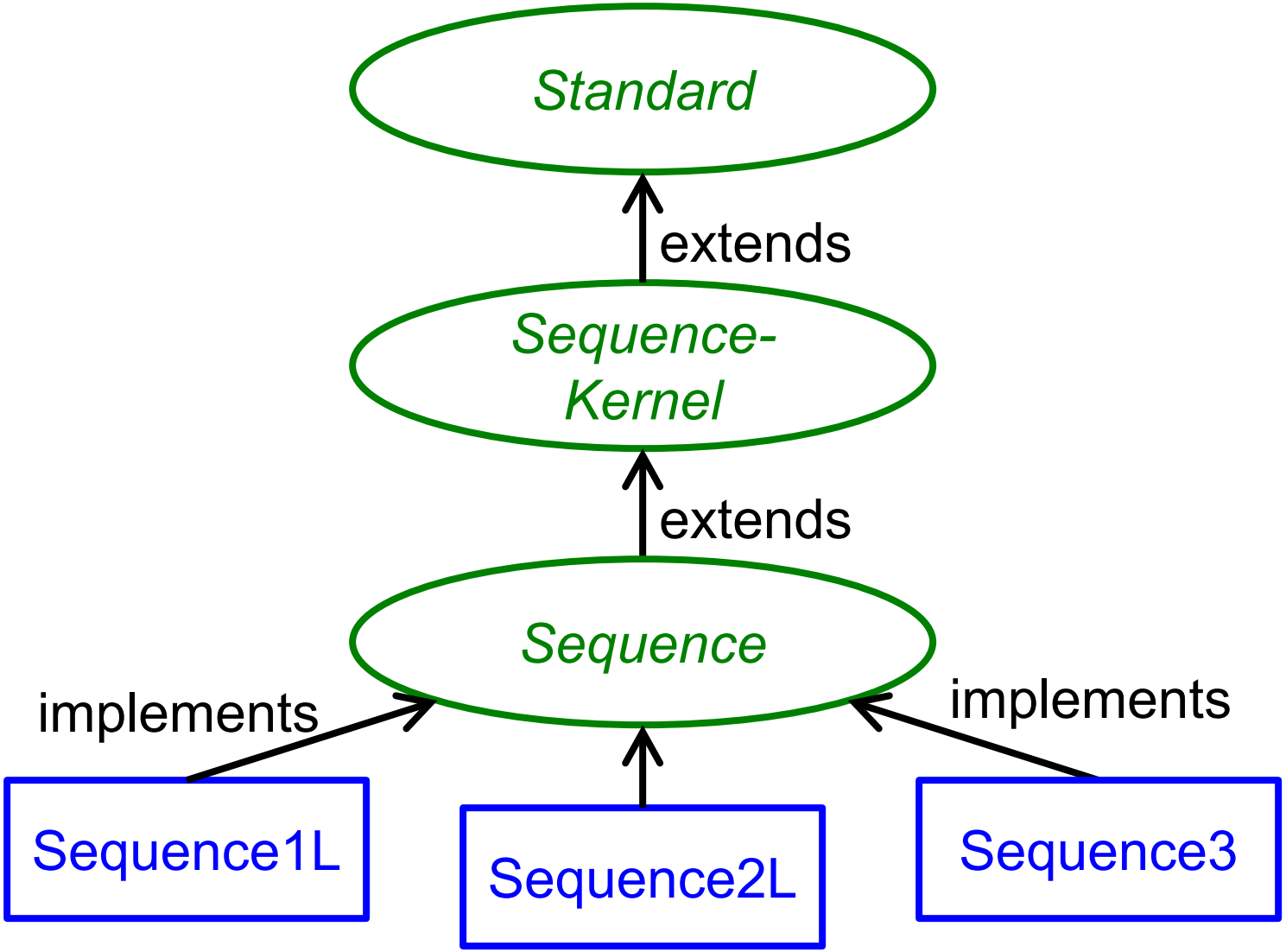
Sequence

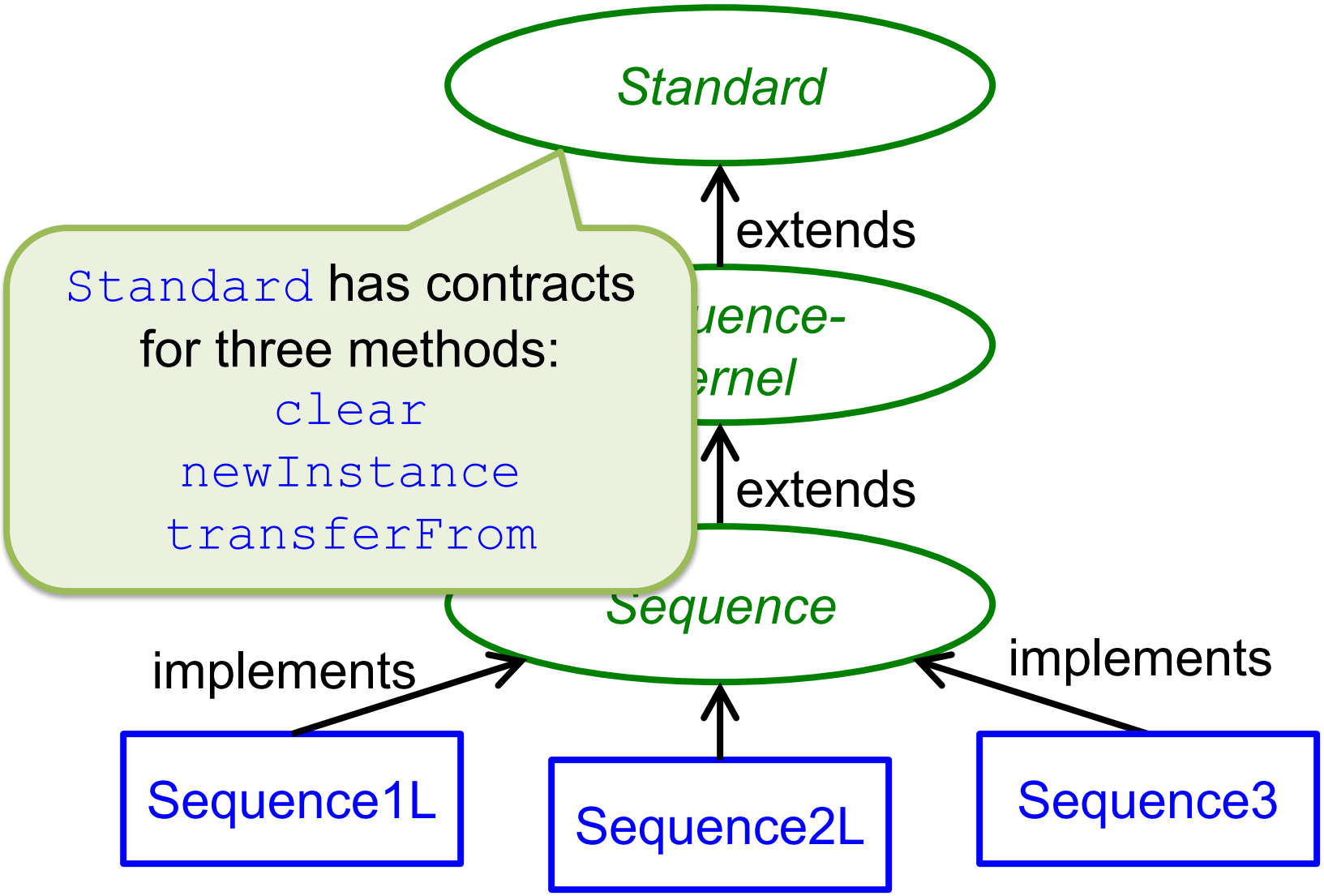
Sequence

- 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

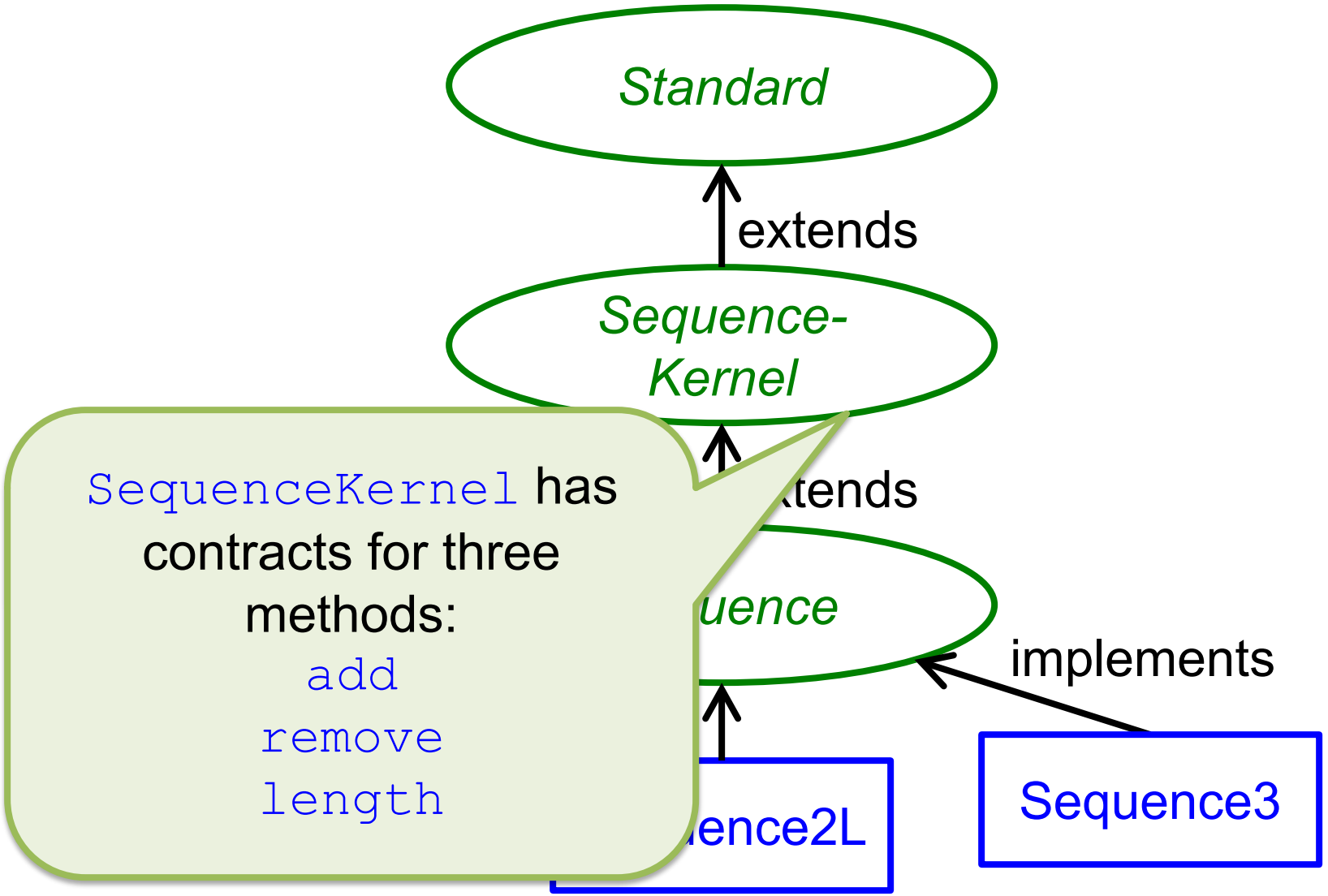
