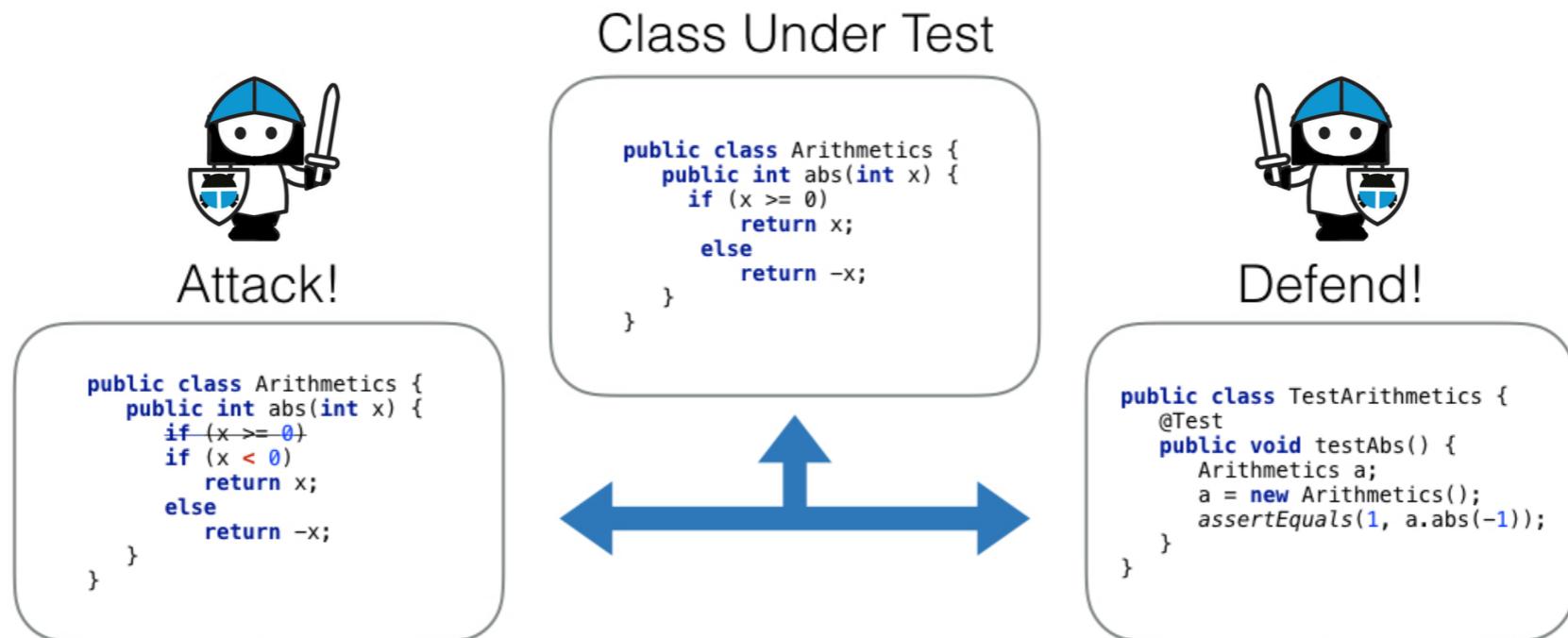




Code Defenders

Teaching Software Testing Concepts using a
Mutation Testing Game

Benjamin Clegg, José Miguel Rojas, Gordon Fraser



The
University
Of
Sheffield.





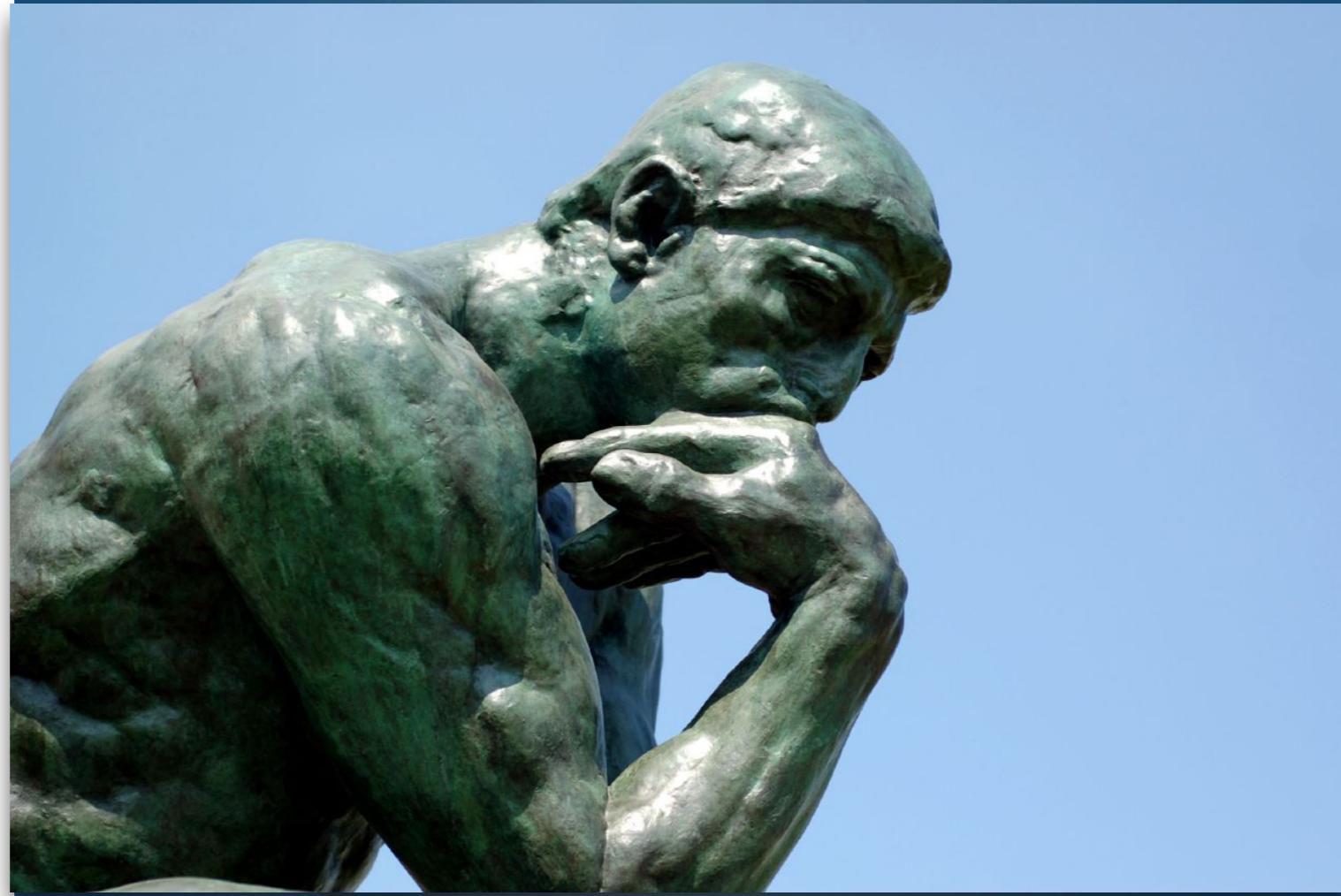
Software Testing



Fundamental for Software Quality



Software Testing



by Brian Hillegas (Flickr)

