Code Defenders

Teaching Software Testing Concepts using a Mutation Testing Game

Benjamin Clegg, **José Miguel Rojas**, Gordon Fraser

Class Under Test

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

Attack!

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

Defend!

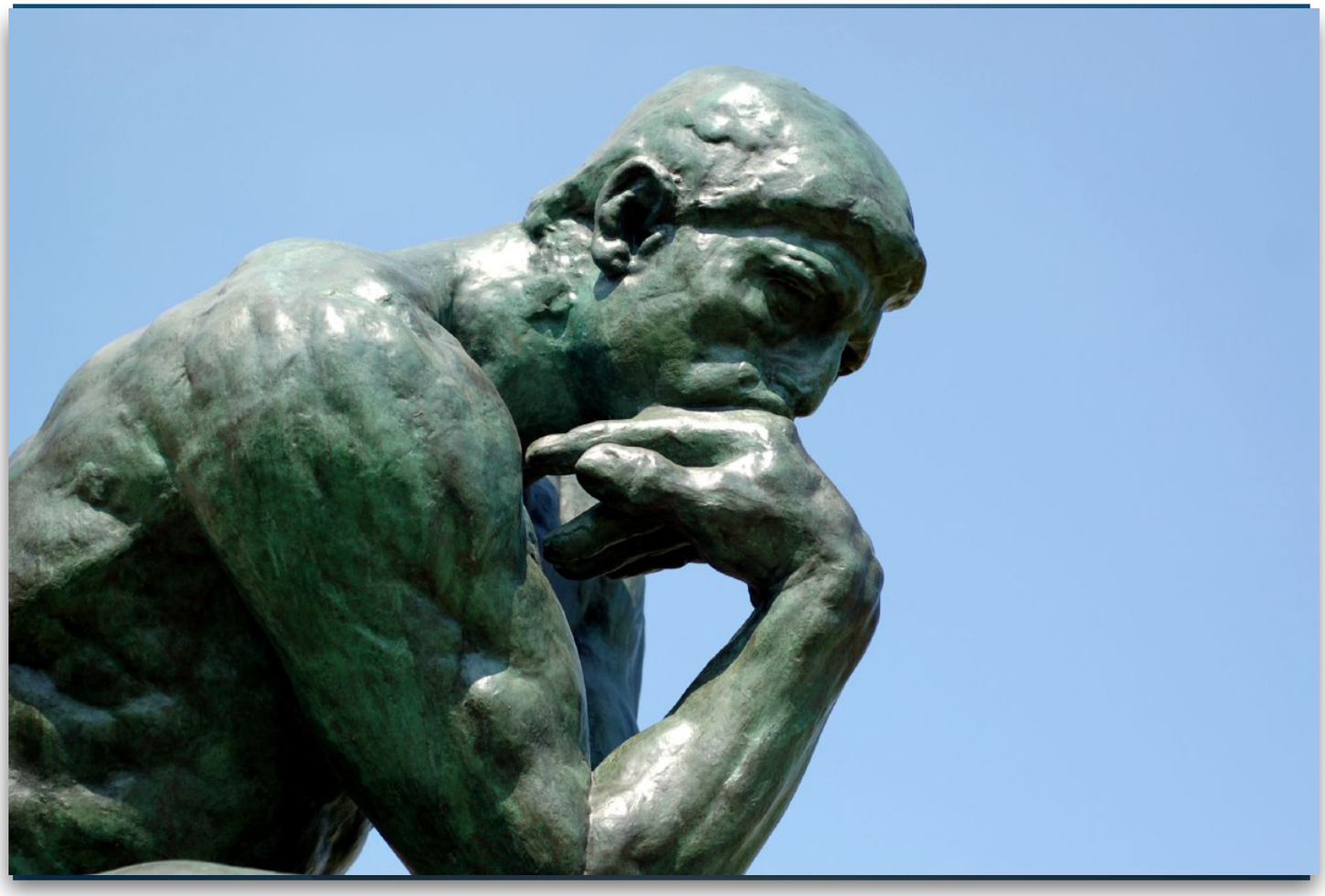
```
public class TestArithmetics {
   @Test
   public void testAbs() {
      Arithmetics a;
      a = new Arithmetics();
      assertEquals(1, a.abs(-1));
   }
}
```

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

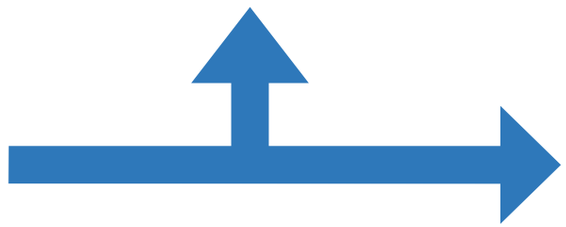
Defenders

Code Defenders

Class Under Test

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

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



Attackers

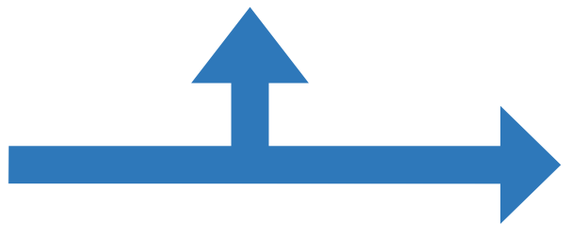
Defenders

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;
   }
}
```



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

Attackers

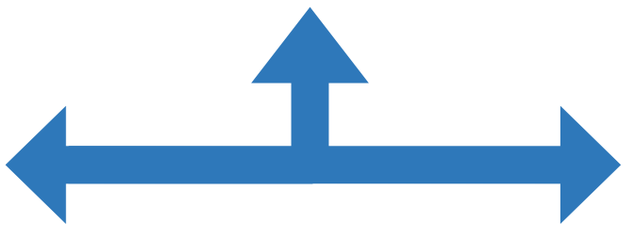
Defenders

Code Defenders

Class Under Test

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

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



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

Attackers

Defenders

Code Defenders

Class Under Test

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

Score points for surviving mutants

