SalesItem Class

---

getName(): The getName method will be tested by creating a new SalesItem object and confirming the method returns the name we created the object with.

getPrice(): The getPrice method will be tested by creating a new SalesItem object and confirming the method returns the price we created the object with.

getNumberOfComments(): Test the getNumberOfComments method by calling the getNumberOfComments method on a SalesItem with no comments yet added which should return 0. Test the method again but this time with a comment added and it should return 1.

addComment(): Test the addComment method by adding a valid comment with an author name that has not been taken before and a valid rating, and this should return true. Continue testing the method with various cases of comments with the same author and invalid ratings which are outside the [1..5] list, all of which should return false.

removeComment(): Begin the testing by adding 3 valid comments to the SalesItem, then continue by removing the first comment, proceed by calling the getNumberOfComments method to check the number of comments on the SalesItem which should return 2 at this point because 1 was removed.

Test the method furthermore by repeating the process and removing another comment which leaves the numberOfComments at 1. Lastly test the boundaries by removing 2 "non-existent" comments by removing 2 comments with the index -1 and 6, then call getNumberOfComments after each which should return 1 because the removeComment method should do nothing.

ratingInvalid(): The ratingInvalid method is a private method so create a small public method called testRatingInvalid in the SalesItem class that tests the ratingInvalid with various test cases such as all of the valid ratings which are [1..5] inclusively and test the boundaries by testing with 2 invalid ratings 0 and 6. If the check fails it will print a statement saying the test case failed and return false.

upvoteComment(): Test the upvoteComment method by first adding a comment to the object of SalesItem created in the setUp method in the unit test. Secondly, create an accessor method called “getVotesForIndex”, that returns the number of votes for the comment which index is passed as parameter. Then test that the votes are equal to zero, since it was just created. Use the upvoteComment method to upvote the comment just created. Lastly, test that the votes for that individual comment has gone up by a factor of one.

downvoteComment(): Test the downvoteComment method by first adding a comment to the object of SalesItem created in the setUp method in the unit test. Secondly, use the accessor method called “getVotesForIndex”, that returns the number of votes for the comment which index is passed as parameter. Then test that the votes are equal to zero, since it was just created. Use the

downvoteComment method to downvote the comment just created. Lastly, test that the votes for that individual comment has gone down by a factor of one.

findMostHelpfulComment():Test the findMostHelpfulComment method by first adding two comments to the object of SalesItem created in the setUp method in the unit test. Secondly, use the upvoteComment method to upvote the first comment in the list. Use the downvoteComment method to downvote the second comment in the list. Then use the findMostHelpfulComment to test that the comment with more votes is the first comment in the ArrayList.

findCommentByAuthor(): Create an accessor method that returns the comment by the author passed as parameter and returns null if it does not exist. Then add a comment to the object of SalesItem created in the setUp method in the unit test. Use the accessor method to test that the author of the comment just created is recognized as the author of the first comment of the array.

SalesItem () [CONSTRUCTOR]: Test the constructor by creating a new object of the class SalesItem. Using the getName() method, test that the name equals the name inputted when creating the object. Using the getPrice() method, test that the price equals the price inputted when creating the object. Using the getNumberOfComments () method, test that the number of comments is equal to zero since it was just created.

Comment Class

Comment(String author, String text, int rating): The Comment constructor will be tested by creating a new Comment object and calling its getter methods to confirm the object's fields have the correct values.

upvote(): The upvote method will be tested by creating a new Comment object and calling the upvote method. Then the getVoteCount method will be called to confirm that the vote count changed accordingly.

downvote(): The downvote method will be tested by creating a new Comment object and calling the downvote method. Then the getVoteCount method will be called to confirm that the vote count changed accordingly.

getRating():The getRating method will be tested by creating a new Comment object with a rating of 3 and then calling the getRating method to verify it returns 3.