Fundamental for Software Quality
...but *very* demanding



Software Testing Education





Code Defenders



by François Philipp (Flickr)

Software Testing Gamification



Mutation Testing



Mutation Testing

Helps shifting testing effort from coverage to fault detection



Mutation Testing

Helps shifting testing effort from coverage to fault detection



Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
}
```



Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
@Test  
void testAbs() {  
    int res = abs(42);  
    assertEquals(42, res);  
}
```

```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
}
```



Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
@Test  
void testAbs() {  
    int res = abs(42);  
    assertEquals(42, res);  
}
```



```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
}
```



Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
@Test  
void testAbs() {  
    int res = abs(42);  
    assertEquals(42, res);  
}
```



```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
  
int abs(int x) {  
    if (x <= 0)  
        return x;  
    else  
        return -x;  
}
```

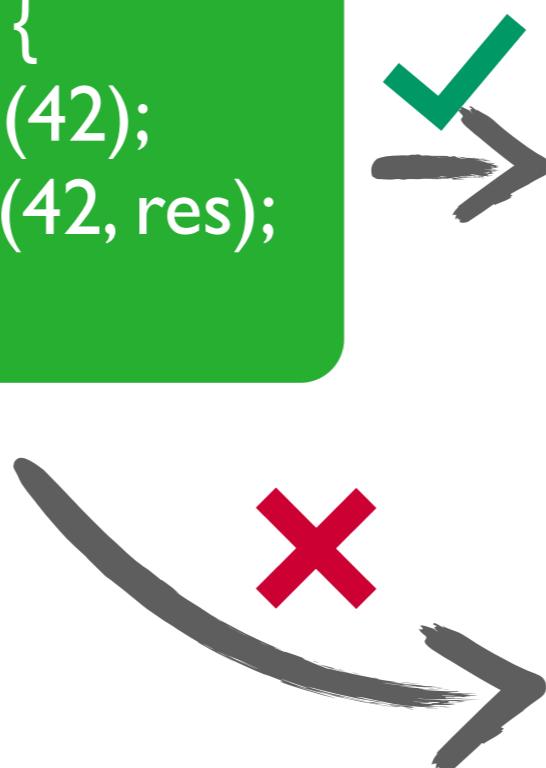


Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
@Test  
void testAbs() {  
    int res = abs(42);  
    assertEquals(42, res);  
}
```

```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
  
int abs(int x) {  
    if (x <= 0)  
        return x;  
    else  
        return -x;  
}
```





Mutation Testing

Helps shifting testing effort from coverage to fault detection

```
@Test  
void testAbs() {  
    int res = abs(42);  
    assertEquals(42, res);  
}
```

```
int abs(int x) {  
    if (x >= 0)  
        return x;  
    else  
        return -x;  
  
int abs(int x) {  
    if (x <= 0)  
        return x;  
    else  
        return -x;  
}
```

Too many mutants, often trivial, redundant or equivalent



Code Defenders

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

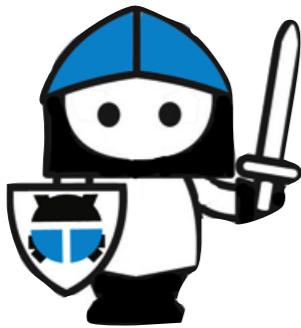


Code Defenders

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Attackers



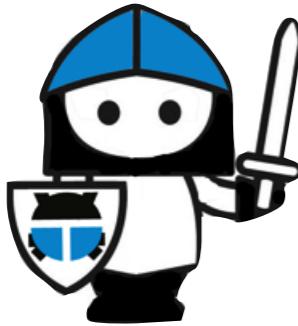


Code Defenders

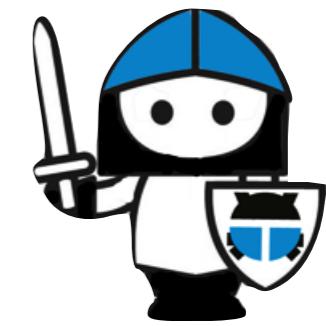
Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Attackers



Defenders





Code Defenders

Class Under Test

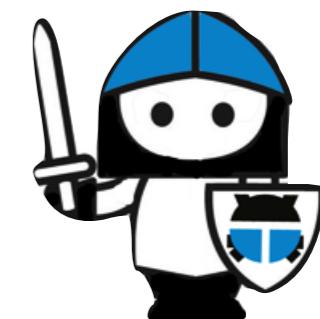
```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Attackers



Defenders



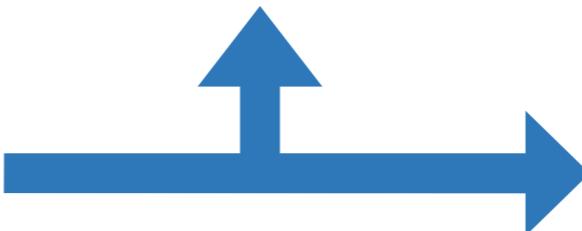


Code Defenders

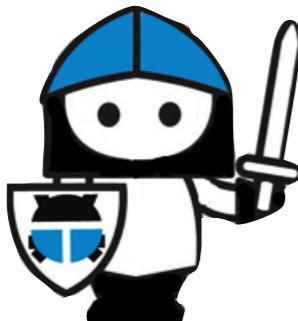
Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

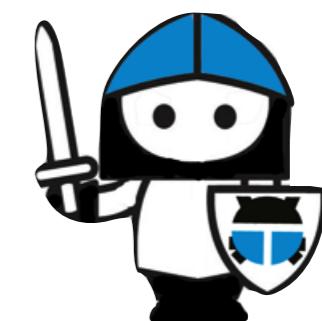
```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```



Attackers



Defenders

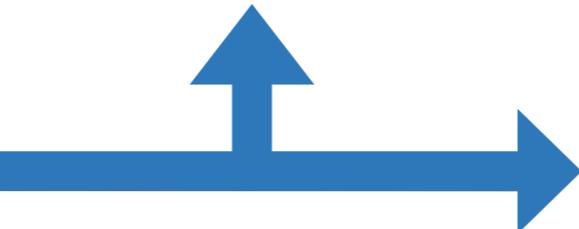




Code Defenders

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```



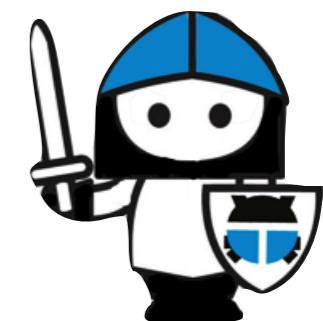
```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Attackers



```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

Defenders





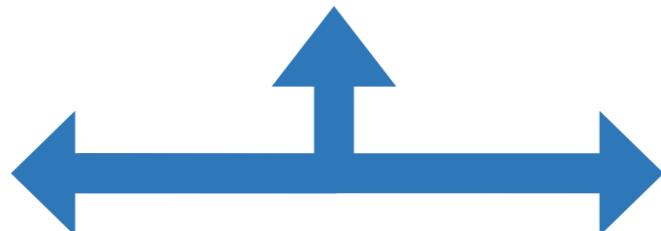
Code Defenders

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

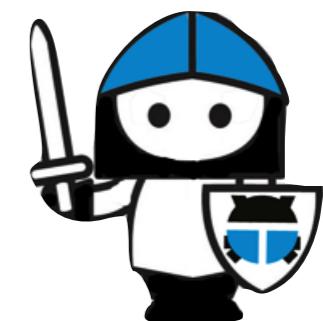
```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```



