table>
| $z = \text{si.remove}(1);$ | $\text{si} = \langle 49, 3, 70 \rangle$
|              | $z = -584$             |
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>si = &lt; 49, 3, 70 &gt;</td>
<td>si = &lt; 49, 3, 70 &gt;</td>
</tr>
<tr>
<td>z = -584</td>
<td>z = 3</td>
</tr>
<tr>
<td>z = si.remove(1);</td>
<td></td>
</tr>
</tbody>
</table>
length

int length()

• Reports the length of this.
• Ensures:

\[ length = |this| \]
entry

T entry(int pos)

• Reports the entry at position pos of this.
• Aliases: reference returned by entry
• Requires:
  \[ 0 \leq pos \text{ and } pos < |this| \]
• Ensures:
  \[ <entry> = this[pos, pos+1) \]
### Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>si = &lt; 49, 3, 70 &gt;</code></td>
<td><code>z = -584</code></td>
</tr>
<tr>
<td><code>z = si.entry(1);</code></td>
<td></td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( si = &lt; 49, 3, 70 &gt; )</td>
</tr>
<tr>
<td></td>
<td>( z = -584 )</td>
</tr>
<tr>
<td>( z = si.\text{entry}(1); )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( si = &lt; 49, 3, 70 &gt; )</td>
</tr>
<tr>
<td></td>
<td>( z = 3 )</td>
</tr>
</tbody>
</table>
Note the alias created here, which you cannot see in the tracing table; you should be able to draw the appropriate diagram showing it.
replaceEntry

T replaceEntry(int pos, T x)

• Replaces the entry at position pos of this with x, and returns the old entry at that position.
• Aliases: reference x
• Updates: this
• Requires: 
  \[ 0 <= pos \text{ and } pos < |this| \]
• Ensures:
  \[
  this = \#this[0, pos) \ast <x> \ast \#this[pos+1, |\#this|) \text{ and } <\text{replaceEntry}> = \#this[pos, pos+1]
  \]
### Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w = \text{si.replaceEntry}(1, z); )</td>
<td>( \text{si} = &lt;49, 70&gt; )</td>
</tr>
<tr>
<td></td>
<td>( z = -8 )</td>
</tr>
<tr>
<td></td>
<td>( w = -584 )</td>
</tr>
</tbody>
</table>
# Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
</table>
| \( w = si.\text{replaceEntry}(1, z); \) | \( si = < 49, 70 > \)  
  \( z = -8 \)  
  \( w = -584 \) |

\( si = < 49, -8 > \)  
\( z = -8 \)  
\( w = 70 \)
Example

Note the alias created here, which you cannot see in the tracing table; you should be able to draw the appropriate diagram showing it.

$\text{State}$

- $si = <49, 70>$
- $z = -8$
- $w = -584$

- $si = <49, -8>$
- $z = -8$
- $w = 70$
Another Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
</table>
| $z = si.replaceEntry(1, z);$ | $si = < 49, 70 >$  
|                       | $z = -8$         |
## Another Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>z = si.replaceEntry(1, z);</code></td>
<td><code>si = &lt; 49, 70 &gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>z = -8</code></td>
</tr>
<tr>
<td></td>
<td><code>si = &lt; 49, -8 &gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>z = 70</code></td>
</tr>
</tbody>
</table>
Another Example

This use of the method avoids creating an alias: it *swaps* $z$ with the entry previously at position 1.

<table>
<thead>
<tr>
<th>State</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$si = \langle 49, 70 \rangle$</td>
<td>$z = -8$</td>
</tr>
<tr>
<td>$z = \text{si.replaceEntry}(1, z)$</td>
<td></td>
</tr>
<tr>
<td>$si = \langle 49, -8 \rangle$</td>
<td>$z = 70$</td>
</tr>
</tbody>
</table>
void append(Sequence<T> s)
• Concatenates ("appends") s to the end of this.
• Updates: this
• Clears: s
• Ensures:
  \[ \text{this} = \#\text{this} \times \#s \]
flip

```c
void flip()
• Reverses ("flips") this.
• Updates: this
• Ensures:
  this = rev(#this)
```
void insert(int pos, Sequence<T> s)

• Inserts s at position pos of this, and clears s.
• Updates: this
• Clears: s
• Requires:
  \( 0 \leq pos \quad \text{and} \quad pos \leq |\text{this}| \)
• Ensures:
  \[ \text{this} = \#\text{this}[0, pos) \times \#s \times \#\text{this}[pos, |\#\text{this}|) \]
## Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>si1 = &lt; 8, 6, 92 &gt;</code></td>
<td><code>si2 = &lt; 1, -7 &gt;</code></td>
</tr>
<tr>
<td><code>si1.insert(2, si2);</code></td>
<td></td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th><strong>Code</strong></th>
<th><strong>State</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>si1 = <code>&lt; 8, 6, 92 &gt;</code>&lt;br&gt;si2 = <code>&lt; 1, -7 &gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>si1.insert(2, si2);</code></td>
<td></td>
</tr>
<tr>
<td><code>si1 = </code>&lt; 8, 6, 1, -7, 92 &gt;<code>&lt;br&gt;si2 = </code>&lt; &gt;`</td>
<td></td>
</tr>
</tbody>
</table>
extract

```cpp
void extract(int pos1, int pos2, Sequence<T> s)
```

- Removes the substring of `this` starting at position `pos1` and ending at position `pos2-1`, and puts it in `s`.
- Updates: `this`
- Replaces: `s`
- Requires:
  
  
  \[0 \leq pos1 \text{ and } pos1 \leq pos2 \text{ and } pos2 \leq |this|\]
- Ensures:

  \[
  this = #this[0, pos1) \times #this[pos2, |#this|) \quad \text{and} \\
  s = #this[pos1, pos2)
  \]
### Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
</table>
|     | $si_1 = < 8, 6, 92, 27, 0 >$
|     | $si_2 = < 1, -7, 562 >$
|     | `si1.extract(1, 3, si2);` |
### Example

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
</table>
| si1 = < 8, 6, 92, 27, 0 >  
| si2 = < 1, -7, 562 >      | si1.extract(1, 3, si2); |
| si1 = < 8, 27, 0 >        
| si2 = < 6, 92 >           |
Resources

• OSU CSE Components API: Sequence
  – http://web.cse.ohio-state.edu/software/common/doc/