```
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;
   }
}
```

Score points for surviving mutants

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

Attackers

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

Class Under Test

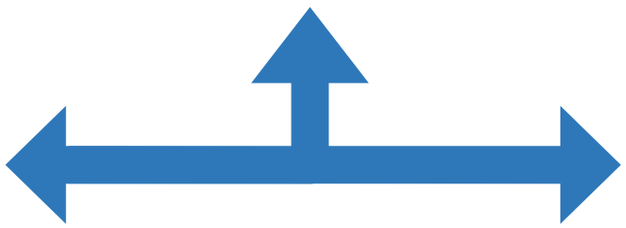
```
public class Arithmetics {
   public int abs(int x) {
      if (x >= 0)
         return x;
      else
         return -x;
   }
}
```

Score points for surviving mutants

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

Score points for effective tests

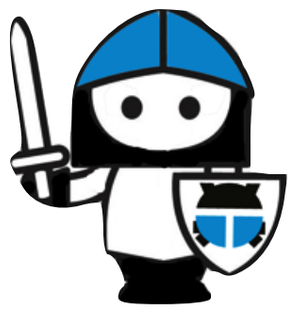
```
public class TestArithmetics {
   @Test
   public void testAbs() {
      Arithmetics a;
      a = new Arithmetics();
      assertEquals(1, a.abs(-1));
   }
}
```



Attackers

Equivalent Mutant Duels

Defenders



Code Defenders

Class Under Test

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

Score points for surviving mutants

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

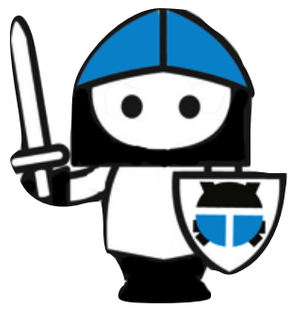
Score points for effective tests

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

equivalent!

Attackers

Defenders



Equivalent Mutant Duels

Code Defenders

Class Under Test

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

Score points for surviving mutants

```
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

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

Score points for surviving mutants

```
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

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

Score points for surviving mutants

```
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!

oh no! :(

equivalent!

Attackers

Defenders

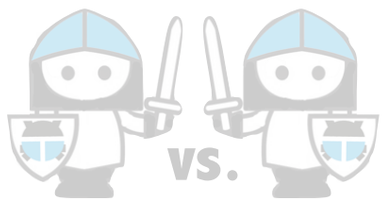
Equivalent Mutant Duels

Code Defenders

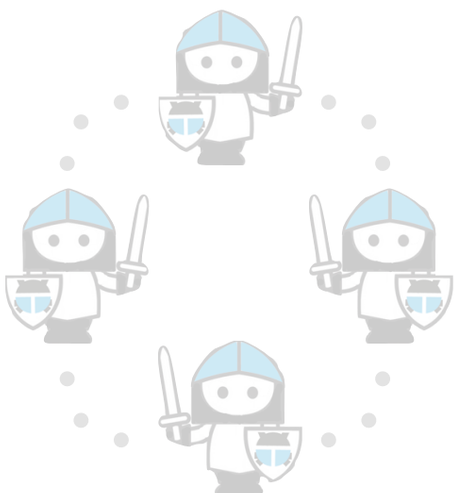
Single-player

vs.

