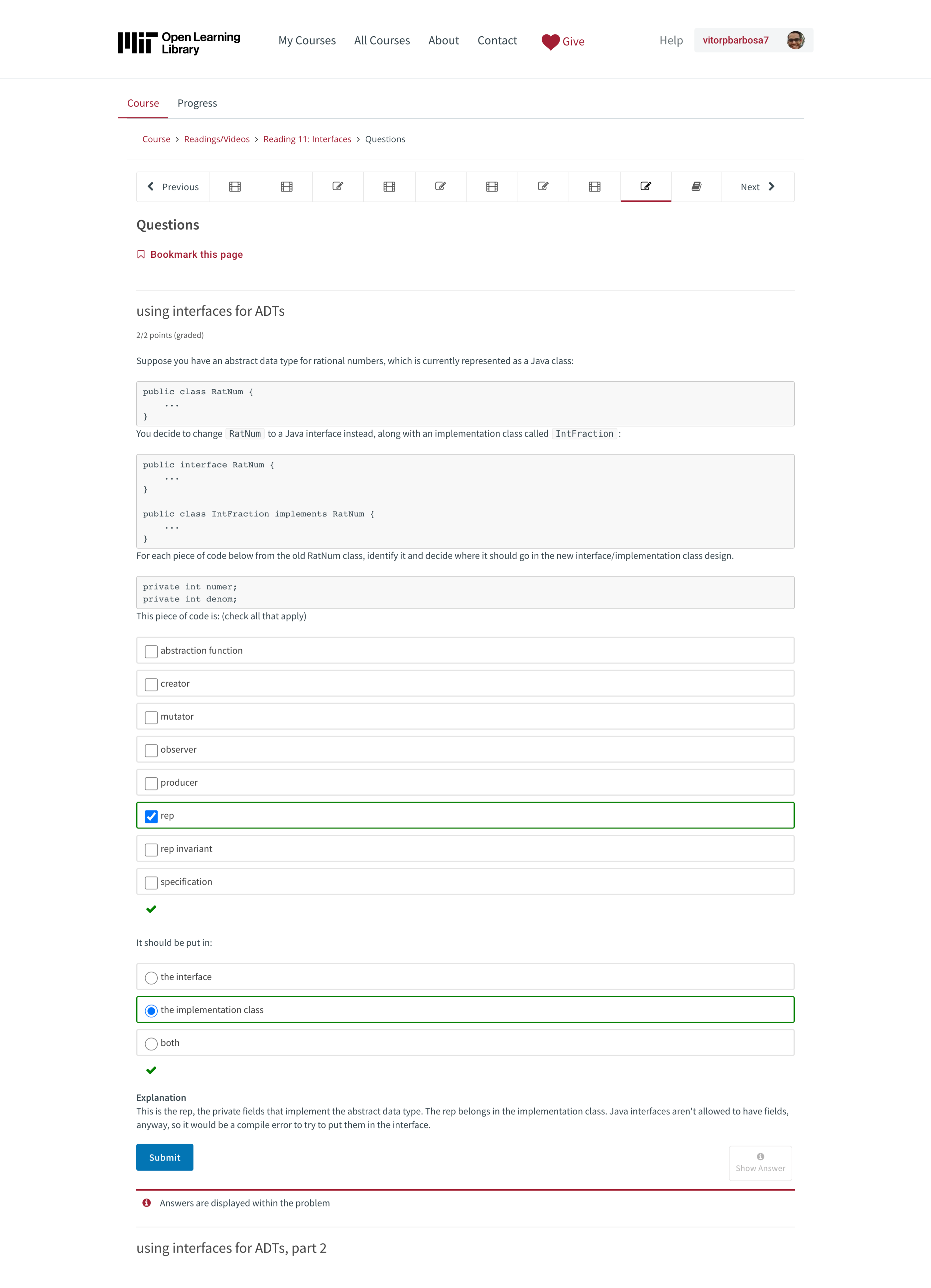


Questions

Bookmark this page

using interfaces for ADTs

2/2 points (graded)