Attackers



Defenders





Code Defenders

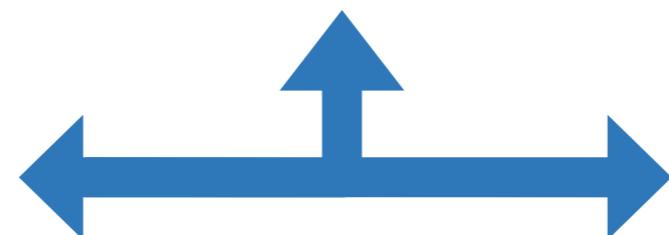
Score points for surviving mutants

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

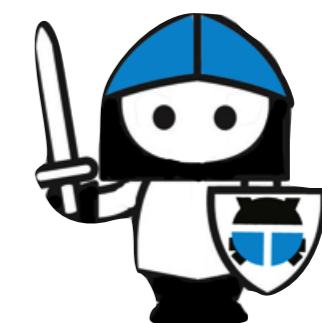
```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```



Attackers



Defenders





Code Defenders

Score points for surviving mutants

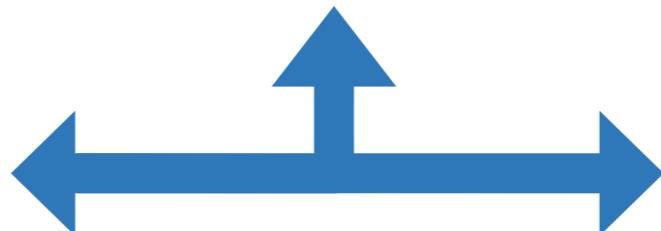
```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Attackers



Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```



Score points for effective tests

```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

Defenders





Code Defenders

Score points for surviving mutants

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

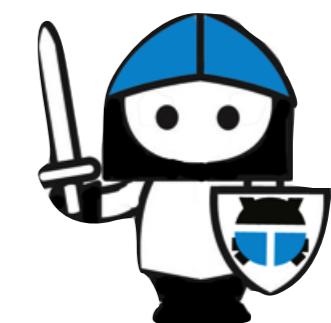
Score points for effective tests

```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

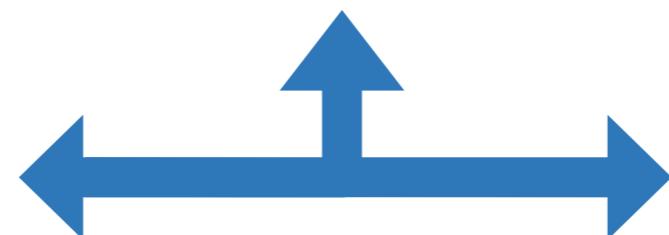
Attackers



Defenders



Equivalent Mutant Duels





Code Defenders

Score points for surviving mutants

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

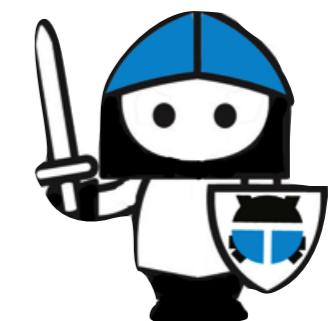
Score points for effective tests

```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

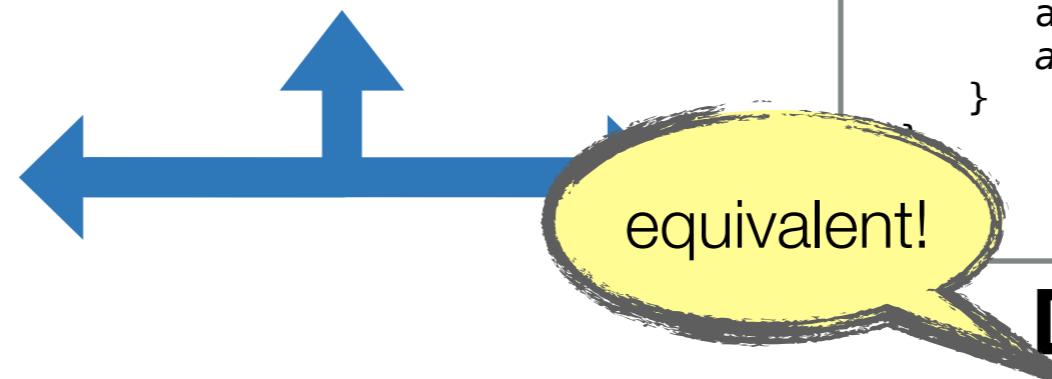
Attackers



Defenders



Equivalent Mutant Duels





Code Defenders

Score points for surviving mutants

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Score points for effective tests

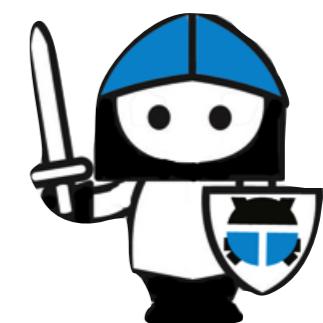
```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

no way! here is a killing test!

equivalent!

Attackers

Defenders



Equivalent Mutant Duels



Code Defenders

Score points for surviving mutants

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Class Under Test

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Score points for effective tests

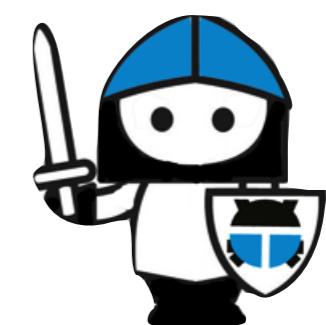
```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

no way! here is a killing test!

equivalent!

Attackers

Defenders



Equivalent Mutant Duels



Code Defenders

Class Under Test

Score points for surviving mutants

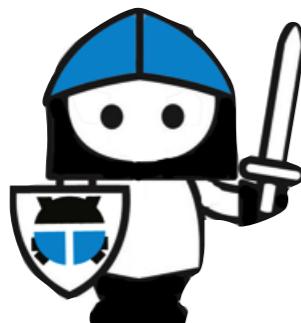
```
public class Arithmetics {  
    public int abs(int x) {  
        if (x < 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

```
public class Arithmetics {  
    public int abs(int x) {  
        if (x >= 0)  
            return x;  
        else  
            return -x;  
    }  
}
```

Score points for effective tests

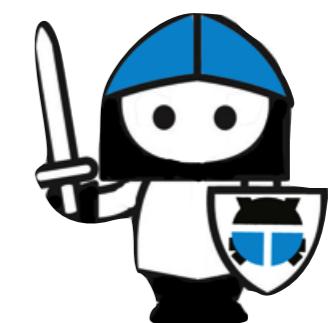
```
public class TestArithmetics {  
    @Test  
    public void testAbs() {  
        Arithmetics a;  
        a = new Arithmetics();  
        assertEquals(1, a.abs(-1));  
    }  
}
```

Attackers



no way! here is
a killing test!
oh no! :(

Defenders



equivalent!

