KEY: Orange Highlight = Test added during this milestone
Red Highlight = Impossible Test Case

<table>
<tr><th>Test</th><th>Equivalence class partitioning(Wect, Sect, Wrect, Srect)</th></tr>
<tr><td>PredicatedListTest
.testDecorated</td><td>

Variables:
List length L
Equivalence classes
L1: L = 0
L2: L > 0
Thus, wect and sect are both covered by using an empty list and a nonempty
This is already done
Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.testEquals</td><td>

Variables:
List length L
Equivalence classes
L1: L = 0
L2: L > 0
Thus, wect and sect are both covered by using an empty list and a nonempty
This is already done
Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.testHashCode</td><td>

Variables:
List length L
Equivalence classes
L1: L = 0
L2: L > 0
Thus, wect and sect are both covered by using an empty list and a nonempty
This is already done
Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.testGet</td><td>

Variables:
List length L
Index I
Equivalence classes
L1: L = 0
L2: L > 0
I1: I < 0
I2: I >= 0 && I < L
I3: I >= L
Wect
<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
</table>

</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>WECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>I2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>I3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>I1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>I2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>I2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>I3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>I3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT5, and SECT6
Wrect and srect are also covered by testing of a null variable list

</td>
</tr>
<tr>
<td>PredicatedListTest
.testIndexOf</td>
<td>

Variables:

List length L
Object O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O is in list
O2: O is not in list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, SECT1
Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.tesetLastIndexOf</td><td>

Variables:

List length L
Object O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O is only instance of O in list
O2: multiple instances of O in list
O3: O is not in list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>O3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>O3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>O3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, WECT3, SECT1, SECT3
Wrect and srect are also covered by testing of a null variable list

</td></tr>
</table>

<table>
<tr>
<td>PredicatedListTest
.testRemove</td>
<td>

Variables:

List length L
Index I

Equivalence classes

L1: L = 0
L2: L > 0
I1: I < 0
I2: I >= 0 && I < L
I3: I >= L

Wect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>WECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>I2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>I3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>I1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>I2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>I2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>I3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>I3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT5, and SECT6
Wrect and srect are also covered by testing of a null variable list

</td>
</tr>
<tr>
<td>PredicatedListTest
.testAdd</td>
<td>

Variables:

List length L
Object O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O can be inserted in list
O2: O cannot be inserted in list

Wect

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td>
</tr>
<tr>
<td>PredicatedListTest
.testAddAll</td>
<td>

Variables:

List length L
List of objects O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O can be fully inserted into list
O2: O can be partially inserted into list
O3: O cannot be inserted at all into list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>O3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>O3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>O3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.testListIterator</td><td>

Variables:

List length L

Equivalence classes

L1: L = 0
L2: L > 0

Thus, wect and sect are both covered by using an empty list and a nonempty

This is already done

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>PredicatedListTest
.testSet</td><td>

Variables:

List length L
Index I
Object O

Equivalence classes

L1: L = 0
L2: L > 0
I1: I < 0
I2: I >= 0 && I < L
I3: I >= L
O1: O can be inserted
O2: O cannot be inserted

Wect

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>I1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>I2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>I3</td><td>O1</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>SECT1</td><td>L1</td><td>I1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>I1</td><td>O1</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>SECT3</td><td>L1</td><td>I2</td><td>O1</td></tr>
<tr><td>SECT4</td><td>L2</td><td>I2</td><td>O1</td></tr>
<tr><td>SECT5</td><td>L1</td><td>I3</td><td>O1</td></tr>
<tr><td>SECT6</td><td>L2</td><td>I3</td><td>O1</td></tr>
<tr><td>SECT7</td><td>L1</td><td>I1</td><td>O2</td></tr>
<tr><td>SECT8</td><td>L2</td><td>I1</td><td>O2</td></tr>
<tr><td>SECT9</td><td>L1</td><td>I2</td><td>O2</td></tr>
<tr><td>SECT10</td><td>L2</td><td>I2</td><td>O2</td></tr>
<tr><td>SECT11</td><td>L1</td><td>I3</td><td>O2</td></tr>
<tr><td>SECT12</td><td>L2</td><td>I3</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT3, SECT5, SECT6, SECT7, SECT9, SECT10, SECT11, SECT12
Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testAsSet</td><td>

Variables:

List length L

Equivalence classes

L1: L = 0
L2: L > 0

Thus, wect and sect are both covered by using an empty list and a nonempty

This is already done

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testAdd</td><td>

Variables:

List length L
Object O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O does not already exist in list
O2: O already exists in list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testAddAll</td>
<td>

Variables:

List length L
List of objects O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O can be fully inserted into list without duplicates
O2: O can be partially inserted into list without duplicates
O3: O cannot be inserted at all into list without duplicate

