Questions

Bookmark this page

MyIterator.next signature

2/2 points (graded)

This example is one of the first we've seen that uses *instance methods*. Instance methods operate on an instance of a class, take an implicit `this` parameter (like the explicit `self` parameter in Python), and can access instance fields.

Let's examine `MyIterator`'s `next` method:

```
public class MyIterator {

    private final ArrayList<String> list;
    private int index;

    ...

    /**
     * Get the next element of the list.
     * Requires: hasNext() returns true.
     * Modifies: this iterator to advance it to the element
     *           following the returned element.
     * @return next element of the list
     */
    public String next() {
        final String element = list.get(index);
        ++index;
        return element;
    }
}
```

Thinking about `next` as an operation as defined in Static Checking: Types...

What are the types of the input(s) to next? Check all that apply.

- [ ] `void` (there are no inputs)
- [ ] `ArrayList`
- [x] `MyIterator`
- [ ] `String`
- [ ] `boolean`
- [ ] `int`

✔

What are the types of the output(s) to next? Check all that apply.

- [ ] `void` (there are no inputs)
- [ ] `ArrayList`
- [ ] `MyIterator`
- [x] `String`

- [ ] `boolean`
- [ ] `int`

✔

Submit

Show Answer

✔ Correct (2/2 points)

MyIterator.next precondition

2/2 points (graded)

`next` has the precondition `requires: hasNext() returns true`.

Which input(s) to `next` are constrained by the precondition? Check all that apply.

- [ ] *none of them*
- [x] `this`
- [ ] `hasNext`
- [ ] `elem`

✔

When the precondition isn't satisfied, the implementation is free to do anything. So:

What does this particular implementation do when the precondition is not satisfied?

- [ ] return `null`
- [ ] return some other element of the list
- [ ] throw a checked exception
- [x] throw an unchecked exception

✔

Explanation
The precondition is on the state of the `MyIterator` instance on which we're calling `next`. Consider this code:

```
MyIterator iter = /* ... create a new MyIterator ... */;
String a = iter.next();
```

For this call to `iter.next()`, the value of `this` will be `iter`, and the precondition says `iter.hasNext()` must be true. Because the code does not check `iter.hasNext()`, we risk violating the precondition.
`hasNext` is not an input; neither is `elem`.
If the precondition is not satisfied, then this implementation will call `list.get(index)` with a value of `index` that is too large, triggering an unchecked `IndexOutOfBoundsException`.

Submit

Show Answer

Answers are displayed within the problem

MyIterator.next postcondition

2/2 points (graded)

Part of the postcondition of `next` is: `@return next element of the list.`

Which output(s) from `next` are constrained by that postcondition? Check all that apply.

- [ ] *none of them*
- [ ] `this`

- [ ] `hasNext`
- [x] the return value

✔

Another part of the postcondition of `next` is `modifies: this iterator to advance it to the element following the returned element.`

What is (are) constrained by that postcondition? Check all that apply.

- [ ] *nothing*
- [x] `this`
- [ ] `hasNext`
- [ ] the return value

✔

Explanation
The first part of the postcondition is on the `String` returned from `next`. The second part of the postcondition is on the state of the `MyIterator` instance after we call `next`.

Submit

Show Answer

Answers are displayed within the problem

Previous Next