



Pitest

...

Mutation tests, the third step in testing your code

What is PIT?

- PIT is a mutation testing system
- PIT automatically modifies versions of your application code
- PIT runs the modified application code against your unit tests

PIT default mutations

- Conditionals Boundary Mutator
- Increments Mutator
- Invert Negatives Mutator
- Math Mutator
- Negate Conditionals Mutator
- Return Values Mutator
- Void Method Calls Mutator

$(a < b) \Rightarrow (a \leq b)$

$i++; \Rightarrow i--;$

$\text{return } -i; \Rightarrow \text{return } i;$

$a = b + c; \Rightarrow a = b - c;$

$(a == b) \Rightarrow (a != b)$

$\text{return size} \Rightarrow \text{return } (x == 0 ? 1 : 0)$

remove void method calls

```
public float negate(final float i) {  
    return -i;  
}
```

=>

```
public float negate(final float i) {  
    return i;  
}
```

Why use pitest?

- Traditional test coverage measures only which code is executed by your tests.
- It does not check that your tests are actually able to detect faults in the executed code.
- It is therefore only able to identify code the is definitely not tested.

Example sen project

- tddlist project -> test driven development of a simplelist interface
- Implements SimpleList class in 4 different ways
 - With array backend
 - LinkedList with dummy head and tail
 - LinkedList with null head and tail
 - LinkedList with dummy head and tail node, in beginning head.next == tail

Modifier and Type	Method and Description
void	<code>add(E e)</code> Add (append) element to end of this list.
boolean	<code>contains(E e)</code> Check for presence of the element in the list.
E	<code>get()</code> Get element at index 0 from this list.
E	<code>get(int idx)</code> Get element from index in this list.
default boolean	<code>isEmpty()</code> Test if the list is empty.
Iterator<E>	<code>iterator()</code> Create an Iterator for this list.
int	<code>size()</code> Report the number of elements in the list.
void	<code>sort(Comparator<? super E> c)</code> Sort this list by the order dictated by the given Comparator.
E	<code>take()</code> Take the first element from this list and remove it from this list.
E	<code>take(int i)</code> Take the i-th element from this list and remove it from this list.






















Unit tests

- All test passed



Code coverage

- First converted to maven project
- Added Jacoco code coverage
- 100% code coverage

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
SimpleArrayList		100%		100%	0	31	0	69	0	17	0	1
SimpleLinkedList3		100%		100%	0	28	0	73	0	15	0	1
SimpleLinkedList2		100%		100%	0	31	0	78	0	14	0	1
SimpleLinkedList		100%		100%	0	28	0	69	0	14	0	1
SimpleArrayList.new Iterator() {...}		100%		100%	0	4	0	5	0	3	0	1
SimpleLinkedList3.new Iterator() {...}		100%		100%	0	4	0	6	0	3	0	1
SimpleLinkedList.new Iterator() {...}		100%		100%	0	4	0	6	0	3	0	1
SimpleLinkedList2.new Iterator() {...}		100%		100%	0	4	0	6	0	3	0	1
SimpleLinkedList.Link		100%		n/a	0	1	0	4	0	1	0	1
SimpleLinkedList2.Link		100%		n/a	0	1	0	4	0	1	0	1
SimpleLinkedList3.Link		100%		n/a	0	1	0	4	0	1	0	1
SimpleList		100%		100%	0	2	0	1	0	1	0	1
Total	0 of 1,269	100%	0 of 126	100%	0	139	0	321	0	76	0	12

But there is more: Pitest

- Conditional boundaries: 10x
- Negated conditional: 3 x
- Integer division <-> multiplication: 2 x
- Integer addition <-> subtraction: 2 x

Number of Classes	Line Coverage	Mutation Coverage
5	100% <div>314/314</div>	91% <div>176/193</div>

Breakdown by Class

Name	Line Coverage	Mutation Coverage
SimpleArrayList.java	100% <div>75/75</div>	86% <div>54/63</div>
SimpleLinkedList.java	100% <div>75/75</div>	98% <div>40/41</div>
SimpleLinkedList2.java	100% <div>84/84</div>	95% <div>42/44</div>
SimpleLinkedList3.java	100% <div>79/79</div>	88% <div>38/43</div>
SimpleList.java	100% <div>1/1</div>	100% <div>2/2</div>