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>O3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>SECT5</td><td>L1</td><td>O3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>O3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testSet</td><td>

Variables:

List length L
Index I
Object O

Equivalence classes

L1: L = 0
L2: L > 0
I1: I < 0
I2: I >= 0 && I < L
I3: I >= L
O1: O is not a duplicate
O2: O is a duplicate

Wect

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>I1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>I2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>I3</td><td>O1</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>SECT1</td><td>L1</td><td>I1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>I1</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>I2</td><td>O1</td></tr>
<tr><td>SECT4</td><td>L2</td><td>I2</td><td>O1</td></tr>
<tr><td>SECT5</td><td>L1</td><td>I3</td><td>O1</td></tr>
<tr><td>SECT6</td><td>L2</td><td>I3</td><td>O1</td></tr>
<tr><td>SECT7</td><td>L1</td><td>I1</td><td>O2</td></tr>
<tr><td>SECT8</td><td>L2</td><td>I1</td><td>O2</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>SECT9</td><td>L1</td><td>I2</td><td>O2</td></tr>
<tr><td>SECT10</td><td>L2</td><td>I2</td><td>O2</td></tr>
<tr><td>SECT11</td><td>L1</td><td>I3</td><td>O2</td></tr>
<tr><td>SECT12</td><td>L2</td><td>I3</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testRemove</td>
<td>

Variables:

List length L
Index I

Equivalence classes

L1: L = 0
L2: L > 0
I1: I < 0
I2: I >= 0 && I < L
I3: I >= L

Wect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>WECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>I2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>I3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>I1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>I1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>I2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>I2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>I3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>I3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, WECT3, SECT1, SECT2, SECT5, and SECT6

</td>
</tr>
</table>

<table>
<tr><td></td><td>

- Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testRemoveIf</td><td></td></tr>
<tr><td>SetUniqueListTest
.testRemoveAll</td><td>

- Variables:
  - List length L
  - List of objects O
- Equivalence classes
  - L1: L = 0
  - L2: L > 0
  - O1: O can be fully removed from list
  - O2: O can be partially removed from list
  - O3: No elements of O exist in list
- Wect
  -

ID	L	O
WECT1	L1	O1
WECT2	L2	O2
WECT3	L1	O3

- Sect
  -

ID	L	I
SECT1	L1	O1
SECT2	L2	O1
SECT3	L1	O2
SECT4	L2	O2
SECT5	L1	O3
SECT6	L2	O3

  -
- Wrect and Srect would involve L < 0 but this is not possible
- Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testRetainAll</td><td>

- Variables:
  - List length L
  - List of objects O
- Equivalence classes
  - L1: L = 0
  - L2: L > 0
  - O1: All elements of O exist in list

</td></tr>
</table>

<table>
<tr>
<td></td>
<td>

O2: Some elements of O exist in list
O3: No elements of O exist in list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>O3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
<tr><td>SECT5</td><td>L1</td><td>O3</td></tr>
<tr><td>SECT6</td><td>L2</td><td>O3</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are also covered by testing of a null variable list

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testClear</td>
<td>

Variables:

List length L

Equivalence classes

L1: L = 0
L2: L > 0

Thus, wect and sect are both covered by using an empty list and a nonempty

This is already done

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testContains</td>
<td>

Variables:

List length L
Object O

Equivalence classes

L1: L = 0
L2: L > 0
O1: O is in list
O2: O is not in list

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by WECT1, SECT1
Wrect and srect are also covered by testing of a null variable list

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testContainsAll</td>
<td>

Variables:

List length L
List of objects O

Equivalence classes

L1: L = 0
L2: L > 0
O1: All elements of O exist in list
O2: Not all elements of O exist in list

Wect

<table>
<tr><td>ID</td><td>L</td><td>O</td></tr>
<tr><td>WECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>WECT2</td><td>L2</td><td>O2</td></tr>
<tr><td>WECT3</td><td>L1</td><td>O3</td></tr>
</table>

Sect

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>SECT1</td><td>L1</td><td>O1</td></tr>
<tr><td>SECT2</td><td>L2</td><td>O1</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>SECT3</td><td>L1</td><td>O2</td></tr>
<tr><td>SECT4</td><td>L2</td><td>O2</td></tr>
</table>

○

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are also covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testIterator</td><td>

Variables:

List length L

Equivalence classes

L1: L = 0
L2: L > 0

Thus, wect and sect are both covered by using an empty list and a nonempty

This is already done

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
<tr><td>SetUniqueListTest
.testListIterator</td><td>

Variables:

List length L

Equivalence classes

L1: L = 0
L2: L > 0

Thus, wect and sect are both covered by using an empty list and a nonempty

This is already done

Wrect and Srect would involve L < 0 but this is not possible
Wrect and srect are covered by testing of a null variable list