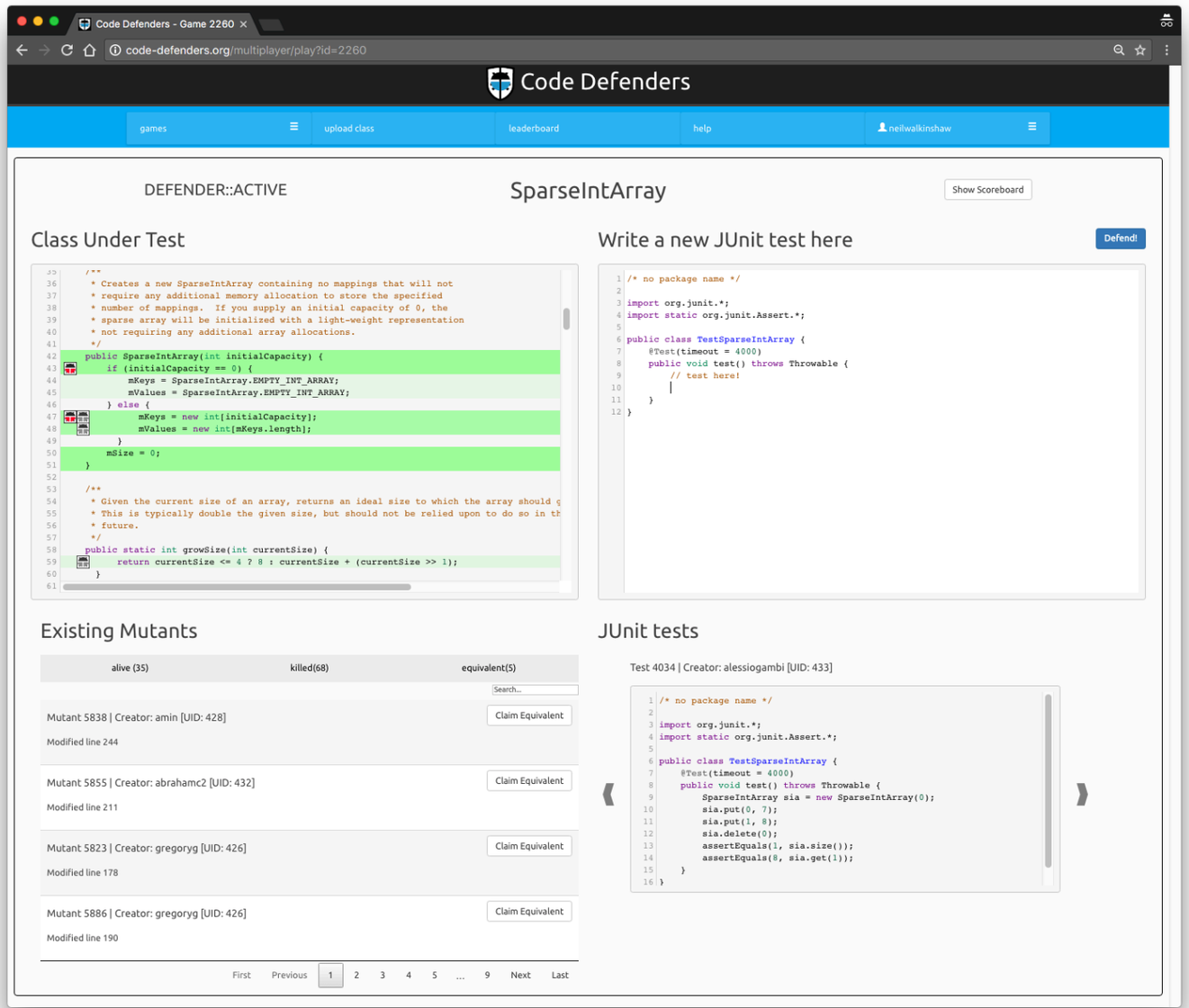
EVOSUITE / MAJOR



Duels



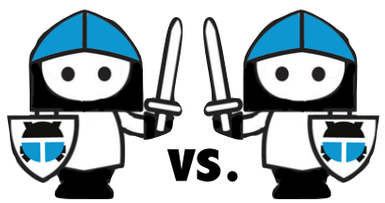
Multi-player



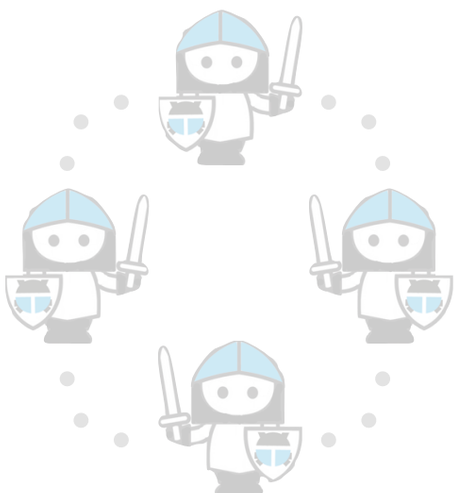
Code Defenders

vs.