Equivalent Mutant Duels



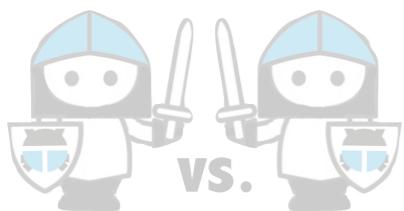
Code Defenders



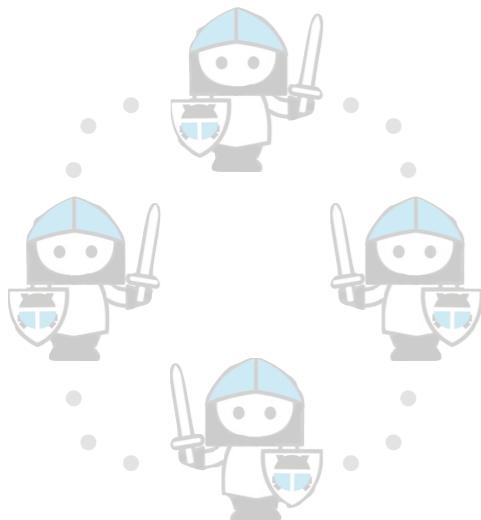
Single-player

vs.

EVASUITE / MAJOR



Duels



Multi-player

The screenshot displays the Code Defenders web application. At the top, there's a navigation bar with links for 'games', 'upload class', 'leaderboard', 'help', and a user profile for 'neilwalkinshaw'. Below the navigation is a section titled 'DEFENDER::ACTIVE' for 'SparseIntArray'. It shows the 'Class Under Test' code:

```
36 /**
37  * Creates a new SparseIntArray containing no mappings that will not
38  * require any additional memory allocation to store the specified
39  * number of mappings. If you supply an initial capacity of 0, the
40  * sparse array will be initialized with a light-weight representation
41  * not requiring any additional array allocations.
42 */
43 public SparseIntArray(int initialCapacity) {
44     if (initialCapacity == 0) {
45         mKeys = SparseIntArray.EMPTY_INT_ARRAY;
46         mValues = SparseIntArray.EMPTY_INT_ARRAY;
47     } else {
48         mKeys = new int[initialCapacity];
49         mValues = new int[mKeys.length];
50     }
51     mSize = 0;
52 }
53 /**
54  * Given the current size of an array, returns an ideal size to which the array should
55  * grow. This is typically double the given size, but should not be relied upon to do so in the
56  * future.
57 */
58 public static int growSize(int currentSize) {
59     return currentSize <= 4 ? 8 : currentSize + (currentSize >> 1);
60 }
```

To the right, there's a text area for 'Write a new JUnit test here' with a 'Defend!' button. Below the code editor, there are sections for 'Existing Mutants' and 'JUnit tests'.

Existing Mutants

alive (35)	killed(68)	equivalent(5)
Mutant 5838 Creator: amin [UID: 428] Modified line 244	<button>Claim Equivalent</button>	
Mutant 5855 Creator: abrahamc2 [UID: 432] Modified line 211	<button>Claim Equivalent</button>	
Mutant 5823 Creator: gregoryg [UID: 426] Modified line 178	<button>Claim Equivalent</button>	
Mutant 5886 Creator: gregoryg [UID: 426] Modified line 190	<button>Claim Equivalent</button>	

JUnit tests

```
1 /* no package name */
2
3 import org.junit.*;
4 import static org.junit.Assert.*;
5
6 public class TestSparseIntArray {
7     @Test(timeout = 4000)
8     public void test() throws Throwable {
9         SparseIntArray sia = new SparseIntArray();
10         sia.put(0, 7);
11         sia.put(1, 8);
12         sia.delete(0);
13         assertEquals(1, sia.size());
14         assertEquals(8, sia.get(1));
15     }
16 }
```



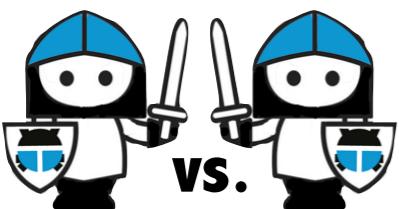
Code Defenders



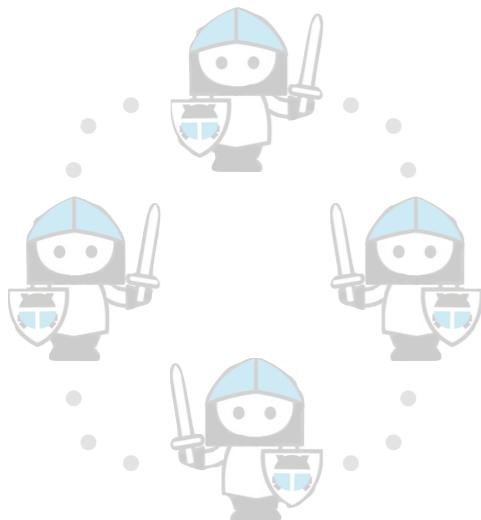
Single-player

vs.

EVASUITE / MAJOR



Duels



Multi-player

The screenshot shows the Code Defenders web application. At the top, there's a navigation bar with links for 'games', 'upload class', 'leaderboard', 'help', and a user profile for 'neilwalkinshaw'. Below the navigation is a section titled 'DEFENDER::ACTIVE' for 'SparseIntArray'. It displays the 'Class Under Test' code:

```
1 /**
2  * Creates a new SparseIntArray containing no mappings that will not
3  * require any additional memory allocation to store the specified
4  * number of mappings. If you supply an initial capacity of 0, the
5  * sparse array will be initialized with a light-weight representation
6  * not requiring any additional array allocations.
7 */
8 public SparseIntArray(int initialCapacity) {
9     if (initialCapacity == 0) {
10         mKeys = SparseIntArray.EMPTY_INT_ARRAY;
11         mValues = SparseIntArray.EMPTY_INT_ARRAY;
12     } else {
13         mKeys = new int[initialCapacity];
14         mValues = new int[mKeys.length];
15     }
16     mSize = 0;
17 }
18 /**
19  * Given the current size of an array, returns an ideal size to which the array should
20  * grow. This is typically double the given size, but should not be relied upon to do so in the
21  * future.
22 */
23 public static int growSize(int currentSize) {
24     return currentSize <= 4 ? 8 : currentSize + (currentSize >> 1);
25 }
```

To the right of the code editor is a panel for 'SparseIntArray' with a heading 'Write a new JUnit test here' and a 'Defend!' button. Below the code editor is a section titled 'Existing Mutants' listing four mutants:

- Mutant 5838 | Creator: amin [UID: 428] - alive (35), killed(68), equivalent(5)
- Mutant 5855 | Creator: abrahamc2 [UID: 432] - alive (35), killed(68), equivalent(5)
- Mutant 5823 | Creator: gregoryg [UID: 426] - alive (35), killed(68), equivalent(5)
- Mutant 5886 | Creator: gregoryg [UID: 426] - alive (35), killed(68), equivalent(5)

At the bottom of the interface, there are sections for 'JUnit tests' and a pagination control.