Interfaces and Classes



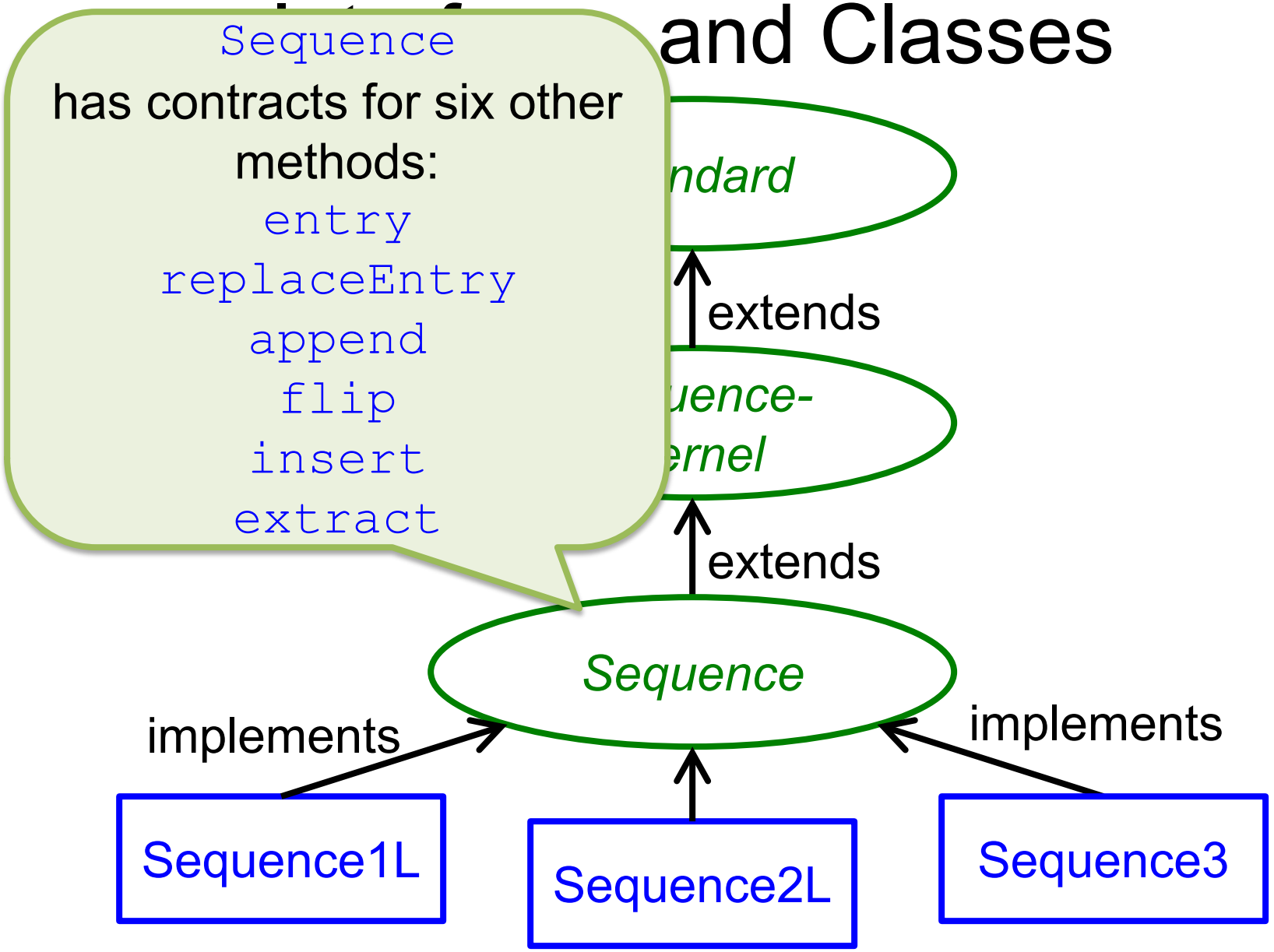
Interfaces and Classes



Interfaces and Classes



and Classes


Sequence
has contracts for six other methods:
entry
replaceEntry
append
flip
insert
extract
ndard
extends
uence-
ernel
extends
Sequence
implements
implements
Sequence1L
Sequence2L
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:

  this = < >

Example

Code	State
`Sequence<Integer> si = new Sequence1L<>();`	

Example

Code	State
`Sequence<Integer> si = new Sequence1L<>();`	
	si = < >

add

void add(**int** pos, T x)