</td></tr>
</table>

<table>
<tr><td>Test</td><td>Boundary Value Analysis</td></tr>
<tr><td>PredicatedListTest
.testGet</td><td>

Variables:

List length L
Index I

Test Set

BVL: {nom: L = 0, nom: L > 0}
BVI: {min-: I = -1, min: I = 0, min+: I = 1, nom: 1 < I < L, max-: I = L - 2, max: I = L - 1, max+: I = L}

BVA (basic)

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>BVA (basic) 1</td><td>L > 0</td><td>0</td></tr>
<tr><td>BVA (basic) 2</td><td>L > 0</td><td>1</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>BVA (basic) 3</td><td>L > 0</td><td>nom</td></tr>
<tr><td>BVA (basic) 4</td><td>L > 0</td><td>L - 2</td></tr>
<tr><td>BVA (basic) 5</td><td>L > 0</td><td>L - 1</td></tr>
</table>

BVA (robust)

<table>
<tr><td>ID</td><td>L</td><td>I</td></tr>
<tr><td>BVA (robust) 1</td><td>L > 0</td><td>-1</td></tr>
<tr><td>BVA (robust) 2</td><td>L > 0</td><td>0</td></tr>
<tr><td>BVA (robust) 3</td><td>L > 0</td><td>1</td></tr>
<tr><td>BVA (robust) 4</td><td>L > 0</td><td>nom</td></tr>
<tr><td>BVA (robust) 5</td><td>L > 0</td><td>L - 2</td></tr>
<tr><td>BVA (robust) 6</td><td>L > 0</td><td>L - 1</td></tr>
<tr><td>BVA (robust) 7</td><td>L > 0</td><td>L</td></tr>
</table>

</td></tr>
<tr><td>PredicatedListTest
.testSet</td><td>

Variables:

List length L
Index I
Object O

Test Set

BVL: {nom: L = 0, nom: L > 0}
BVI: {min-: I = -1, min: I = 0, min+: I = 1, nom: 1 < I < L, max-: I = L - 2, max: I = L - 1, max+: I = L}
BVO: {nom: O can be inserted, nom: O cannot be inserted}

BVA (basic)

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>BVA (basic) 1</td><td>L > 0</td><td>0</td><td>O can be inserted</td></tr>
<tr><td>BVA (basic) 2</td><td>L > 0</td><td>1</td><td>O can be inserted</td></tr>
<tr><td>BVA (basic) 3</td><td>L > 0</td><td>nom</td><td>O can be inserted</td></tr>
</table>

</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>BVA (basic) 4</td><td>L > 0</td><td>L - 2</td><td>O can be inserted</td></tr>
<tr><td>BVA (basic) 5</td><td>L > 0</td><td>L - 1</td><td>O can be inserted</td></tr>
</table>

BVA (robust)

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>BVA (robust) 1</td><td>L > 0</td><td>-1</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 2</td><td>L > 0</td><td>0</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 3</td><td>L > 0</td><td>1</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 4</td><td>L > 0</td><td>nom</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 5</td><td>L > 0</td><td>L - 2</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 6</td><td>L > 0</td><td>L - 1</td><td>O can be inserted</td></tr>
<tr><td>BVA (robust) 7</td><td>L > 0</td><td>L</td><td>O can be inserted</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest .testSet</td>
<td>

Variables:

List length L
Index I
Object O

Test Set

BVL: {nom: L = 0, nom: L > 0}
BVI: {min-: I = -1, min: I = 0, min+: I = 1, nom: 1 < I < L, max-: I = L - 2, max: I = L - 1, max+: I = L}
BVO: {nom: O is unique, nom: O is not unique}

BVA (basic)

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>BVA (basic) 1</td><td>L > 0</td><td>0</td><td>O is unique</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>BVA (basic) 2</td><td>L > 0</td><td>1</td><td>O is unique</td></tr>
<tr><td>BVA (basic) 3</td><td>L > 0</td><td>nom</td><td>O is unique</td></tr>
<tr><td>BVA (basic) 4</td><td>L > 0</td><td>L - 2</td><td>O is unique</td></tr>
<tr><td>BVA (basic) 5</td><td>L > 0</td><td>L - 1</td><td>O is unique</td></tr>
</table>

BVA (robust)

<table>
<tr><td>ID</td><td>L</td><td>I</td><td>O</td></tr>
<tr><td>BVA (robust) 1</td><td>L > 0</td><td>-1</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 2</td><td>L > 0</td><td>0</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 3</td><td>L > 0</td><td>1</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 4</td><td>L > 0</td><td>nom</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 5</td><td>L > 0</td><td>L - 2</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 6</td><td>L > 0</td><td>L - 1</td><td>O is unique</td></tr>
<tr><td>BVA (robust) 7</td><td>L > 0</td><td>L</td><td>O is unique</td></tr>
</table>

</td>
</tr>
</table>

Test	Decision Table
PredicatedListTest .testEquals	● Conditions ○ C1: L <= 0? ○ C2: Objects are equal? ● Actions ○ A1: Objects are found to be equal ● Table ○

<table>
<tr><td></td><td>
<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>True</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>False</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest .testGet</td><td>

Conditions

C1: L <= 0?
C2: Index in range?