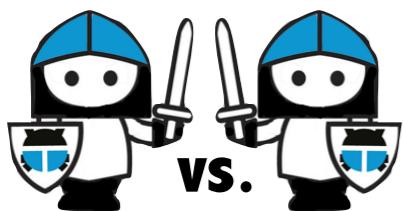
Code Defenders



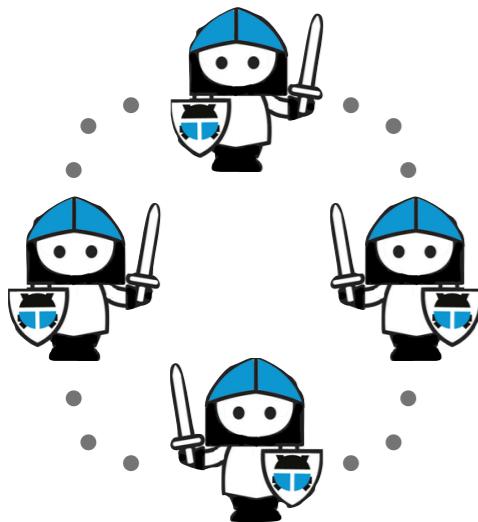
Single-player

vs.

EVASUITE / MAJOR



Duels



Multi-player

The screenshot shows the Code Defenders web application. At the top, there's a navigation bar with links for 'games', 'upload class', 'leaderboard', 'help', and a user profile for 'neilwalkinshaw'. Below the navigation is a section titled 'DEFENDER::ACTIVE' for 'SparseIntArray'. It shows the 'Class Under Test' code:

```
35 /**
 * Creates a new SparseIntArray containing no mappings that will not
 * require any additional memory allocation to store the specified
 * number of mappings. If you supply an initial capacity of 0, the
 * sparse array will be initialized with a light-weight representation
 * not requiring any additional array allocations.
 */
36
37 public SparseIntArray(int initialCapacity) {
38     if (initialCapacity == 0) {
39         mKeys = SparseIntArray.EMPTY_INT_ARRAY;
40         mValues = SparseIntArray.EMPTY_INT_ARRAY;
41     } else {
42         mKeys = new int[initialCapacity];
43         mValues = new int[mKeys.length];
44     }
45     mSize = 0;
46 }
47
48 /**
 * Given the current size of an array, returns an ideal size to which the array should
 * grow. This is typically double the given size, but should not be relied upon to do so in the
 * future.
 */
49 public static int growSize(int currentSize) {
50     return currentSize <= 4 ? 8 : currentSize + (currentSize >> 1);
51 }
```

To the right of the code editor is a 'Write a new JUnit test here' area with a 'Defend!' button. Below the code editor is a 'Existing Mutants' section listing four mutants:

- Mutant 5838 | Creator: amin [UID: 428] - Modified line 244
- Mutant 5855 | Creator: abrahamc2 [UID: 432] - Modified line 211
- Mutant 5823 | Creator: gregoryg [UID: 426] - Modified line 178
- Mutant 5886 | Creator: gregoryg [UID: 426] - Modified line 190

At the bottom of the interface, there are buttons for 'First', 'Previous', 'Next', and 'Last'.



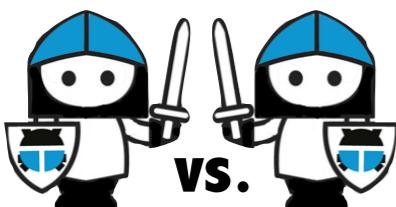
Code Defenders



Single-player

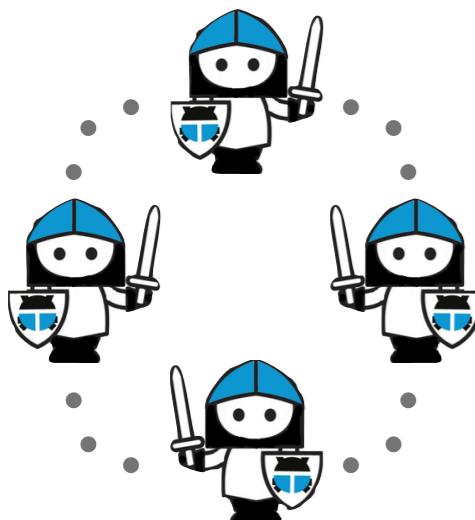
vs.

EVSUITE / MAJOR



Duels

vs.



Multi-player

DEFENDER::ACTIVE

Class Under Test

```
55  /**
56  * Creates a new SparseIntArray containing no mappings that will not
57  * require any additional memory allocation to store the specified
58  * number of mappings. If you supply an initial capacity of 0, the
59  * sparse array will be initialized with a light-weight representation
60  * not requiring any additional array allocations.
61  */
62
63  /**
64  * Given the current size of an array, returns an ideal size to which the array should
65  * be grown. This is typically double the given size, but should not be relied upon to do so in
66  * all cases.
67  */
68  public static int growSize(int currentSize) {
69      return currentSize <= 4 ? 8 : currentSize + (currentSize >> 1);
70  }
```

Existing Mutants

alive (35)

killed(68)

equivalent(5)

Search...



