

/ 1

My Courses All Courses About Contact



Help vitorpbarbosa7



Course Progress Course > Readings/Videos > Reading 7: Mutability and Immutability > Questions Ø. Questions □ Bookmark this page secret agent 1/1 point (graded) Suppose we have a List $that's \ used \ in \ a \ number \ of \ places \ in \ our \ code. \ that's \ used \ in \ a \ number \ of \ places \ in \ our \ code. \ that's \ used \ in \ a \ number \ of \ places \ in \ our \ code. \ that's \ used \ in \ a \ number \ of \ places \ in \ our \ code.$ Which of these specifications would be dangerous to use because of mutation? Check all that apply. public static List<String> sort(List< String> list) Requires: nothing Effects: modifies list so its elements are sorted lexicographically, and returns list public static List<String> sort(List< String> list) Requires: nothing Effects: returns a copy of list with the elements sorted lexicographically public List<String> sort(List< String> list) Effects: returns a new List with the elements from list, sorted lexicographically **√** 4. public List<String> sort(List< String> list) Effects: returns a new List by removing all elements from list and sorting them lexicographically public void sort(List< String> list) Requires: nothing Effects: modifies list so its elements are sorted lexicographically None of them Submit Show Answer ✓ Correct (1/1 point) agent of change Which of those specifications cause aliasing when a client calls the method? Check all that apply.

Page 2
Questions | Reading 7: Mutability and Immutability | 6.005.1x Courseware | MIT Open Learning Library
https://openlearninglibrary.mit.edu/courses/course-v1:MITx+6.005.1x+3T2016/courseware/Readings_Videos/07-Mutability-Immutability/?activate_block_id=block_v1%3AMITx%2B6.005.1x%2B3T2016%2Btype%40s..

2	
₹ 3	
₹ 4	
▼ 5	
None of them	
Explanation Calling every single one of these methods creates an alias for as long as the function is running. The caller must have a reference to the list argument and now the local variable in the method does as well. That's two references to the same	object.
Submit	1 Show Answer

Previous

Next >

△ ○ Some Rights Reserved

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