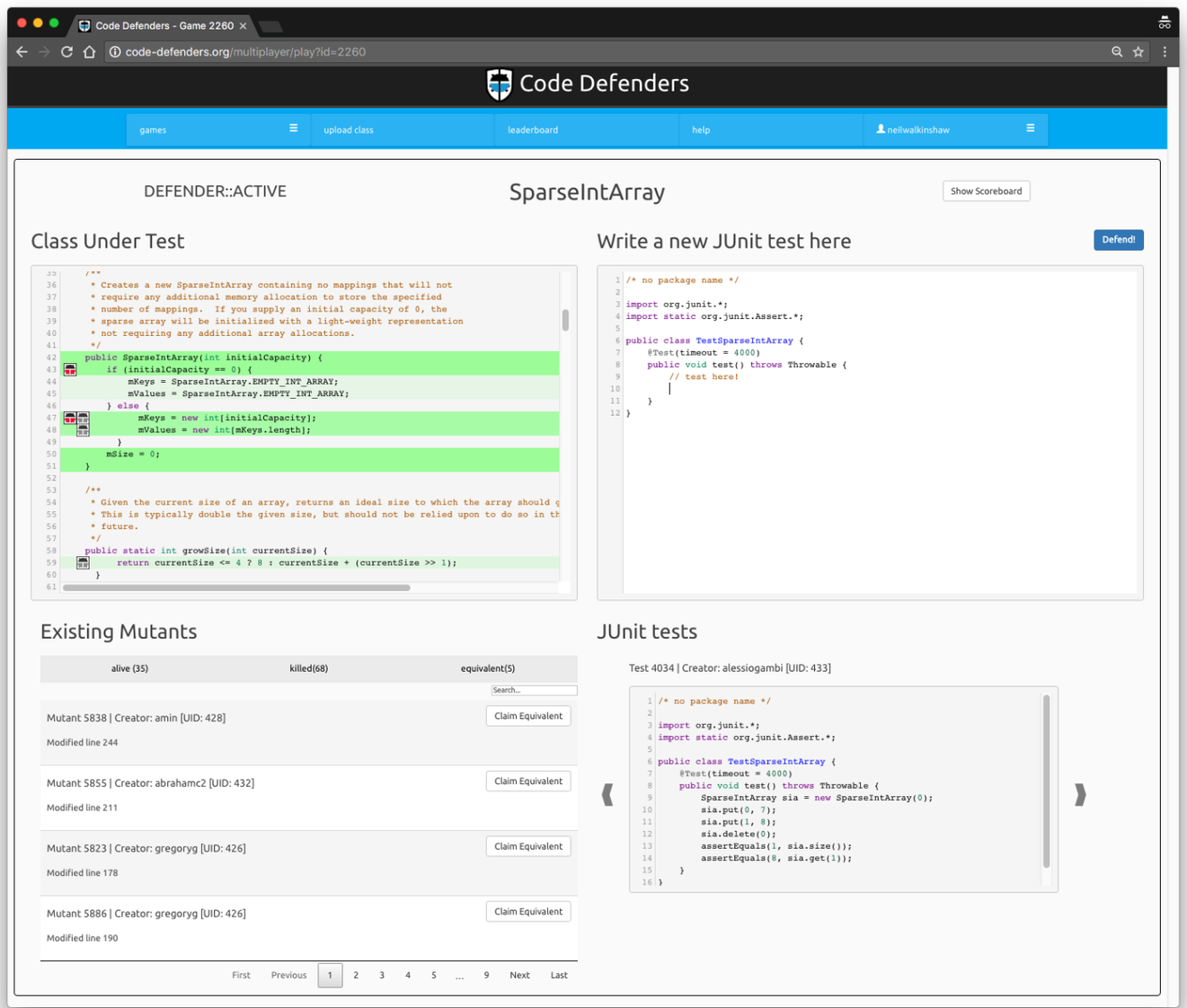
EVOSUITE / MAJOR



Duels



Multi-player

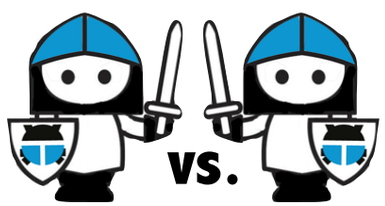


Code Defenders

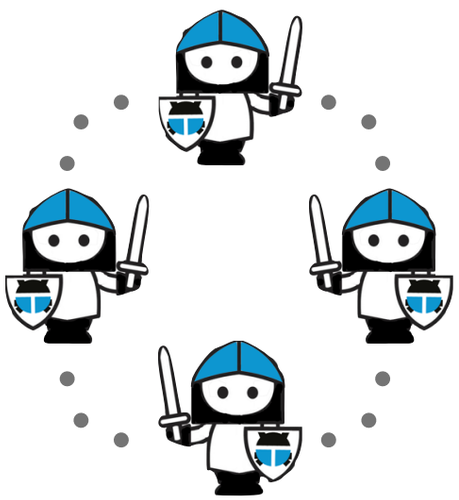
vs.

EVOSUITE / MAJOR