- Adds x at position pos of **this**.
- Aliases: reference x
- Updates: **this**
- Requires:

  0 <= pos **and** *pos <= |this|*

- Ensures:

  this *= #**this**[0, pos) * <x> * #**this**[pos, |#**this**|)*

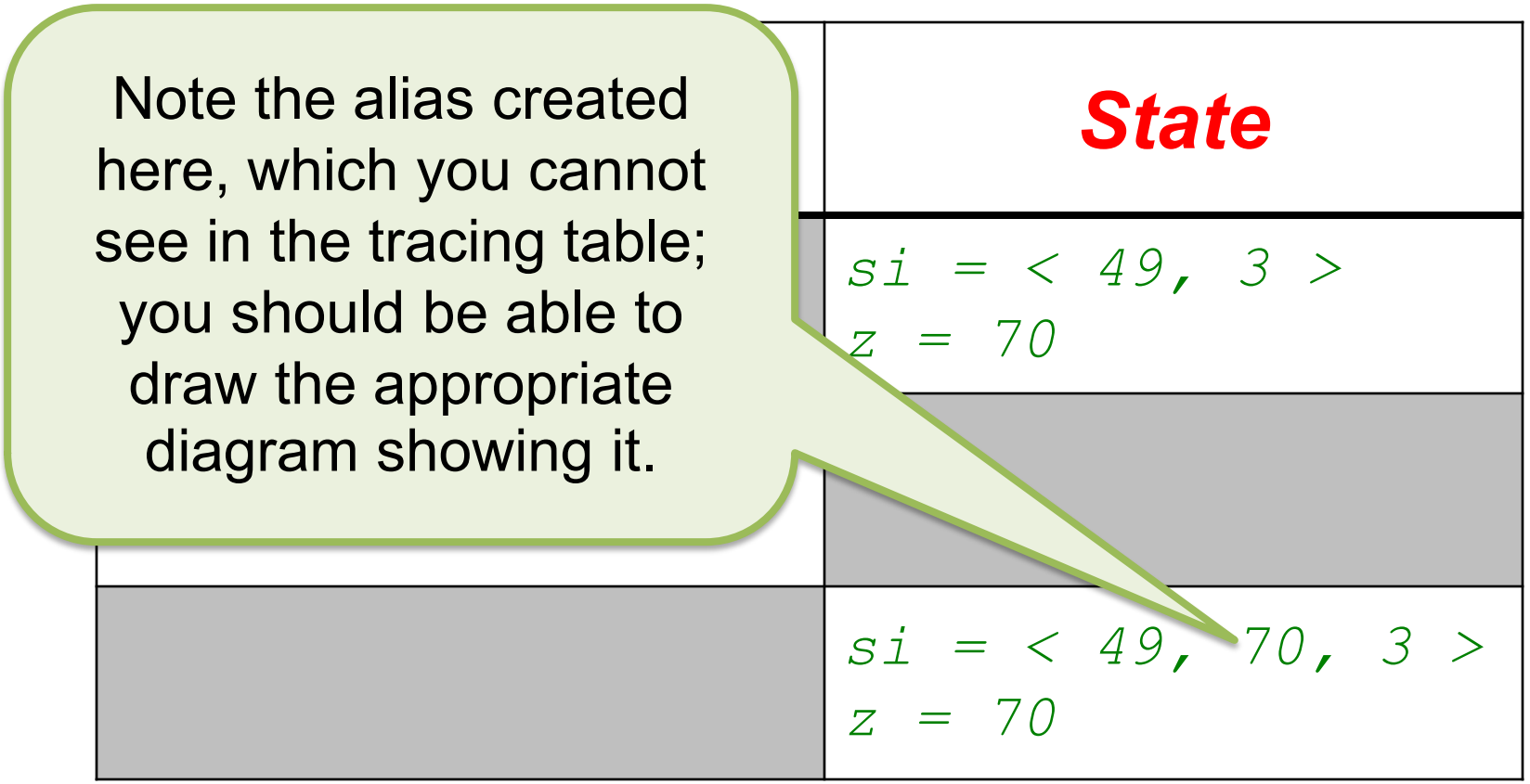
Example

Code	State
	si = < 49, 3 > *z = 70*
`si.add(1, z);`	

Example

Code	State
	si = < 49, 3 > *z = 70*
`si.add(1, z);`	
	si = < 49, 70, 3 > *z = 70*