Early Adopters



Early Adopters



Paul Amman



Early Adopters



Jeff Offutt

Paul Amman

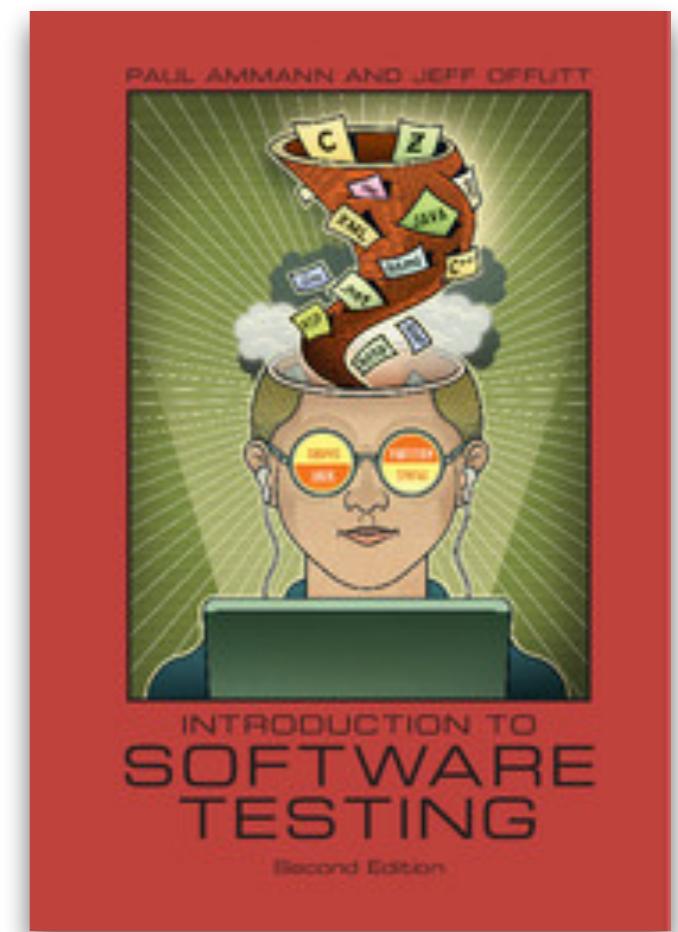


Early Adopters



Jeff Offutt

Paul Amman



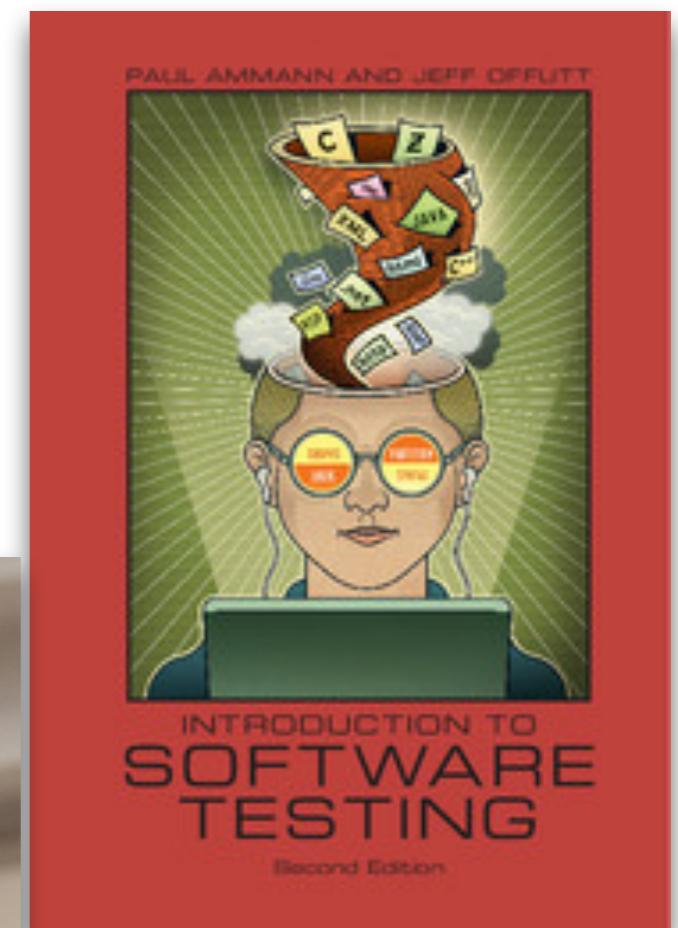


Early Adopters



Jeff Offutt

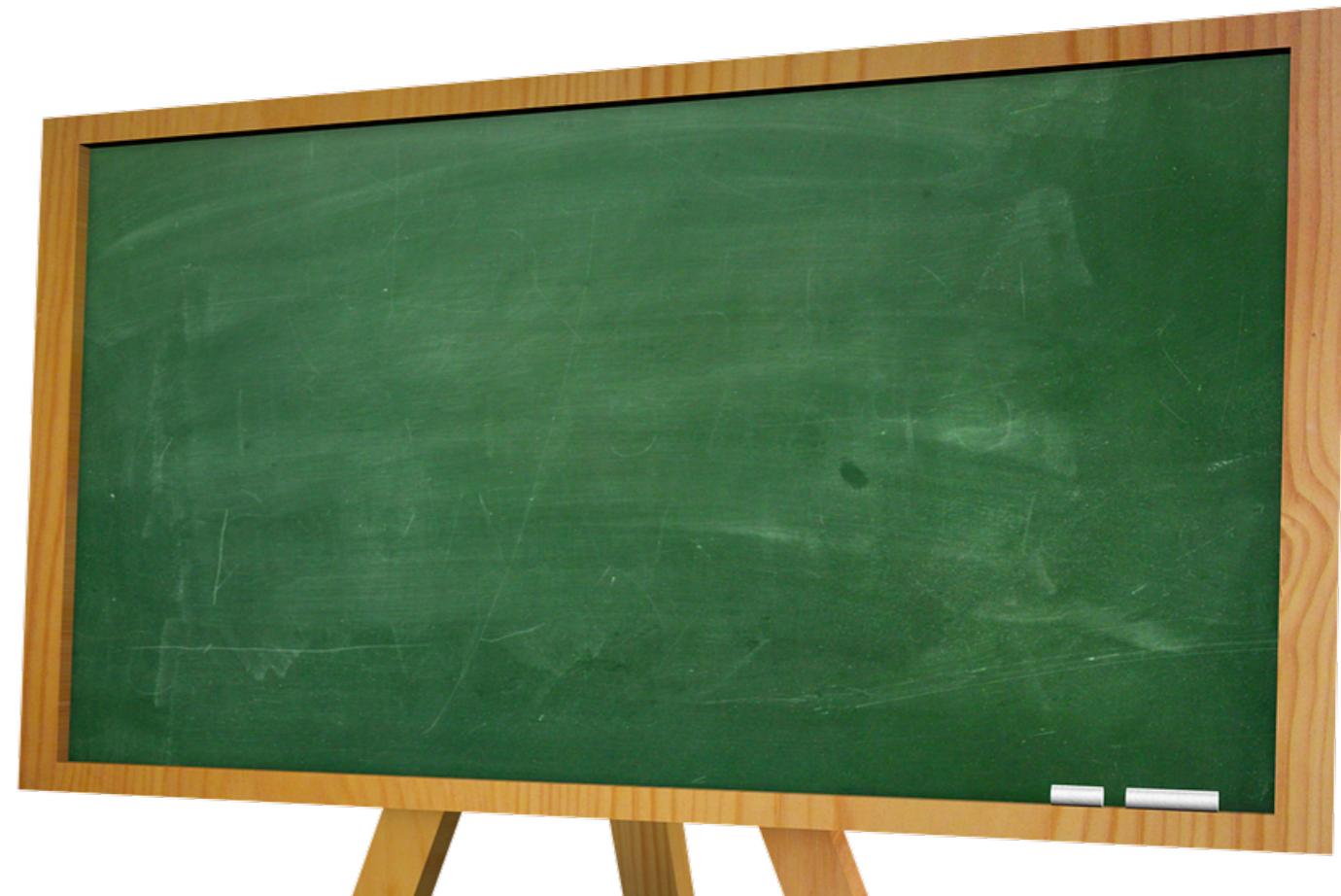
Paul Amman



René Just (UMass)



Teaching Software Testing Concepts with Code Defenders





Encode Software Testing Concepts as Puzzles



Encode Software Testing Concepts as Puzzles





Encode Software Testing Concepts as Puzzles





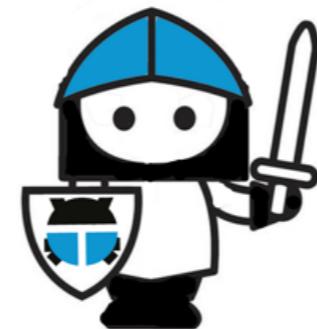
Statement Coverage

Defending puzzle



Statement Coverage

Defending puzzle





Statement Coverage

Defending puzzle

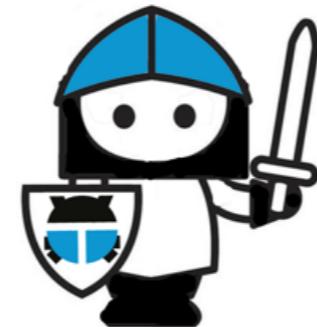
Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```



Statement Coverage

Defending puzzle



Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```



Statement Coverage

Defending puzzle

...and this **surviving mutant**

Given this **code under**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*3+7);
```



Statement Coverage

Defending puzzle



Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

...and this **surviving mutant**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*3+7);  
}
```



Statement Coverage

Defending puzzle

and this **test suite**

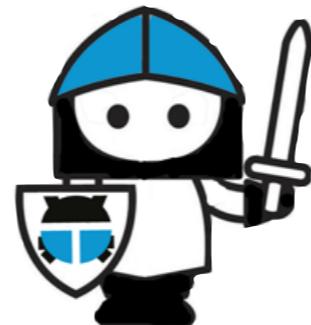
```
@Test  
void test() {  
    int x = 5;  
    assertEquals(3,m(x));  
}  
  
@Test  
void test() {  
    int m(int x)  
    if (int x =  
        return 3;  
    return (x*8  
}  
}
```

