|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case#** | **Purpose of the Test Case** | **Input Data** | **Expected Output** |
| 1 | Test the add method, which adds an item to the bag. An item is added if it is in the bag array of shopping bag.  • Case 1: Add one item called apple, the method should have the apple in the bag and the count be 1. Tests if add puts an item in the bag and increases the number of items present by 1.  • Case 2: Add one item called skirt, the method should have the skirt in the bag and the count be 2. Tests if add puts a distinct item in the bag and increases the number of items present by 1.  • Case 3: Add one item called apple, the method should have the apple in the bag and the count be 3. Tests if add puts a similar item in the bag and increases the number of items present by 1. | • Case 1: apple  • Case 2: skirt  • Case 3: apple | • Case 1:  Bag: {apple}  Number of items: 1  • Case 2:  Bag: {apple, skirt}  Number of items: 2  • Case 3:  Bag: {apple, skirt, apple}  Number of items: 3 |
| 2 | Test the remove method, which removes an item from the bag. An item is removed if it is in the bag array of shopping bag.  • Case 1: Remove one item called apple, the method should return true, remove an apple in the bag, and the count be 2. Tests if remove method removes an item in the bag and decreases the number of items present by 1.  • Case 2: Remove one item called puppy, the method should return false, not remove any items in the bag, and keep the count at 2. Tests if remove method returns false when tries to remove an item not present in bag. Also keeps the number of items the same.  • Case 3: Removes two items called apple, the method should return true for both instances, remove two apples in the bag, and the count be 0. Tests if remove method removes an item even though the bag will be empty and decreases the number of items to 0. | • Case 1: skirt  • Case 2: puppy  • Case 3: apple, apple | • Case 1:  Return: True  Bag: {apple, apple}  Number of items: 2  • Case 2:  Return: False  Bag: {apple, apple}  Number of items: 2  • Case 3:  Instance 1:  Return: True  Bag: {apple}  Number of items: 1  Instance2  Return: True  Bag: {}  Number of items: 0 |
| 3 | Test the grow method, which grows the bag by 5 when the bag reaches its capacity.  • Case 1: add 5 apples, the method should have 5 apples in the bag, have a count of 5, and an array length of 5. Tests if grow method does not grow when it reaches 5 which is its max capacity.  • Case 2: adds 1 pencil, the bag should have 5 apples and 1 pencil in the bag, have a count of 6, and an array length of 10. Tests if grow method makes array length increase by 5 and if it adds the item in the bag and if it counts it as an item.  • Case 3: adds 5 apples, the bag should have 10 apples and 1 pencil in the bag, have a count of 11, and an array length of 15. Tests if grow method makes array length increase of 5 and doesn’t just double the size of the array. It also makes sure to check if it adds the items in the bag and if it counts all the items. | • Case 1: add 5 apples  • Case 2: add 2 pencil  • Case 3: add 5 apples | • Case 1:  Bag: {apple, apple, apple, apple, apple}  Number of items: 5  Array length: 5  • Case 2:  Bag: {apple, apple, apple, apple, apple, skirt}  Number of items: 6  Array length: 10  • Case 3:  Bag : {apple, apple, apple, apple, apple, pencil, apple, apple, apple, apple, apple}  Number of items: 11  Array length: 15 |
| 4 | Tests the saleTax method which should add up all the sale tax of all taxable items.  • Case 1: add 10 apples and 1 pencil which should add up all the sale tax of each item since they are all taxable. Tests to see if sale tax is added up properly.  • Case 2: adds a skirt which should not add any sale tax to the total of the sale tax since it is not taxable. Tests the method to make sure it does not add up nontaxable items. | • Case 1: 10 apples and 1 pencil  • Case 2: 10 apples, 1 skirt, and 1 pencil | • Case 1:  Sales tax: $1.47  • Case 2:  Sales tax: $1.47 |