Example


State
si = < 49, 3 >
z = 70
si = < 49, 70, 3 >
z = 70
Note the alias created here, which you cannot see in the tracing table; you should be able to draw the appropriate diagram showing it.

remove

`T remove(int pos)`

- Removes and returns the entry at position `pos` of **`this`**.
- Updates: **`this`**
- Requires:

  `0 <= pos  and  pos < |this|`

- Ensures:

```
this = #this[0, pos) *
       #this[pos+1, |#this|)  and
<remove> = #this[pos, pos+1)
```

Example

Code	State
	si = < 49, 3, 70 > *z = -584*
`z = si.remove(1);`	

Example

Code	State
	si = < 49, 3, 70 > *z = -584*
`z = si.remove(1);`	
	si = < 49, 70 > *z = 3*

`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 <= pos and pos < |this|`

- Ensures:

  `<entry> = this[pos, pos+1)`

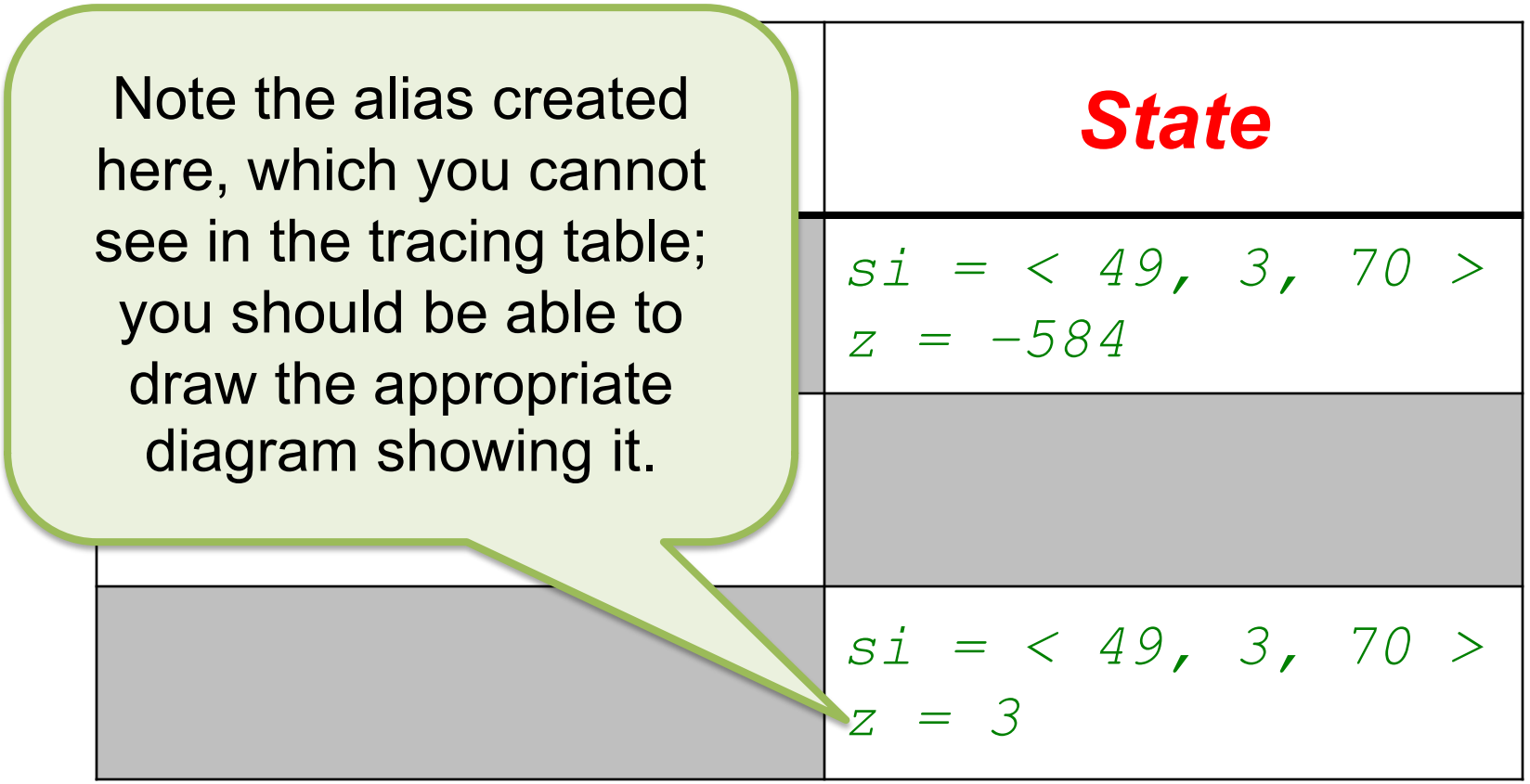
Example

Code	State
	si = < 49, 3, 70 > *z = -584*
`z = si.entry(1);`	

Example

Code	State
	si = < 49, 3, 70 > *z = -584*
`z = si.entry(1);`	
	si = < 49, 3, 70 > *z = 3*

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.
State
si = < 49, 3, 70 >
z = -584
si = < 49, 3, 70 >
z = 3

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` ***`and`*** *`pos < |this|`*
- Ensures:

  `this` *`= #this[0, pos) * <x> *`*

  `#this[pos+1, |#this|)` ***`and`***

  `<replaceEntry> = #this[pos, pos+1)`

Example

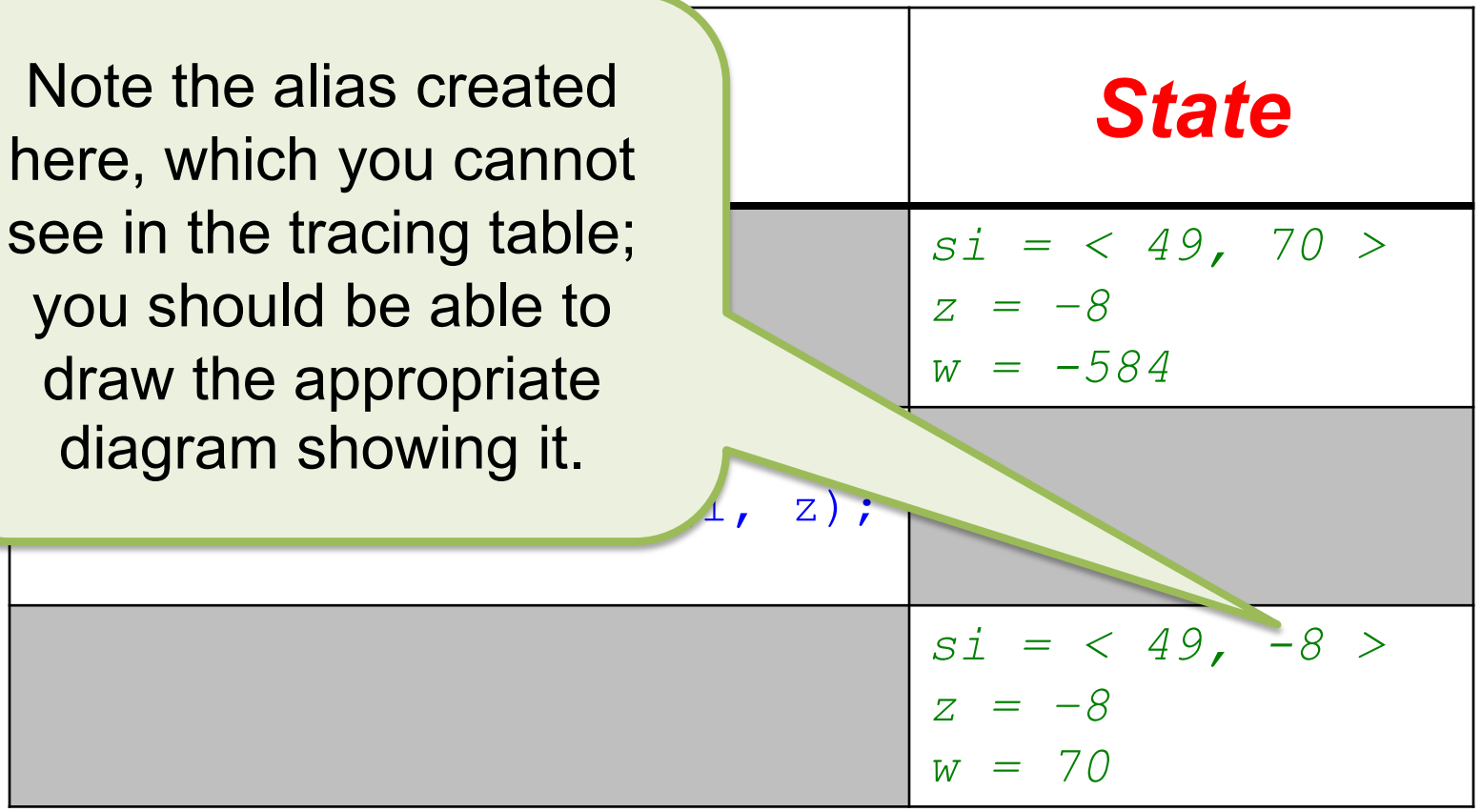
Code	State
	si = < 49, 70 > *z = -8* *w = -584*
`w = si.replaceEntry(1, z);`	