Suppose you have an abstract data type for rational numbers, which is currently represented as a Java class:

```
public class RatNum {
    ...
}
```

You decide to change `RatNum` to a Java interface instead, along with an implementation class called `IntFraction`:

```
public interface RatNum {
    ...
}

public class IntFraction implements RatNum {
    ...
}
```

For each piece of code below from the old RatNum class, identify it and decide where it should go in the new interface/implementation class design.

```
private int numer;
private int denom;
```

This piece of code is: (check all that apply)

□ abstraction function

□ creator

□ mutator

□ observer

□ producer

☑ rep

□ rep invariant

□ specification

✓

It should be put in:

○ the interface

● the implementation class

○ both

✓

Explanation

This is the rep, the private fields that implement the abstract data type. The rep belongs in the implementation class. Java interfaces aren't allowed to have fields, anyway, so it would be a compile error to try to put them in the interface.

Submit

Show Answer

Answers are displayed within the problem

using interfaces for ADTs, part 2

2/2 points (graded)

```
// denom > 0
//   numer/denom is in reduced form
```

This piece of code is: (check all that apply)

□ abstraction function

□ creator

□ mutator

□ observer

□ producer

□ rep

☑ rep invariant

□ specification

✓

It should be put in:

○ the interface

● the implementation class

○ both

✓

Explanation

This is the rep invariant, because it describes relationships that must be true of the rep fields. It belongs only in the implementation class, because the rep only appears in the implementation class. The interface should have no knowledge of the rep.

Submit

Show Answer

Answers are displayed within the problem

using interfaces for ADTs, part 3

2/2 points (graded)

```
// represents the rational number numer / denom
```

This piece of code is: (check all that apply)

☑ abstraction function

□ creator

□ mutator

□ observer

□ producer

□ rep

□ rep invariant

□ specification

✓

It should be put in:

○ the interface

● the implementation class

○ both

✓

Explanation

This is the abstraction function, which explains how the rep is interpreted as a rational number. It belongs in the implementation class, again because only the implementation class knows what the rep is.

Submit

Show Answer

Answers are displayed within the problem

using interfaces for ADTs, part 4

1/2 points (graded)

```
    /**
     * @param  that  another RatNum
     * @return a RatNum equal to (this / that)
     */
```

This piece of code is: (check all that apply)

□ abstraction function

□ creator

□ mutator

□ observer

☑ producer ✓

□ rep

□ rep invariant

☑ specification ✓

It should be put in:

● the interface

○ the implementation class

○ both

✓

Explanation

This is part of a specification of an operation. The operation must be a producer, because it returns a RatNum. This specification needs to appear only in the interface. The implementation class should inherit it from the interface. The same comment should *not* be written in both interface and implementation, because that wouldn't be DRY. Instead, the implementation class can simply put this Javadoc comment above its `divide()` method:

```
/** @see RatNum.divide() */
```

Submit

Show Answer

Answers are displayed within the problem

using interfaces for ADTs, part 5

2/2 points (graded)

```
    public boolean isZero()
```

This piece of code is: (check all that apply)