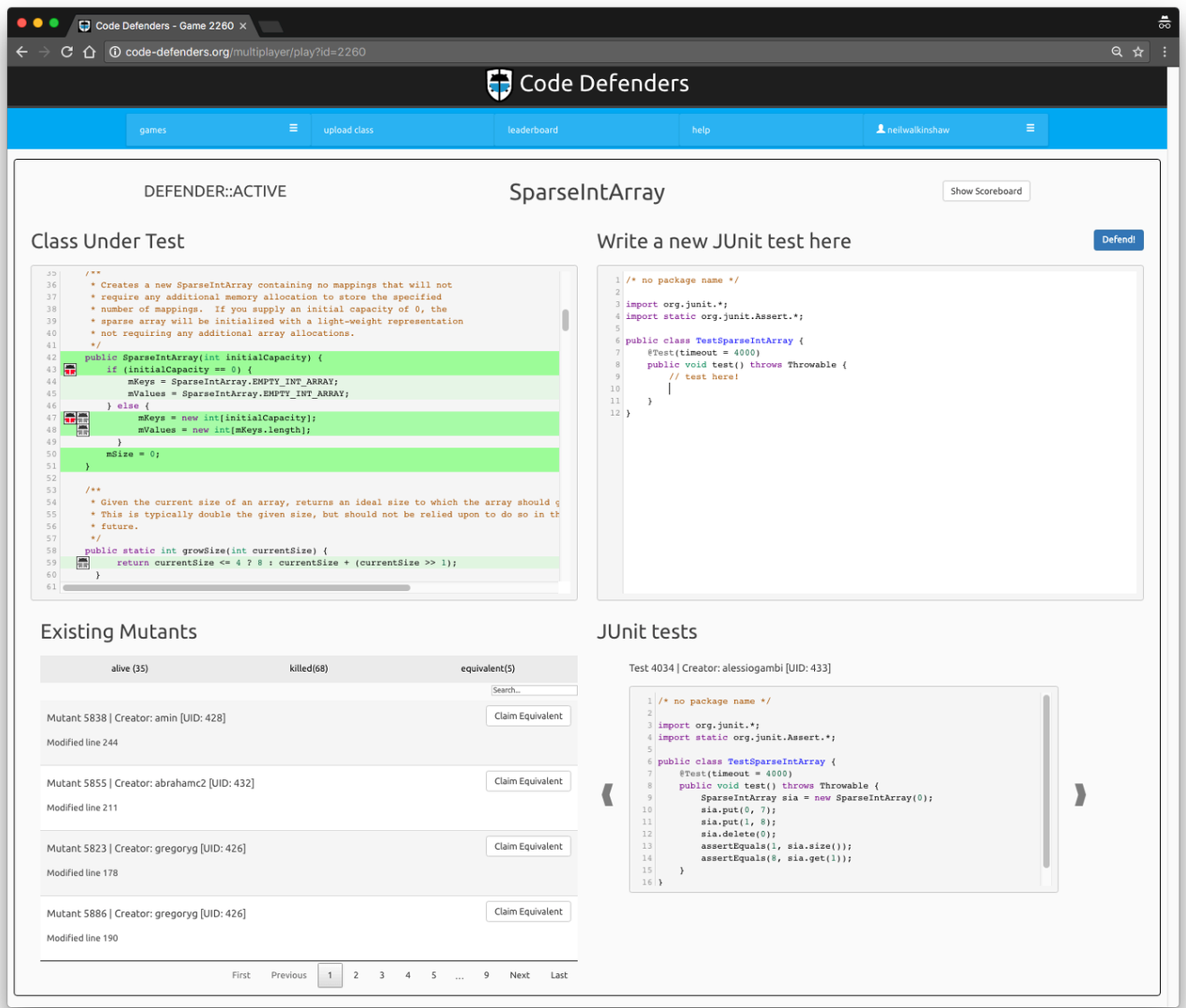
Single-player



Duels



Multi-player

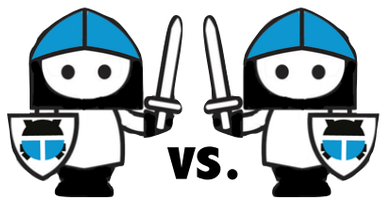


Code Defenders

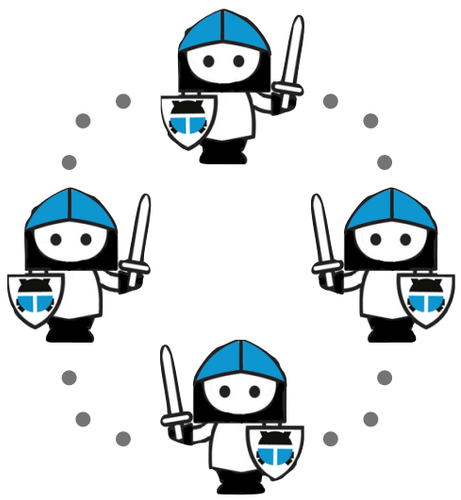
Single-player

vs.