Actions

A1: Object is gotten

Table

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object at index</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Error</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest .testIndexOf</td><td>

Conditions

C1: L <= 0?
C2: Object in list?

Actions

A1: Index is gotten

Table

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Index of object</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>-1</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest .tesetLastIndexOf</td><td>

Conditions

C1: L <= 0?

</td></tr>
</table>

<table>
<tr>
<td></td>
<td>

C2: Object in list?

Actions

A1: Index is gotten

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Index of object</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>-1</td></tr>
</table>

</td>
</tr>
<tr>
<td>PredicatedListTest .testRemove</td>
<td>

Conditions

C1: L <= 0?
C2: Object in list?

Actions

A1: Object removed

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object removed</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>

</td>
</tr>
<tr>
<td>PredicatedListTest .testAdd</td>
<td>

Conditions

C1: L <= 0?
C2: Object can be inserted?

Actions

A1: Object is inserted

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td></td><td>inserted</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>
○
</td></tr>
<tr><td>PredicatedListTest
.testAddAll</td><td>
● Conditions
○ C1: L <= 0?
○ C2: Objects can be inserted?
● Actions
○ A1: Objects are inserted
● Table
○
<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Objects inserted</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>
○
</td></tr>
<tr><td>PredicatedListTest
.testSet</td><td>
● Conditions
○ C1: L <= 0?
○ C2: Index in range?
○ C3: Object can be inserted
● Actions
○ A1: Object at index is set
● Table
○
<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>C3</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Y/T</td><td>Object at index is set</td></tr>
<tr><td>TC3</td><td>N/F</td><td>Y/T</td><td>N/F</td><td>Error</td></tr>
<tr><td>TC4</td><td>N/F</td><td>N/F</td><td>Y/T</td><td>Error</td></tr>
<tr><td>TC5</td><td>N/F</td><td>N/F</td><td>N/F</td><td>Error</td></tr>
</table>
○
</td></tr>
</table>

<table>
<tr>
<td>PredicatedListTest
.testSublist</td>
<td>

Conditions

C1: L <= 0?
C2: Start index in range?
C3: End index in range?

Actions

A1: Sublist is made

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>C3</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Y/T</td><td>Sublist</td></tr>
<tr><td>TC3</td><td>N/F</td><td>Y/T</td><td>N/F</td><td>Error</td></tr>
<tr><td>TC4</td><td>N/F</td><td>N/F</td><td>Y/T</td><td>Error</td></tr>
<tr><td>TC5</td><td>N/F</td><td>N/F</td><td>N/F</td><td>Error</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testAdd</td>
<td>

Conditions

C1: L <= 0?
C2: Object is unique?

Actions

A1: Object is inserted without removal

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object inserted without removal</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Object inserted with removal</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest
.testAddAll</td>
<td>

Conditions

C1: L <= 0?
C2: Objects are unique?

Actions

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

A1: Objects are inserted without removals

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Objects inserted without removal</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Objects inserted with removal</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest.testSet</td>
<td>

Conditions

C1: L <= 0?
C2: Index in range?
C3: Object is unique

Actions

A1: Object at index is set without removal

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>C3</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Y/T</td><td>Object at index is set without removal</td></tr>
<tr><td>TC3</td><td>N/F</td><td>Y/T</td><td>N/F</td><td>Object at index is set with removal</td></tr>
<tr><td>TC4</td><td>N/F</td><td>N/F</td><td>-</td><td>Error</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest.testRemove</td>
<td>

Conditions

C1: L <= 0?
C2: Object in list?

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

Actions

A1: Object removed

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Object removed</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest .testRemoveIf</td>
<td>

Conditions

C1: L <= 0?
C2: Objects exist that should be removed?

Actions

A1: Object removed

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Objects removed</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>

</td>
</tr>
<tr>
<td>SetUniqueListTest .testRemoveAll</td>
<td>

Conditions

C1: L <= 0?
C2: Objects exist to be removed?