□ abstraction function

□ creator

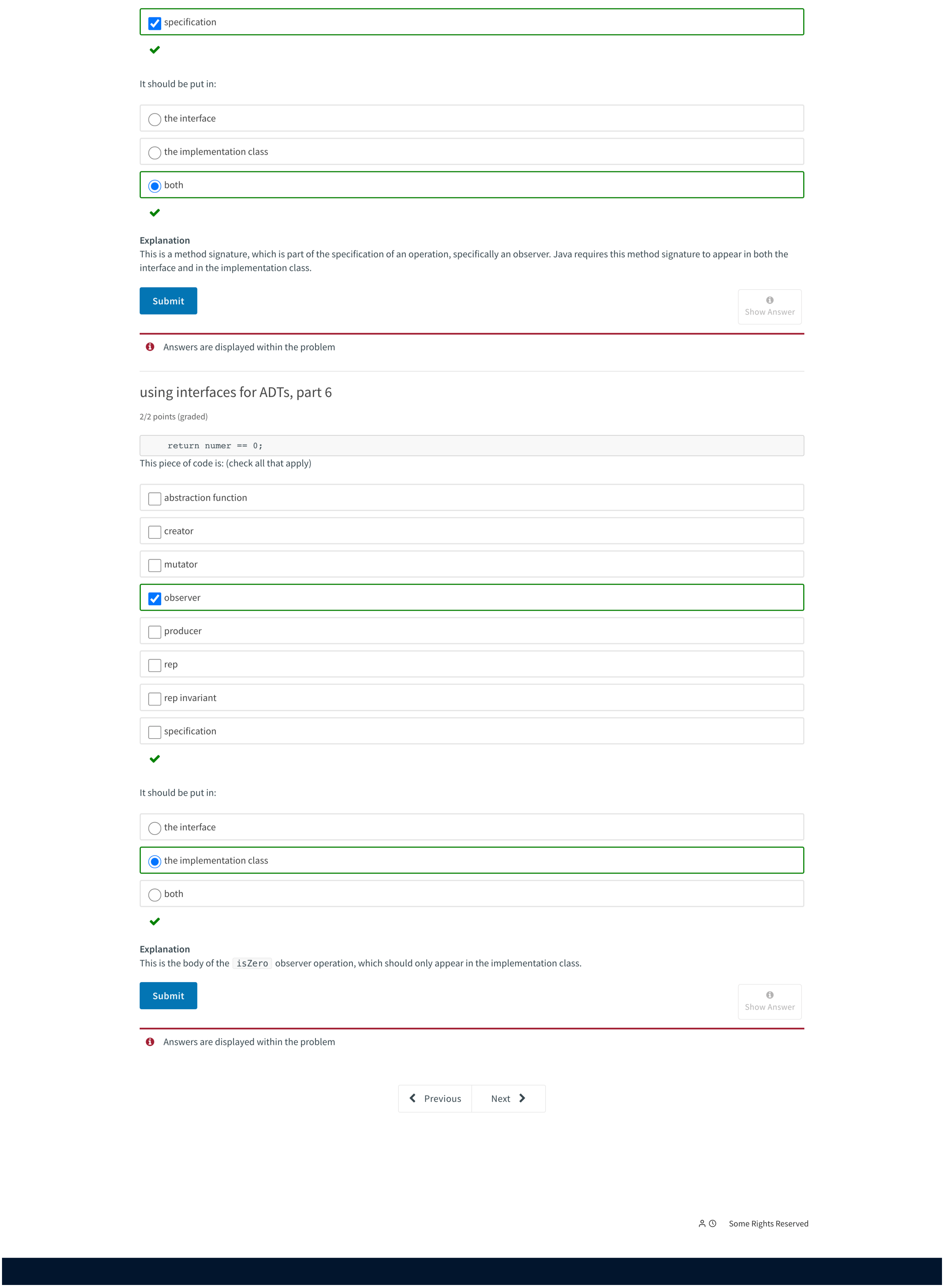
□ mutator

☑ observer

□ producer

□ rep

□ rep invariant



☑ specification

✓

It should be put in:

○ the interface

○ the implementation class

● both

✓

Explanation
This is a method signature, which is part of the specification of an operation, specifically an observer. Java requires this method signature to appear in both the interface and in the implementation class.

Answers are displayed within the problem

using interfaces for ADTs, part 6

2/2 points (graded)

```
    return numer == 0;
```

This piece of code is: (check all that apply)

□ abstraction function

□ creator

□ mutator

☑ observer

□ producer

□ rep

□ rep invariant

□ specification

✓

It should be put in:

○ the interface

● the implementation class

○ both

✓

Explanation
This is the body of the `isZero` observer operation, which should only appear in the implementation class.

Answers are displayed within the problem

Open Learning Library	Connect
About	Contact
Accessibility	Twitter
All Courses	Facebook
Why Support MIT Open Learning?	
Help	

Privacy Policy Terms of Service