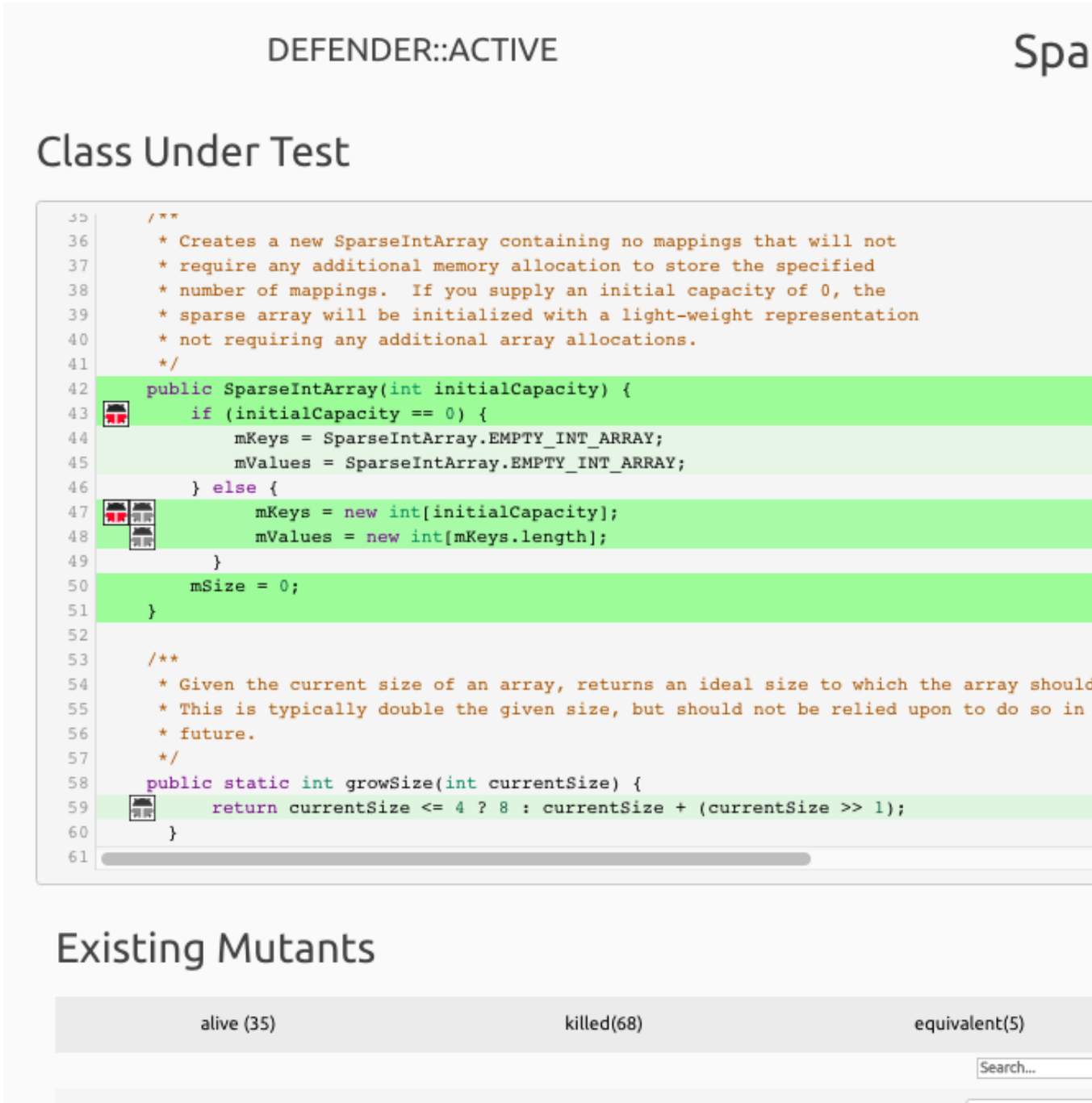
EVOSUITE / MAJOR



Duels



Multi-player



Early Adopters

Early Adopters

Paul Amman

Early Adopters

Paul Amman

Jeff Offutt

Early Adopters

Jeff Offutt

Paul Amman

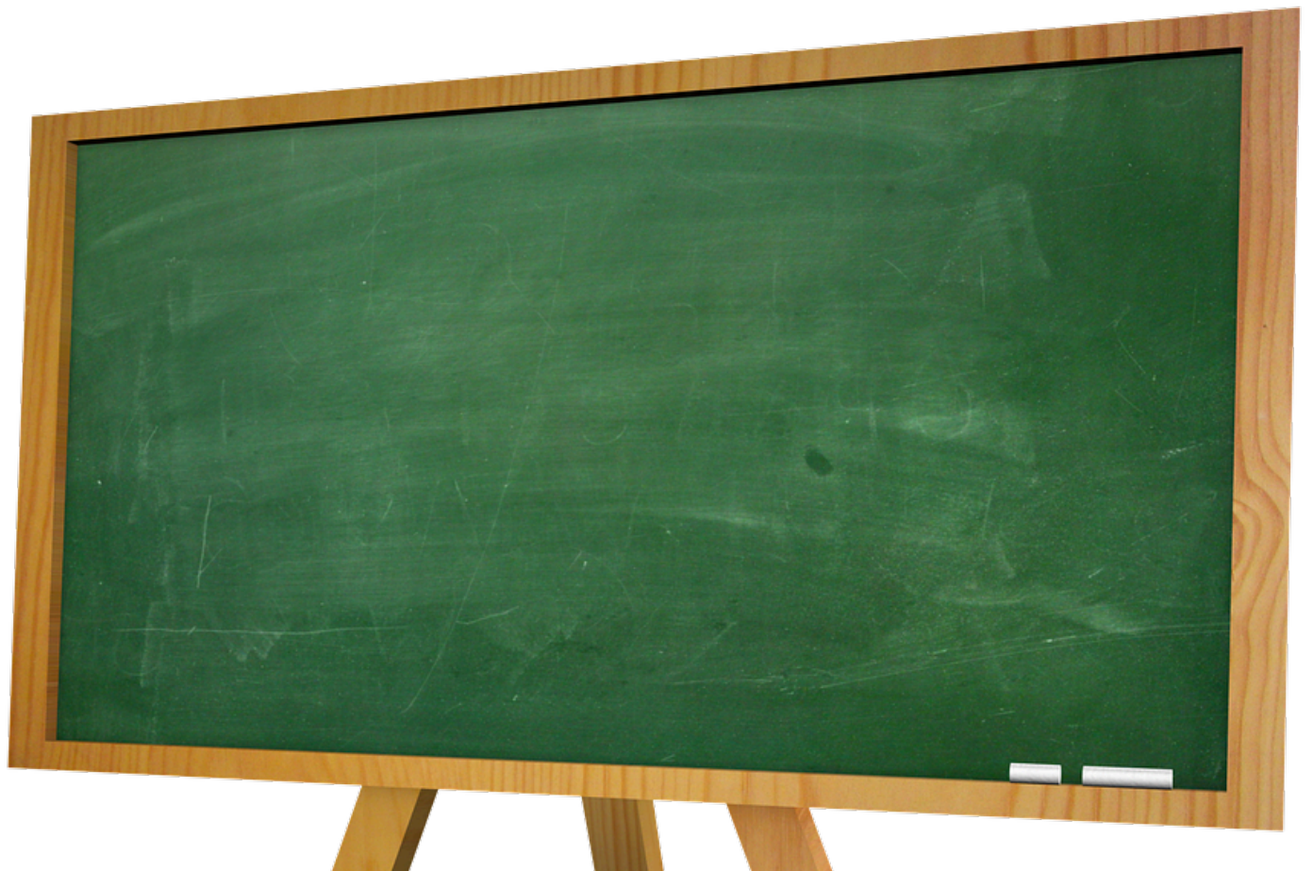
Early Adopters

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**

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

Given this **code under**

```
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);
}
```

…and this **surviving mutant**

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

Statement Coverage

Defending puzzle

Given this **code ur**

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

and this **test suite**

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


}
```

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

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() {
```

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**

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

Given this **code**

```
int m(int
   int y =
   if (x =
      y = 3
   }
   return
}
```

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 te**

```
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**

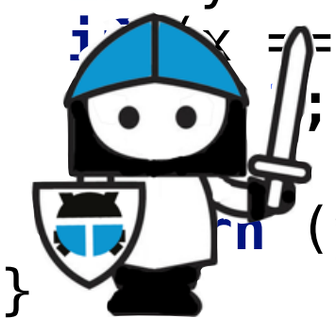
```
int m(int x) {
  int y = 1;
  if (x == 5) {
    y = 3;
  }
  return (y * 8);
}
```

…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;
  i(x == 5) {
    ;
  rn (y * 8);
}
```



Branch Coverage

Attacking puzzle

Given this **code under test**

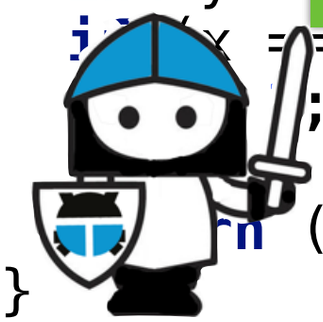
```
int m(int x) {
  int y = 1;
  if (x == 5) {
    y = 3;
  }
  return (y * 8);
}
```