Example

Code	State
	si = < 49, 70 > *z = -8* *w = -584*
`w = si.replaceEntry(1, z);`	
	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.



Another Example

Code	State
	si = < 49, 70 > *z = -8*
`z = si.replaceEntry(1, z);`	

Another Example

Code	State
	si = < 49, 70 > *z = -8*
`z = si.replaceEntry(1, z);`	
	si = < 49, -8 > *z = 70*

Another Example

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

	State
	si = < 49, 70 > *z = -8*
`z = si.replaceEntry(1, z);`	
	si = < 49, -8 > *z = 70*

append

void append(Sequence<T> s)

- Concatenates ("appends") `s` to the end of **`this`**.
- Updates: **`this`**
- Clears: `s`
- Ensures:

  this = #***this*** * #*s*

flip

void flip()

- Reverses ("flips") **this**.
- Updates: **this**
- Ensures:

  this = ***rev***(#***this***)

insert

```
void insert(int pos, Sequence<T> s)
```

- Inserts `s` at position `pos` of **`this`**, and clears `s`.
- Updates: **`this`**
- Clears: `s`
- Requires:

  `0 <= pos  and  pos <= |this|`

- Ensures:

  `this = #this[0, pos) * #s * #this[pos, |#this|)`

Example

Code	State
	si1 = < 8, 6, 92 > *si2 = < 1, -7 >*
`si1.insert(2, si2);`	

Example

Code	State
	si1 = < 8, 6, 92 > *si2 = < 1, -7 >*
`si1.insert(2, si2);`	
	si1 = < 8, 6, 1, -7, 92 > *si2 = < >*

extract

`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 <= pos1 and pos1 <= pos2 and pos2 <= |this|`

- Ensures:

  `this = #this[0, pos1) * #this[pos2, |#this|) and`
  `s = #this[pos1, pos2)`

Example

Code	State
	si1 = < 8, 6, 92, 27, 0 > *si2 = < 1, -7, 562 >*
`si1.extract(1, 3, si2);`	

Example

Code	State
	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/