

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

```
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.

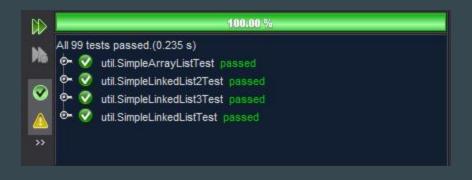
Example sen project

- 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	add (E e) Add (append) element to end of this list.
boolean	contains (E e) Check for presence of the element in the list.
E	get () Get element at index o from this list.
E	get(int idx) Get element from index in this list.
default boolean	isEmpty() Test if the list is empty.
Iterator <e></e>	iterator() Create an Iterator for this list.
int	size() Report the number of elements in the list.
void	sort (Comparator super E c) Sort this list by the order dictated by the given Comparator.
E	take () Take the first element from this list and remove it from this list.
Е	take (int i) 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		Missed	Cxty	Missed	Lines	Missed	Methods =	Missed	Classes
		100%	8	100%	0	31	0	69	0	17	0	1
→ SimpleLinkedList3		100%		100%	0	28	0	73	0	15	0	1
		100%		100%	0	31	0	78	0	14	0	1
		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
		100%	=	100%	0	4	0	6	0	3	0	1
	1	100%		n/a	0	1	0	4	0	1	0	1
	1	100%		n/a	0	1	0	4	0	1	0	1
	1	100%		n/a	0	1	0	4	0	1	0	1
	1	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 C	overage	Mutation Coverage				
5 1009	%	314/314		176/193			
Breakdown by Class	i						
Name	Lin	ne Coverage		Mutati	tation Coverage		
SimpleArrayList.java	100%	75/75		86%	54/63		
SimpleLinkedList.java	100%	75/75		98%	40/41		
SimpleLinkedList2.java	100%	84/84		95%	42/44		
SimpleLinkedList3.java	100%	79/79		88%	38/43		
SimpleList.java	100%	1/1	- 11	100%	2/2		