…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

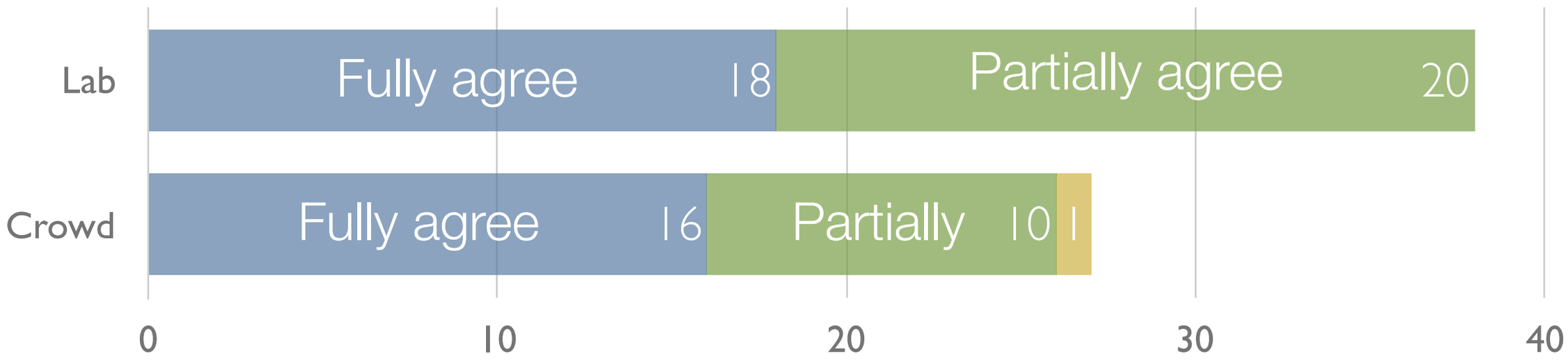
When Thursday 14:00-15:30

Session Testing II

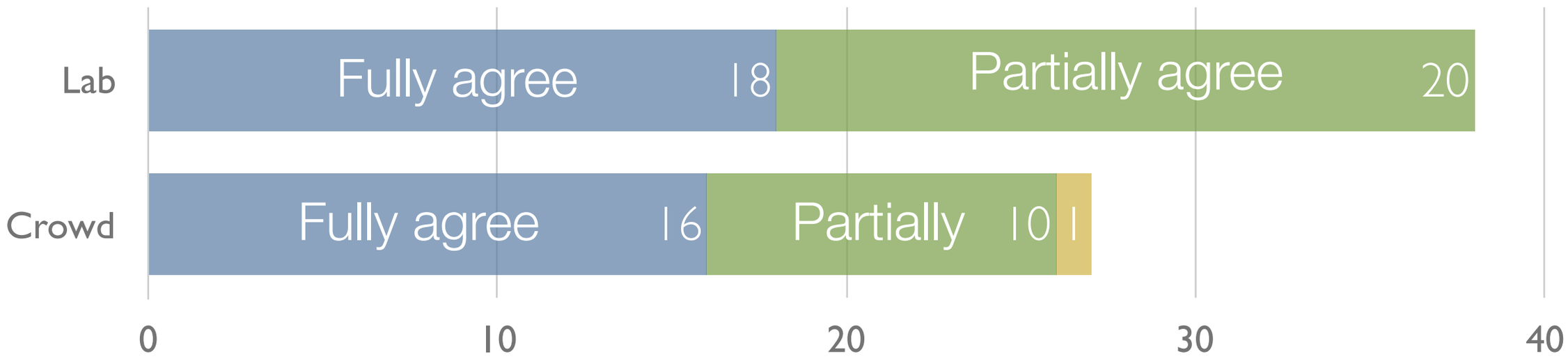
Room Libertador C

