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>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

Test	Boundary Value Analysis
PredicatedListTest .testGet	● 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><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></table>

<table>
<tr>
<td></td>
<td>
<table>
<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>
<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)</td><td>L > 0</td><td>L - 1</td><td>O can be</td></tr>
</table>

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

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>5</td><td></td><td></td><td>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>
</table>

<table>
<tr><td>Test</td><td>Decision Table</td></tr>
<tr>
<td>PredicatedListTest
.testEquals</td>
<td>

Conditions

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

Actions

A1: Objects are found to be equal

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>
</table>

<table>
<tr><td></td><td>○</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?
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>
</table>

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

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

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

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

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>
<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

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

	○

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

Test	Data Flow Coverage
PredicatedListTest .testEquals	●
PredicatedListTest .testGet	●

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-some-C-uses	<99, 100>	<99, 100>	(object = {2, 4, 6, 8, 10})
All-C-user-some-P-uses	<99, 100>	<99, 100>	(object = {2, 4, 6, 8, 10})
All DU paths	<99, 100>	<99, 100>	(object = {2, 4, 6, 8, 10})

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

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

</td></tr>
<tr><td>PredicatedListTest.testRemove</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><125, 126></td><td><125, 126></td><td>(object = 2)</td></tr>
<tr><td>All-use</td><td><125, 126></td><td><125, 126></td><td>(object = 2)</td></tr>
</table>

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

<table>
<tr><td></td><td>
<table>
<tr><td>All-P-user-some-C-uses</td><td><125, 126></td><td><125, 126></td><td>(object = 2)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><125, 126></td><td><125, 126></td><td>(object = 2)</td></tr>
<tr><td>All DU paths</td><td><125, 126></td><td><125, 126></td><td>(object = 2)</td></tr>
</table>
</td></tr>
<tr><td>PredicatedListTest.testAdd</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><131, 133></td><td><131, 132></td><td><131, 132, 133></td><td>(object = 2)</td></tr>
<tr><td>All-use</td><td><131, 133></td><td><131, 132>
<131, 132, 133></td><td><131, 132, 133></td><td>(object = 2)</td></tr>
<tr><td>All-P-user-some-C-uses</td><td><131, 133></td><td><131, 132></td><td><131, 132, 133></td><td>(object = 2)</td></tr>
<tr><td>All-C-user-some-P-uses</td><td><131, 133></td><td><131, 132>
<131, 132, 133></td><td><131, 132, 133></td><td>(object = 2)</td></tr>
<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}*,</td><td><137, {138, 139}*,</td><td><137, {138, 139}*,</td><td>(coll = {12, 14, 16})</td></tr>
</table>

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

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>138, 141></td><td>138>
<137,
{138,
139}*,
138, 139>
<137,
{138,
139}*,
138, 139,
141></td><td>138, 141></td><td></td></tr>
<tr><td>All-P-uses
-some-C-u
ses</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-us
es</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
paths</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>
</table>
</td></tr>
<tr><td>PredicatedListTest
.testListIterator</td><td>●
<table>
<tr><td>Criterion</td><td></td><td>Program Paths</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<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><th>Criterion</th><th>index</th><th>object</th><th>Program Paths</th><th>Test Inputs</th></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-uses</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 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>
</table>