Actions

A1: Objects removed

Table

<table>
<tr><th>Test Case ID</th><th>C1</th><th>C2</th><th>Expected Output</th></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Objects</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td></td><td>removed</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>No change</td></tr>
</table>
○
</td></tr>
<tr><td>SetUniqueListTest
.testRetainAll</td><td>

Conditions

C1: L <= 0?
C2: Objects exist to be retained?

Actions

A1: Objects retained

Table

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Objects retained</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>Empty List</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest
.testContains</td><td>

Conditions

C1: L <= 0?
C2: Object in list?

Actions

A1: Object found in list

Table

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>True</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>False</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest
.testContainsAll</td><td>

Conditions

C1: L <= 0?
C2: Objects in list?

Actions

A1: Objects found in list

Table

</td></tr>
</table>

<table>
<tr><td></td><td>

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>True</td></tr>
<tr><td>TC3</td><td>N/F</td><td>N/F</td><td>False</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest
.testSublist</td><td>

Conditions

C1: L <= 0?
C2: Start index in range?
C3: End index in range?

Actions

A1: Sublist is made

Table

<table>
<tr><td>Test Case ID</td><td>C1</td><td>C2</td><td>C3</td><td>Expected Output</td></tr>
<tr><td>TC1</td><td>Y/T</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>TC2</td><td>N/F</td><td>Y/T</td><td>Y/T</td><td>Sublist</td></tr>
<tr><td>TC3</td><td>N/F</td><td>Y/T</td><td>N/F</td><td>Error</td></tr>
<tr><td>TC4</td><td>N/F</td><td>N/F</td><td>Y/T</td><td>Error</td></tr>
<tr><td>TC5</td><td>N/F</td><td>N/F</td><td>N/F</td><td>Error</td></tr>
</table>

</td></tr>
</table>

Test	Data Flow Coverage
PredicatedListTest .testEquals	●

Criterion	object	Program Paths	Test Inputs
All-def	<99, 100>	<99, 100>	(object = {2, 4, 6, 8, 10})
All-use	<99, 100>	<99, 100>	(object = {2, 4, 6, 8, 10})
All-P-user-so	<99, 100>	<99, 100>	(object = {2,

<table>
<tr><td></td><td>
<table>
<tr><td>me-C-uses</td><td></td><td></td><td>4, 6, 8, 10})</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><99, 100></td><td><99, 100></td><td>(object = {2, 4, 6, 8, 10})</td></tr>
<tr><td>All DU paths</td><td><99, 100></td><td><99, 100></td><td>(object = {2, 4, 6, 8, 10})</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest.testGet</td><td>

<table>
<tr><td>Criterion</td><td>index</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><110, 111></td><td><110, 111></td><td>(index = 0)</td></tr>
<tr><td>All-use</td><td><110, 111></td><td><110, 111></td><td>(index = 0)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><110, 111></td><td><110, 111></td><td>(index = 0)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><110, 111></td><td><110, 111></td><td>(index = 0)</td></tr>
<tr><td>All DU paths</td><td><110, 111></td><td><110, 111></td><td>(index = 0)</td></tr>
</table>

</td></tr>
<tr><td>PredicatedListTest.testIndexOf</td><td>

<table>
<tr><td>Criterion</td><td>object</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><115, 116></td><td><115, 116></td><td>(object = 2)</td></tr>
<tr><td>All-use</td><td><115, 116></td><td><115, 116></td><td>(object = 2)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><115, 116></td><td><115, 116></td><td>(object = 2)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><115, 116></td><td><115, 116></td><td>(object = 2)</td></tr>
<tr><td>All DU paths</td><td><115, 116></td><td><115, 116></td><td>(object = 2)</td></tr>
</table>

</td></tr>
<tr><td>PredicatedListTest.tesetLastIndexOf</td><td>

<table>
<tr><td>Criterion</td><td>object</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><120, 121></td><td><120, 121></td><td>(object = 2)</td></tr>
</table>

</td></tr>
</table>

	All-use	<120, 121>	<120, 121>	(object = 2)
	All-P-user-some-C-uses	<120, 121>	<120, 121>	(object = 2)
	All-C-user-some-P-uses	<120, 121>	<120, 121>	(object = 2)
	All DU paths	<120, 121>	<120, 121>	(object = 2)

PredicatedListTest.testRemove

-

Criterion	index	Program Paths	Test Inputs
All-def	<125, 126>	<125, 126>	(object = 2)
All-use	<125, 126>	<125, 126>	(object = 2)
All-P-user-some-C-uses	<125, 126>	<125, 126>	(object = 2)
All-C-user-some-P-uses	<125, 126>	<125, 126>	(object = 2)
All DU paths	<125, 126>	<125, 126>	(object = 2)

