Code Defenders

A Mutation Testing Game

José Miguel Rojas

Gameplay

Class Under Test

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

Gameplay

Class Under Test

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

Attackers

Gameplay

Class Under Test

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

Attackers

Defenders

Gameplay

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

Gameplay

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

Gameplay

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

Gameplay

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

Gameplay

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

Gameplay

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

Gameplay

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

Gameplay

Class Under Test

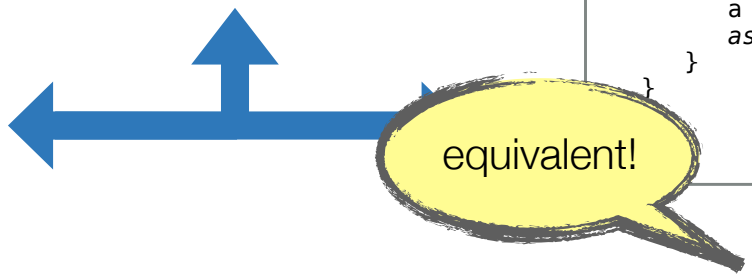
```
public class Arithmetics {
   public int abs(int x) {
      if (x >= 0)
         return x;
      else
         return -x;
   }
}
```

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

Gameplay

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

Equivalent Mutant Duels

Defenders

Gameplay

Class Under Test

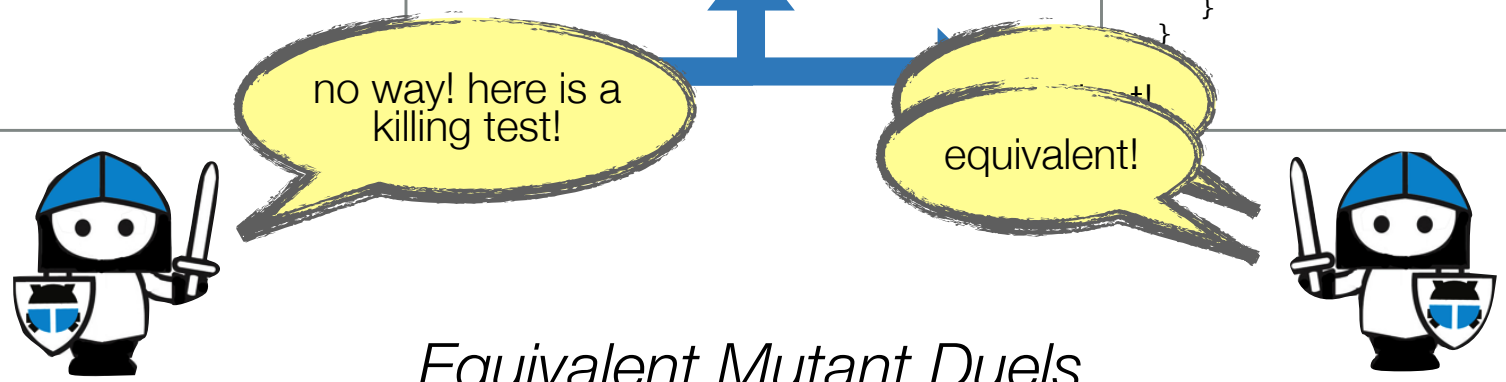
```
public class Arithmetics {
   public int abs(int x) {
      if (x >= 0)
         return x;
      else
         return -x;
   }
}
```

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

Gameplay

Class Under Test

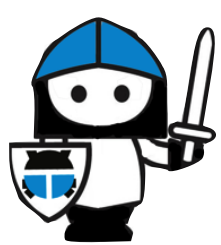
```
public class Arithmetics {
   public int abs(int x) {
      if (x >= 0)
         return x;
      else
         return -x;
   }
}
```