Given this **code** or



Statement Coverage

Defending puzzle



Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

...and this **surviving mutant**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*3+7);  
}
```

and this **test suite**

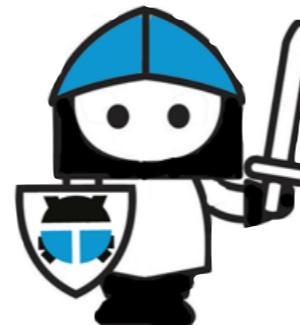
```
@Test  
void test() {  
    int x = 5;  
    assertEquals(3,m(x));  
}  
@Test  
void test() {  
}
```



Statement Coverage

Defending puzzle

create a **new test** that **detects** the mutant



and this **test suite**

Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

...and this **surviving mutant**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*3+7);  
}
```

```
@Test  
void test() {  
    int x = 5;  
    assertEquals(3,m(x));  
}  
@Test  
void test() {  
}
```



Statement Coverage

Defending puzzle

create a **new test** that **detects** the mutant

Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

...and this **surviving mutant**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*(3+7));  
}
```

and this **test suite**

```
@Test  
void test() {  
    int x = 5;  
    assertEquals(3,m(x));  
}  
@Test  
void test() {
```





Statement Coverage

Defending puzzle

create a **new test** that **detects** the mutant

Given this **code under test**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*8);  
}
```

...and this **surviving mutant**

```
int m(int x) {  
    if (int x == 5)  
        return 3;  
    return (x*(3+7));  
}
```

and this **test suite**

```
@Test  
void test() {  
    int x = 5;  
    assertEquals(3,m(x));  
}  
@Test  
void test() {  
    int x = 3;  
    assertEquals(24,m(x));  
}
```





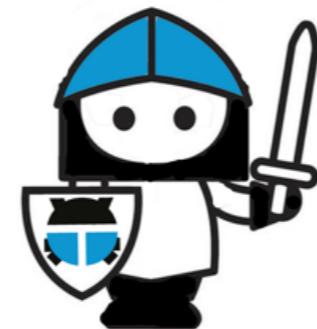
Branch Coverage

Attacking puzzle



Branch Coverage

Attacking puzzle





Branch Coverage

Attacking puzzle

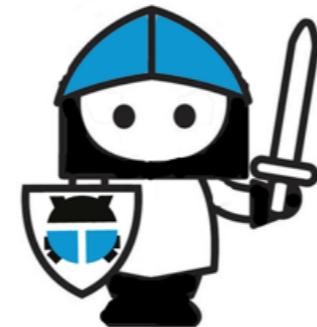
Given this **code under test**

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```



Branch Coverage

Attacking puzzle



Given this **code under test**

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```



Branch Coverage

Attacking puzzle

...and this **passing test**

Given this code

```
@Test  
public void test() {  
  
    int m(int x) {  
        int y = 1;  
        if (x == 1) {  
            y = 3;  
        }  
        return y;  
    }  
    assertEquals(24, m(x));  
}
```



Branch Coverage

Attacking puzzle



Given this **code under test**

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```

...and this **passing test**

```
@Test  
public void test() {  
    int x = 5;  
    assertEquals(24, m(x));  
}
```



Branch Coverage

Attacking puzzle

...create a surviving mutant.

Given this **code under test**:

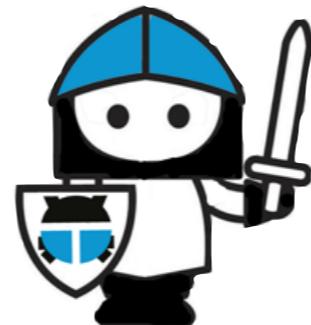
```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8)  
}
```

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```



Branch Coverage

Attacking puzzle



Given this **code under test**

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```

...and this **passing test**

```
@Test  
public void test() {  
    int x = 5;  
    assertEquals(24, m(x));  
}
```

...create a surviving mutant.

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```



Branch Coverage

Attacking puzzle

Given this **code under test**

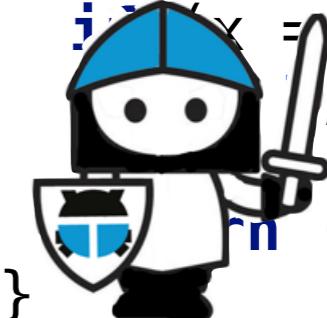
```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```

...and this **passing test**

```
@Test  
public void test() {  
    int x = 5;  
    assertEquals(24, m(x));  
}
```

...create a surviving mutant.

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```





Branch Coverage

Attacking puzzle

Given this **code under test**

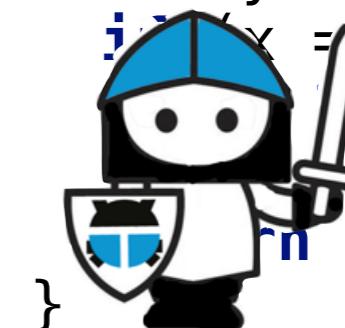
```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```

...and this **passing test**

```
@Test  
public void test() {  
    int x = 5;  
    assertEquals(24, m(x));  
}
```

...create a surviving mutant.

```
int m(int x) {  
    int y = 0;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```





Branch Coverage

Attacking puzzle



Given this **code under test**

```
int m(int x) {  
    int y = 1;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```

...and this **passing test**

```
@Test  
public void test() {  
    int x = 5;  
    assertEquals(24, m(x));  
}
```

...create a surviving mutant.

```
int m(int x) {  
    int y = 0;  
    if (x == 5) {  
        y = 3;  
    }  
    return (y * 8);  
}
```