PredicatedListTest.testAdd

-

Criterion	index	object	Program Paths	Test Inputs
All-def	<131, 133>	<131, 132>	<131, 132, 133>	(object = 2)
All-use	<131, 133>	<131, 132> <131, 132, 133>	<131, 132, 133>	(object = 2)
All-P-user-some-C-uses	<131, 133>	<131, 132>	<131, 132, 133>	(object = 2)
All-C-user-some-P-uses	<131, 133>	<131, 132> <131, 132, 133>	<131, 132, 133>	(object = 2)

<table>
<tr><td></td><td><table>
<tr><td>All DU paths</td><td><131, 133></td><td><131, 132>
<131, 132, 133></td><td><131, 132, 133></td><td>(object = 2)</td></tr>
</table></td></tr>
<tr><td>PredicatedListTest.testAddAll</td><td>

<table>
<tr><td>Criterion</td><td>index</td><td>coll</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><137, {138, 139}*, 138, 141></td><td><137, 138></td><td><137, {138, 139}*, 138, 141></td><td>(coll = {12, 14, 16})</td></tr>
<tr><td>All-use</td><td><137, {138, 139}*, 138, 141></td><td><137, {138, 139}*, 138>
<137, {138, 139}*, 138, 139>
<137, {138, 139}*, 138, 139, 141></td><td><137, {138, 139}*, 138, 141></td><td>(coll = {12, 14, 16})</td></tr>
<tr><td>All-P-uses-some-C-uses</td><td><137, {138, 139}*, 138, 141></td><td><137, {138, 139}*, 138></td><td><137, {138, 139}*, 138, 141></td><td>(coll = {12, 14, 16})</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><137, {138, 139}*, 138, 141></td><td><137, {138, 139}*, 138>
<137, {138, 139}*, 138, 139>
<137, {138, 139}*, 138, 139, 141></td><td><137, {138, 139}*, 138, 141></td><td>(coll = {12, 14, 16})</td></tr>
<tr><td>All DU</td><td><137,</td><td><137,</td><td><137,</td><td>(coll = {12,</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>paths</td><td>{138, 139}*, 138, 141></td><td>{138, 139}*, 138>
<137, {138, 139}*, 138, 139>
<137, {138, 139}*, 138, 139, 141></td><td>{138, 139}*, 138, 141></td><td>14, 16})</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest
.testListIterator</td><td>

<table>
<tr><td>Criterion</td><td></td><td>Program Paths</td></tr>
<tr><td>All-def</td><td></td><td></td></tr>
<tr><td>All-use</td><td></td><td></td></tr>
<tr><td>All-P-user-some-C-uses</td><td></td><td></td></tr>
<tr><td>All-C-user-some-P-uses</td><td></td><td></td></tr>
<tr><td>All DU paths</td><td></td><td></td></tr>
</table>

</td></tr>
<tr><td>PredicatedListTest
.testSet</td><td>

<table>
<tr><td>Criterion</td><td>index</td><td>object</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><155, 156, 157></td><td><155, 156></td><td><155, 156, 157></td><td>(index = 0, object = 0)</td></tr>
<tr><td>All-use</td><td><155, 156, 157></td><td><155, 156>
<155, 156, 157></td><td><155, 156, 157></td><td>(index = 0, object = 0)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><155, 156, 157></td><td><155, 156></td><td><155, 156, 157></td><td>(index = 0, object = 0)</td></tr>
<tr><td>All-C-user-some-P-us</td><td><155, 156, 157></td><td><155, 156></td><td><155, 156, 157></td><td>(index = 0, object = 0)</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>es</td><td></td><td><155, 156, 157></td><td></td><td></td></tr>
<tr><td>All DU paths</td><td><155, 156, 157></td><td><155, 156>
<155, 156, 157></td><td><155, 156, 157></td><td>(index = 0, object = 0)</td></tr>
</table>
</td></tr>
<tr><td>SetUniqueListTest .testAdd</td><td>

<table>
<tr><td>Criterion</td><td>object</td><td>sizeBefore</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><125, 127, 130></td><td><127, 130, 133></td><td><125, 127, 130, 133></td><td>(object = 1)</td></tr>
<tr><td>All-use</td><td><125, 127, 130></td><td><127, 130, 133></td><td><125, 127, 130, 133></td><td>(object = 1)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><125, 127, 130></td><td><127, 130, 133></td><td><125, 127, 130, 133></td><td>(object = 1)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><125, 127, 130></td><td><127, 130, 133></td><td><125, 127, 130, 133></td><td>(object = 1)</td></tr>
<tr><td>All DU paths</td><td><125, 127, 130></td><td><127, 130, 133></td><td><125, 127, 130, 133></td><td>(object = 1)</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest .testAddAll</td><td>

