

My Courses All Courses About Contact



Help vitorpbarbosa7



Course Progress

Course > Readings/Videos > Reading 9: Abstract Data Types > Questions



Questions

□ Bookmark this page

testing an ADT

1/4 points (graded)

Consider the following datatype:

```
/** Immutable datatype representing a student's progress through school. */
 class Student {
     /** make a freshman */
     public Student() { ... }
     /** @return a student promoted to the next year, i.e.
freshman returns a sophomore,
              sophomore returns a junior,
              junior returns a senior,
              senior returns an alum,
              alum stays an alum and can't be promoted further. */
     public Student promote() { ... }
/** @return number of years of school completed, i.e.
              0 for a freshman, 4 for an alum */
     public int getYears() { ... }
How many parts are there in a reasonable input-space partition of the Student() constructor?
```

✓ Answer: 1

Explanation

Student() takes no parameters, so its input space is the empty set. There is only one way to partition the empty set -- as a partition whose only part is the empty set. So the partition has one part.

How many parts are there in a reasonable, but not exhaustive, input-space partition of promote()?

3 Answer: 3

promote() has only one input: the Student object it is called on. A reasonable but not exhaustive partition would include each of the boundary values (freshman and alum) as a separate part, and all the intermediate values (sophomore, junior, senior) together as a part. \\

How many parts are there in a reasonable, but not exhaustive, input-space partition of getYears()?

Explanation

getYears() has only one input: the Student object it is called on. A reasonable but not exhaustive partition would include each of the boundary values (freshman and alum) as a separate part and all the intermediate values (sophomore, junior, senior) together as a part.

Each of the following method call sequences is a single test case. (Note that we omit the detail about which object each method is called on.) Which of these test cases would cover all the partitions above? Check all that apply.

promote(), promote(), promote()
Student(), promote(), promote(), promote(), getYears()
Student(), promote(), getYears(), promote(), getYears(), promote(), getYears(), promote(), getYears()
Student(), getYears(), promote(), getYears(), promote(), promote(), promote(), getYears()
Student(), getYears(), promote(), getYears(), promote(), getYears(), promote(), getYears(), promote(), getYears()

Explanation

The first test case only calls promote(), which doesn't cover the other operations.

The second test case only calls getYears() once, presumably on the final (alum) value, so it has no way to check whether the intermediate values produced by Student() and promote() were correct.

The third test case doesn't cover all the parts of getYears() -- it never calls getYears() on a freshman, i.e. the result of Student().

Page 2
Questions | Reading 9: Abstract Data Types | 6.005.1x Courseware | MIT Open Learning Library
https://openlearninglibrary.mit.edu/courses/course-v1:MITx+6.005.1x+3T2016/courseware/Readings_Videos/09-Abstract-Data-Types/?activate_block_id=block-v1%3AMITx%2B6.005.1x%2B3T2016%2Btype%40sequ...

The fourth test case builds a sequence of Student objects, stepping from freshman to alum and also calls promote() again on the alum to make sure it doesn't change. As a result, Student() is called and checked in the one input it can be (i.e. no arguments), and promote() and getYears() are each called in the three reasonable parts (freshman, sophomore/junior/senior, and alum).

The fifth test case is even more exhaustive than the fourth, so it also covers all the parts.

Show Answer

Previous Next >

△ ③ Some Rights Reserved

Open Learning Library

About Accessibility All Courses

Facebook

Connect

Contact

Twitter

Why Support MIT Open Learning?

Help