Other Testing Concepts



Other Testing Concepts

Testing loops



Other Testing Concepts

Testing loops

Data-flow testing



Other Testing Concepts

Testing loops

Data-flow testing

Boundary value analysis



Other Testing Concepts

Testing loops

Data-flow testing

Boundary value analysis

Model-based testing



Other Testing Concepts

Testing loops

Data-flow testing

Boundary value analysis

Model-based testing

Combinatorial testing



Other Testing Concepts

Testing loops

Data-flow testing

Boundary value analysis

Model-based testing

Combinatorial testing

Mutation testing



Related Work





Related Work



When Thursday 14:00-15:30

Session Testing II

Room Libertador C



Related Work



When Thursday 14:00-15:30

Session Testing II

Room Libertador C



Code Defenders is Fun



Code Defenders is Fun

I enjoyed playing Code Defenders



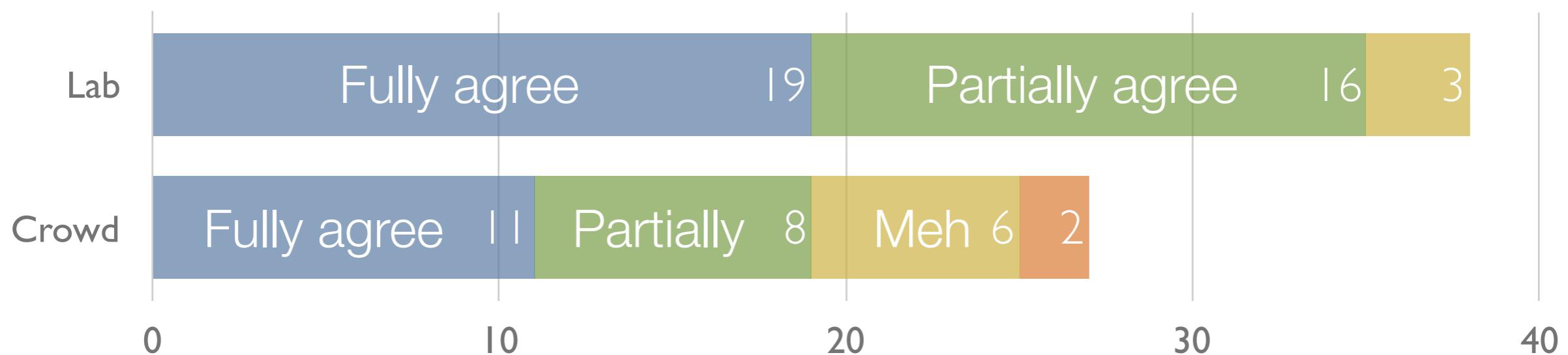


Code Defenders is Fun

I enjoyed playing Code Defenders

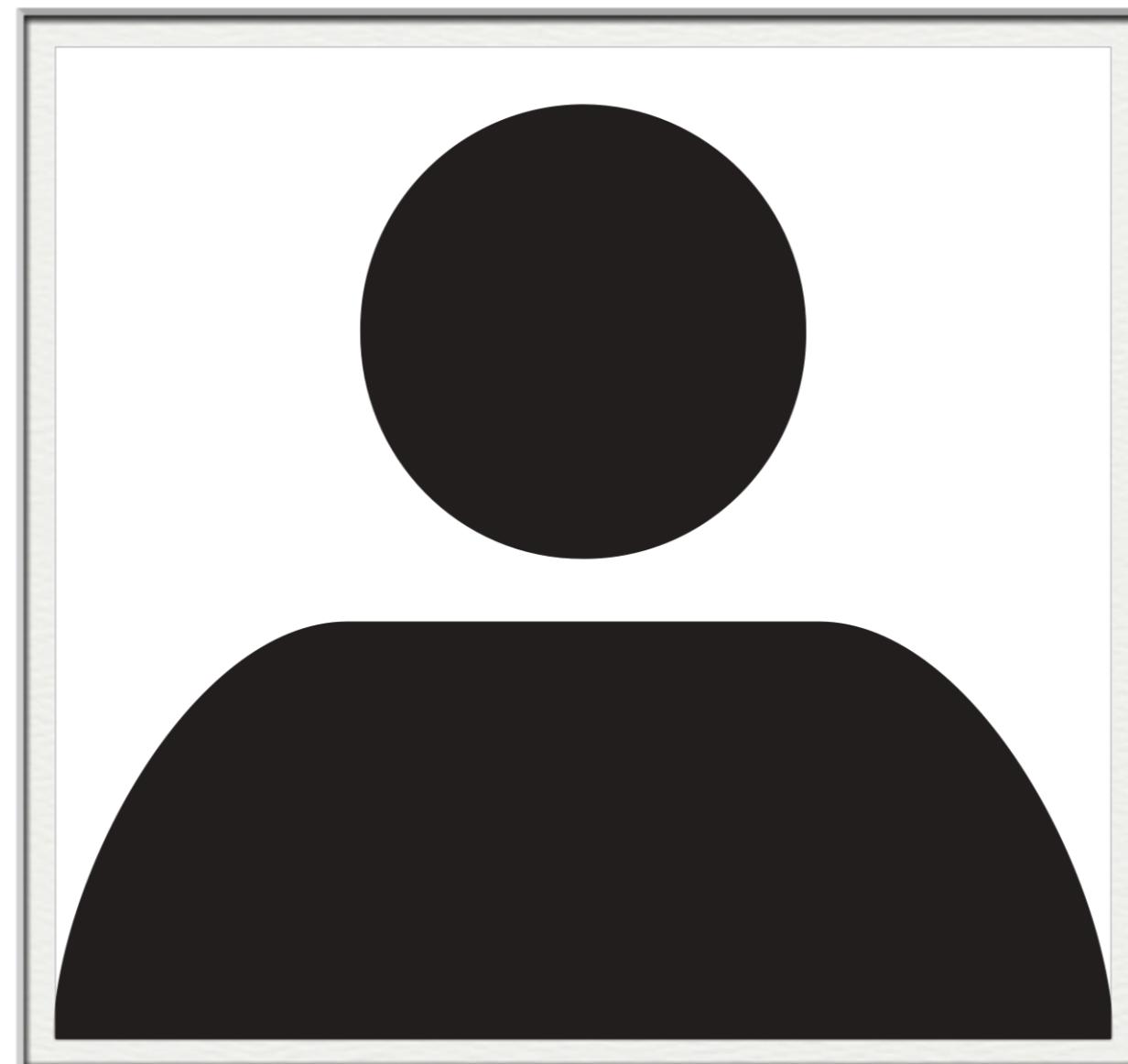


Playing Code Defenders is more fun than manual testing





Early Adopters



You?



Code Defenders - Game 2260 × code-defenders.org/multiplayer/play?id=2260

Code Defenders

DEFENDER::ACTIVE SparseIntArray

Class Under Test

```
50 /**
51 * Creates a new SparseIntArray containing no mappings that will not
52 * require any additional memory allocation to store the specified
53 * number of mappings. If you supply an initial capacity of 0, the
54 * sparse array will be initialized with a light-weight representation
55 * not requiring any additional array allocations.
56 */
57 public SparseIntArray(int initialCapacity) {
58     if (initialCapacity == 0) {
59         mKeys = SparseIntArray.EMPTY_INT_ARRAY;
60         mValues = SparseIntArray.EMPTY_INT_ARRAY;
61     } else {
62         mKeys = new int[initialCapacity];
63         mValues = new int[mKeys.length];
64     }
65     mSize = 0;
66 }
67 /**
68 * Given the current size of an array, returns an ideal size to which the array should
69 * grow. This is typically double the given size, but should not be relied upon to do so in the
70 * future.
71 */
72 public static int growSize(int currentSize) {
73     return currentSize <= 4 ? 8 : currentSize + (currentSize >> 1);
74 }
```

Write a new JUnit test here

Show Scoreboard

Defend!

Existing Mutants

alive (35)	killed(68)	equivalent(5)
Mutant 5838 Creator: amin [UID: 428]	Modified line 244	Claim Equivalent
Mutant 5855 Creator: abrahamc2 [UID: 432]	Modified line 211	Claim Equivalent
Mutant 5823 Creator: gregoryg [UID: 426]	Modified line 178	Claim Equivalent
Mutant 5886 Creator: gregoryg [UID: 426]	Modified line 190	Claim Equivalent

JUnit tests

Test 4034 | Creator: alessiogambi [UID: 433]

```
1 /* no package name */
2
3 import org.junit.*;
4 import static org.junit.Assert.*;
5
6 public class TestSparseIntArray {
7     @Test(timeout = 4000)
8     public void test() throws Throwable {
9         SparseIntArray sia = new SparseIntArray(0);
10        sia.put(0, 7);
11        sia.put(1, 8);
12        sia.delete(0);
13        assertEquals(1, sia.size());
14        assertEquals(8, sia.get(1));
15    }
16 }
```

code-defenders.org