<table>
<tr><td>Criterion</td><td>coll</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><170, 171></td><td><170, 171></td><td>(coll = {1, 2, 3, 4, 5, 1, 2, 3})</td></tr>
<tr><td>All-use</td><td><170, 171></td><td><170, 171></td><td>(coll = {1, 2, 3, 4, 5, 1, 2, 3})</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><170, 171></td><td><170, 171></td><td>(coll = {1, 2, 3, 4, 5, 1, 2, 3})</td></tr>
<tr><td>All-C-user-so</td><td><170, 171></td><td><170, 171></td><td>(coll = {1, 2,</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>me-P-uses</td><td></td><td></td><td>3, 4, 5, 1, 2, 3})</td></tr>
<tr><td>All DU paths</td><td><170, 171></td><td><170, 171></td><td>(coll = {1, 2, 3, 4, 5, 1, 2, 3})</td></tr>
</table>
</td></tr>
<tr><td>SetUniqueListTest.testSet</td><td>

<table>
<tr><th>Criterion</th><th>index</th><th>object</th><th>pos</th><th>removed</th><th>Program Paths</th><th>Test Inputs</th></tr>
<tr><td>All-def</td><td><213, 214, 215></td><td><213, 214></td><td><214, 215, {217, 220}*, 217></td><td><215, {217, 220}*, 217, 223></td><td><213, 214, 215, {217, 220}*, 217, 223, 224, 226></td><td>(index = 1, object = 5)</td></tr>
<tr><td>All-use</td><td><213, 214, 215>
<213, 214, 215, {217, 220}*, 217></td><td><213, 214>
<213, 214, 215>
<213, 214, 215, {217, 220}*, 217, 223, 224></td><td><214, 215, {217, 220}*, 217>
<214, 215, {217, 220}*></td><td><215, {217, 220}*, 217, 223>
<215, {217, 220}*, 217, 223, 224, 226></td><td><213, 214, 215, {217, 220}*, 217, 223, 224, 226></td><td>(index = 1, object = 5)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><213, 214, 215, {217, 220}*, 217></td><td><213, 214></td><td><214, 215, {217, 220}*, 217></td><td><215, {217, 220}*, 217, 223></td><td><213, 214, 215, {217, 220}*, 217, 223, 224, 226></td><td>(index = 1, object = 5)</td></tr>
<tr><td>All-C-user-so</td><td><213, 214,</td><td><213, 214></td><td><214, 215,</td><td><215, {217,</td><td><213, 214,</td><td>(index = 1,</td></tr>
</table>

</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>me-P-uses</td><td>215></td><td><213, 214, 215> <213, 214, 215, {217, 220}*, 217, 223, 224></td><td>{217, 220}*></td><td>220}*, 217, 223> <215, {217, 220}*, 217, 223, 224, 226></td><td>215, {217, 220}*, 217, 223, 224, 226></td><td>object = 5)</td></tr>
<tr><td>All DU paths</td><td><213, 214, 215> <213, 214, 215, {217, 220}*, 217></td><td><213, 214> <213, 214, 215> <213, 214, 215, {217, 220}*, 217, 223, 224></td><td><214, 215, {217, 220}*, 217> <214, 215, {217, 220}*></td><td><215, {217, 220}*, 217, 223> <215, {217, 220}*, 217, 223, 224, 226></td><td><213, 214, 215, {217, 220}*, 217, 223, 224, 226></td><td>(index = 1, object = 5)</td></tr>
</table>
</td>
</tr>
<tr>
<td>SetUniqueListTest.testRemove</td>
<td>

<table>
<tr><th>Criterion</th><th>index</th><th>result</th><th>Program Paths</th><th>Test Inputs</th></tr>
<tr><td>All-def</td><td><239, 240></td><td><240, 241></td><td><239, 240, 241, 242></td><td>(index = 2)</td></tr>
<tr><td>All-use</td><td><239, 240></td><td><240, 241> <240, 241, 242></td><td><239, 240, 241, 242></td><td>(index = 2)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><239, 240></td><td><240, 241></td><td><239, 240, 241, 242></td><td>(index = 2)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><239, 240></td><td><240, 241> <240, 241, 242></td><td><239, 240, 241, 242></td><td>(index = 2)</td></tr>
<tr><td>All DU</td><td><239,</td><td><240,</td><td><239, 240,</td><td>(index = 2)</td></tr>
</table>

</td>
</tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>paths</td><td>240></td><td>241>
<240, 241, 242></td><td>241, 242></td><td></td></tr>
</table>
</td></tr>
<tr><td>SetUniqueListTest
.testRemoveIf</td><td>