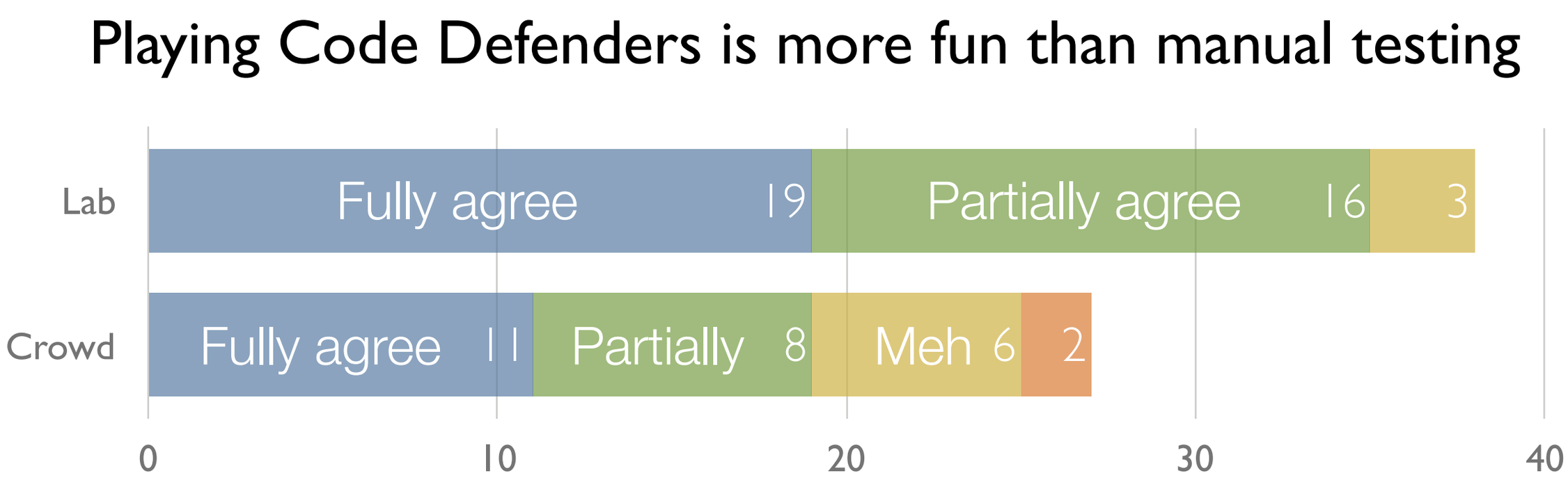
Code Defenders is Fun

Code Defenders is Fun

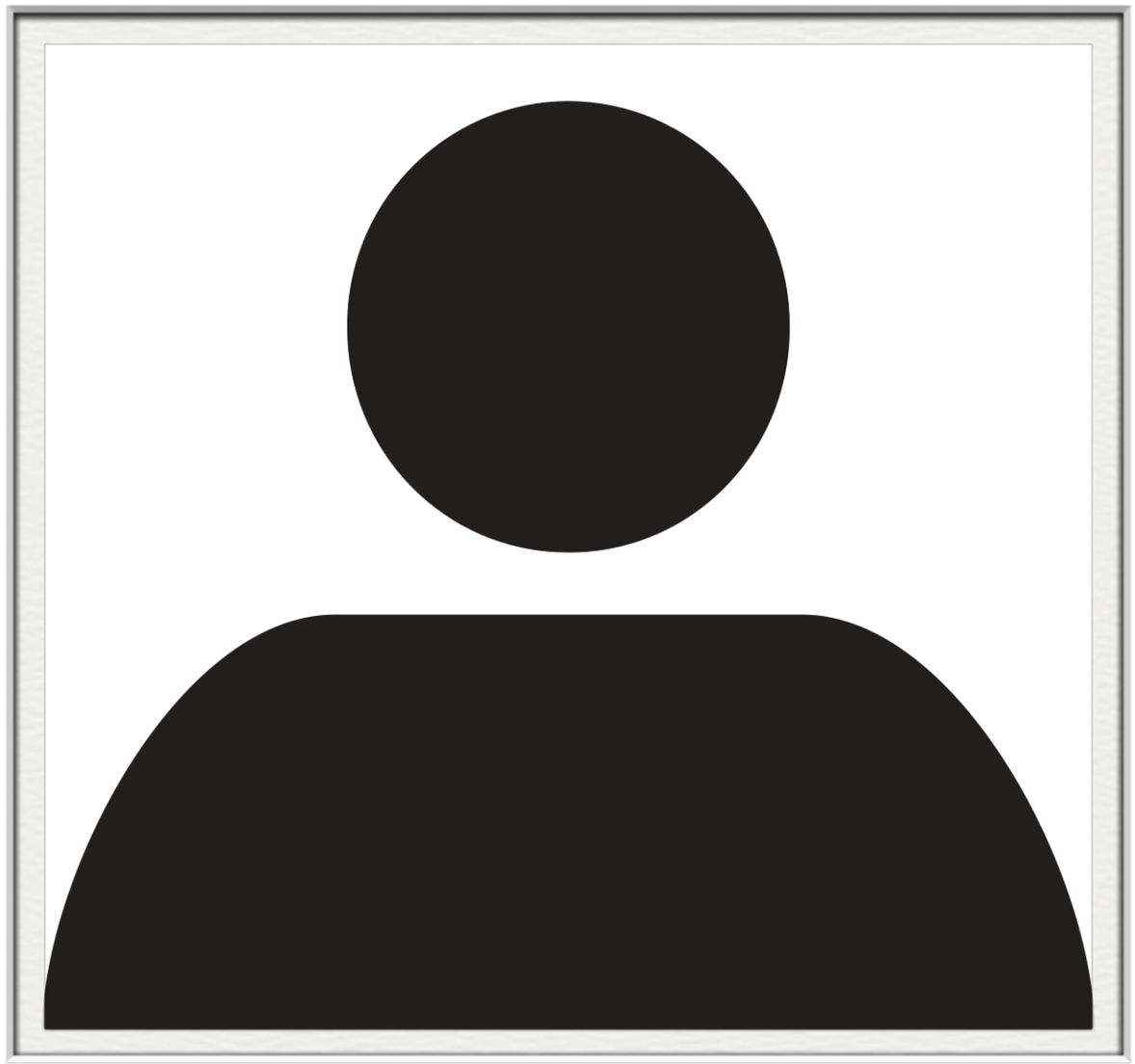


Code Defenders is Fun

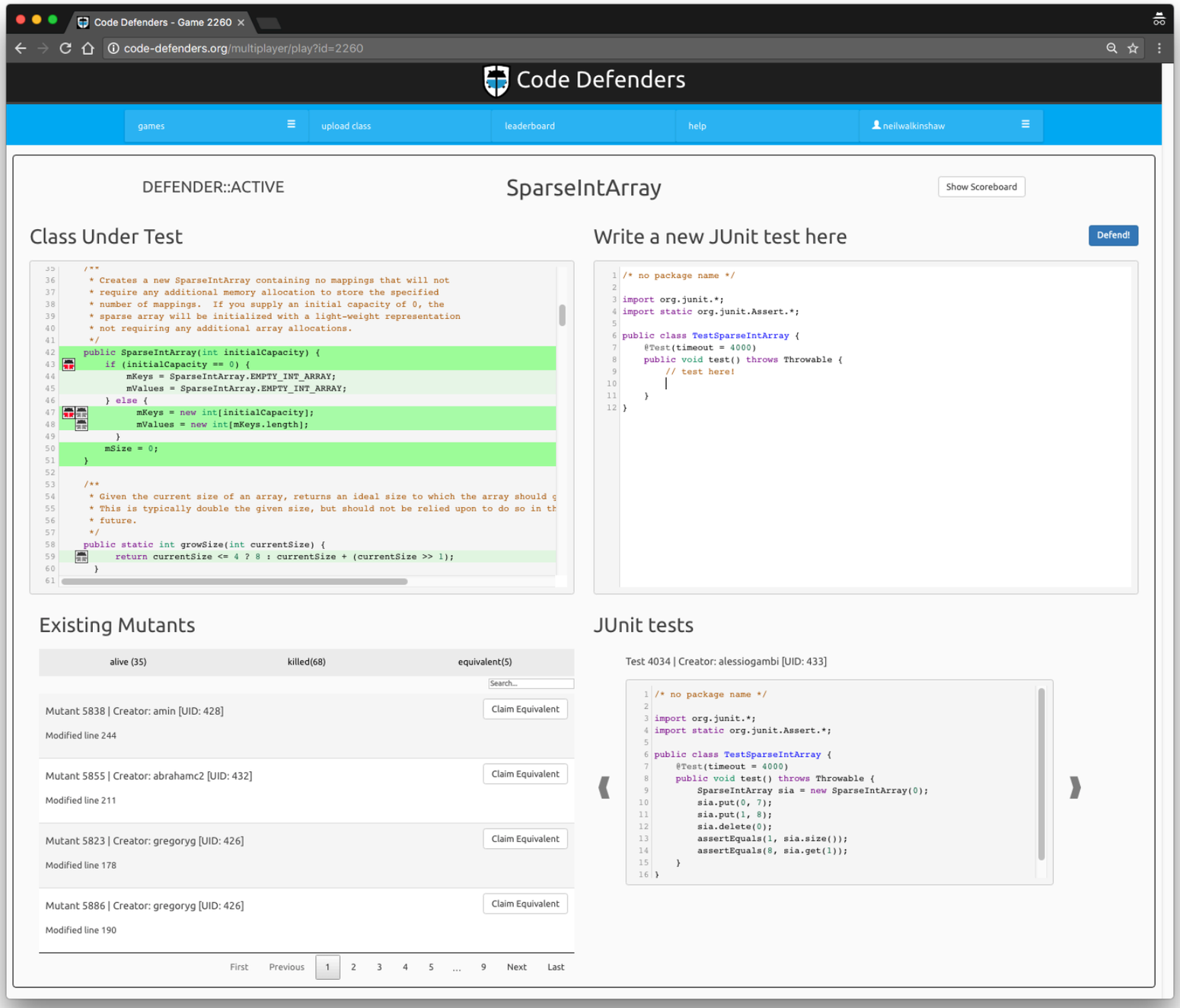




Early Adopters



You?



code-defenders.org