<table>
<tr><td>Criterion</td><td>filter</td><td>result</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><249, 250></td><td><250, 251, 252></td><td><249, 250, 251, 252></td><td>(filter = (list element % 2) == 0)</td></tr>
<tr><td>All-use</td><td><249, 250>
<249, 250, 251></td><td><250, 251, 252></td><td><249, 250, 251, 252></td><td>(filter = (list element % 2) == 0)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><249, 250></td><td><250, 251, 252></td><td><249, 250, 251, 252></td><td>(filter = (list element % 2) == 0)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><249, 250>
<249, 250, 251></td><td><250, 251, 252></td><td><249, 250, 251, 252></td><td>(filter = (list element % 2) == 0)</td></tr>
<tr><td>All DU paths</td><td><249, 250>
<249, 250, 251></td><td><250, 252></td><td><249, 250, 251, 252></td><td>(filter = (list element % 2) == 0)</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest
.testRemoveAll</td><td>

<table>
<tr><td>Criterion</td><td>coll</td><td>result</td><td>name</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><256, 257, {258, 259}*, 258></td><td><257, {258, 259}*, 258, 259></td><td><258, {259, 258}*, 259></td><td><256, 257, {258, 259}*, 258, 261></td><td>(coll = {2, 3, 4, 6})</td></tr>
<tr><td>All-use</td><td><256, 257,</td><td><257, {258,</td><td><258, {259,</td><td><256, 257,</td><td>(coll = {2, 3, 4,</td></tr>
</table>

</td></tr>
</table>

		{258, 259}*, 258>	259}*, 258, 259> <257, {258, 259}*, 258, 261>	258}*, 259>	{258, 259}*, 258, 261>	6})
	All-P-user-some-C-uses	<256, 257, {258, 259}*, 258>	<257, {258, 259}*, 258, 259>	<258, {259, 258}*, 259>	<256, 257, {258, 259}*, 258, 261>	(coll = {2, 3, 4, 6})
	All-C-user-some-P-uses	<256, 257, {258, 259}*, 258>	<257, {258, 259}*, 258, 259> <257, {258, 259}*, 258, 261>	<258, {259, 258}*, 259>	<256, 257, {258, 259}*, 258, 261>	(coll = {2, 3, 4, 6})
	All DU paths	<256, 257, {258, 259}*, 258>	<257, {258, 259}*, 258, 259> <257, {258, 259}*, 258, 261>	<258, {259, 258}*, 259>	<256, 257, {258, 259}*, 258, 261>	(coll = {2, 3, 4, 6})
SetUniqueListTest.testRetainAll	●					

Criterion	coll	result	Program Paths	Test Inputs
All-def	<274, 275>	<275, 276>	<274, 275, 276, 279, 281, 285>	(coll = {1, 5, 7})
All-use	<274, 275>	<275, 276>	<274, 275, 276, 279,	(coll = {1, 5, 7})

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td><275, 276, 279, 281, 285></td><td>281, 285></td><td></td></tr>
<tr><td>All-P-user-some-C-uses</td><td><274, 275></td><td><275, 276></td><td><274, 275, 276, 279, 281, 285></td><td>(coll = {1, 5, 7})</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><274, 275></td><td><275, 276, 279, 281, 285></td><td><274, 275, 276, 279, 281, 285></td><td>(coll = {1, 5, 7})</td></tr>
<tr><td>All DU paths</td><td><274, 275></td><td><275, 276>
<275, 276, 279, 281, 285></td><td><274, 275, 276, 279, 281, 285></td><td>(coll = {1, 5, 7})</td></tr>
</table>
</td></tr>
<tr><td>SetUniqueListTest.testContains</td><td>

<table>
<tr><td>Criterion</td><td>object</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><295, 296></td><td><295, 296></td><td>(object = 1)</td></tr>
<tr><td>All-use</td><td><295, 296></td><td><295, 296></td><td>(object = 1)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><295, 296></td><td><295, 296></td><td>(object = 1)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><295, 296></td><td><295, 296></td><td>(object = 1)</td></tr>
<tr><td>All DU paths</td><td><295, 296></td><td><295, 296></td><td>(object = 1)</td></tr>
</table>

</td></tr>
<tr><td>SetUniqueListTest.testContainsAll</td><td>

<table>
<tr><td>Criterion</td><td>coll</td><td>Program Paths</td><td>Test Inputs</td></tr>
<tr><td>All-def</td><td><300, 301></td><td><300, 301></td><td>(coll = {1, 5})</td></tr>
<tr><td>All-use</td><td><300, 301></td><td><300, 301></td><td>(coll = {1, 5})</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><300, 301></td><td><300, 301></td><td>(coll = {1, 5})</td></tr>
<tr><td>All-C-user-so</td><td><300, 301></td><td><300, 301></td><td>(coll = {1, 5})</td></tr>
</table>

</td></tr>
</table>

	me-P-uses			
	All DU paths	<300, 301>	<300, 301>	(coll = {1, 5})