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! :(

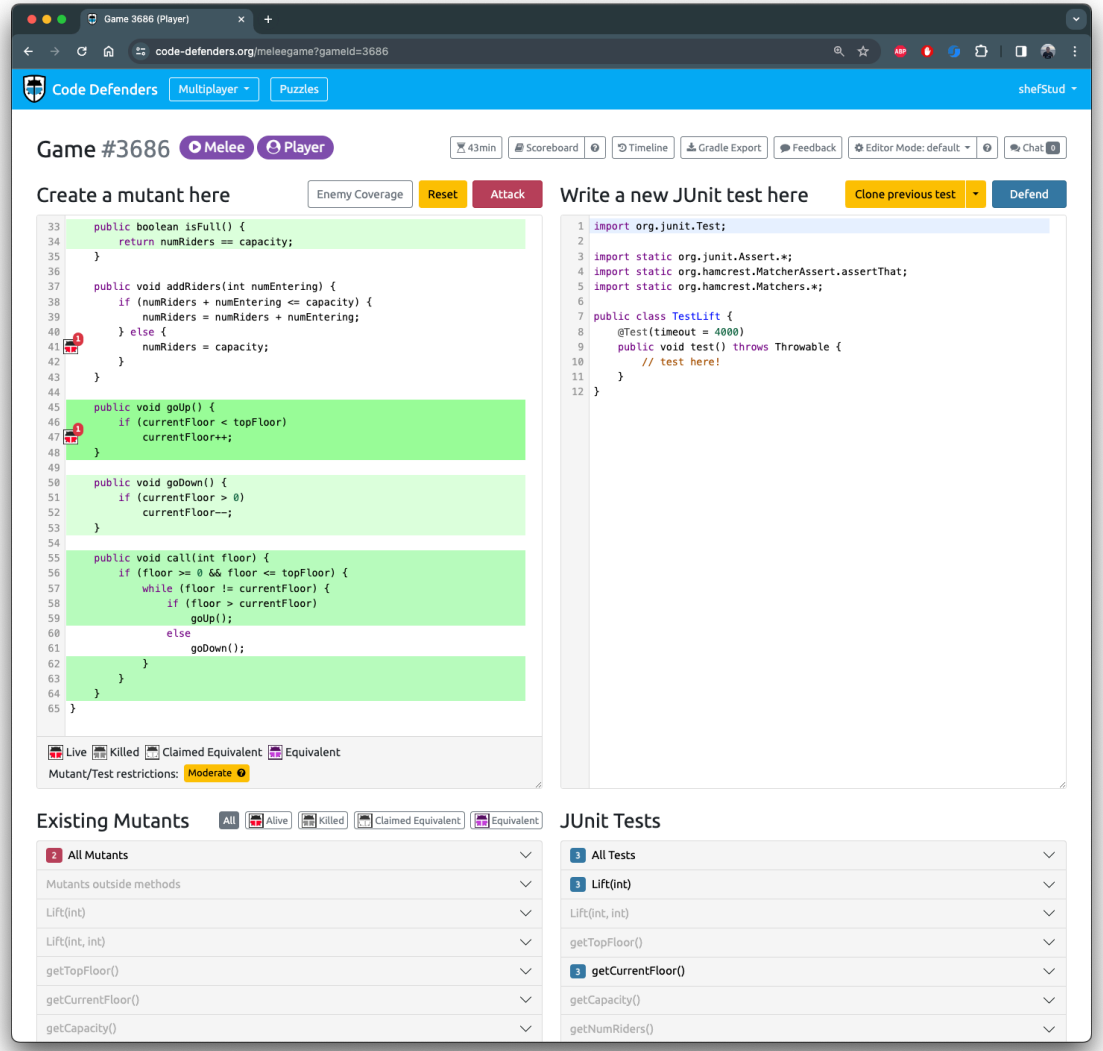
equivalent!

Attackers

Equivalent Mutant Duels

Defenders

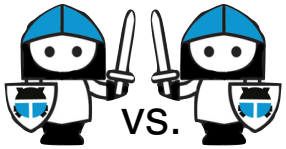
Play Modes



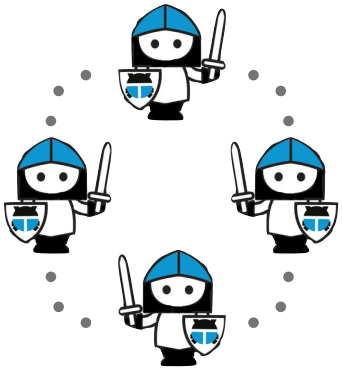
Single-player

vs.

EVOSUITE / MAJOR



Two-player



Multi-player

Code Coverage and Mutants

```
37      public void addRiders(int numEntering) {
38          if (numRiders + numEntering <= capacity) {
39              numRiders = numRiders + numEntering;
40          } else {
41              numRiders = capacity;
42          }
43      }
44
45      public void goUp() {
46          if (currentFloor < topFloor)
47              currentFloor++;
48      }
49
50      public void goDown() {
51          if (currentFloor > 0)
52              currentFloor--;
53      }
54
55      public void call(int floor) {
56          if (floor >= 0 && floor <= topFloor) {
57              while (floor != currentFloor) {
58                  if (floor > currentFloor)
59                      goUp();
60                  else
61                      goDown();
```

Code Coverage and Mutants

```
37      public void addRiders(int numEntering) {
38          if (numRiders + numEntering <= capacity) {
39              numRiders = numRiders + numEntering;
40          } else {
41              numRiders = capacity;
42          }
43      }
44
45      public void goUp() {
46          if (currentFloor < topFloor)
47              currentFloor++;
48      }
49
50      public void goDown() {
51          if (currentFloor > 0)
52              currentFloor--;
53      }
54
55      public void call(int floor) {
56          if (floor >= 0 && floor <= topFloor) {
57              while (floor != currentFloor) {
58                  if (floor > currentFloor)
59                      goUp();
60                  else
61                      goDown();
```

The greener the line, the more tests that cover it!

Code Coverage and Mutants

```
37      public void addRiders(int numEntering) {
38          if (numRiders + numEntering <= capacity) {
39              numRiders = numRiders + numEntering;
40
41
42
43      }
44
45      public void goUp() {
46          if (currentFloor < topFloor)
47              currentFloor++;
48      }
49
50      public void goDown() {
51          if (currentFloor > 0)
52              currentFloor--;
53      }
54
55      public void call(int floor) {
56          if (floor >= 0 && floor <= topFloor) {
57              while (floor != currentFloor) {
58                  if (floor > currentFloor)
59                      goUp();
60                  else
61                      goDown();
```

Mutant markers:

Live | Killed | Claimed Equivalent | Equivalent

The greener the line, the more tests that cover it!

Scoreboard

Scoreboard

7 3

Attackers	Mutants	Alive / Killed / Equivalent	Duels Won / Lost / Ongoing	Total Points
user2	4	3 / 1 / 0	0 / 0 / 1	7
Attacking Team	4	3 / 1 / 0	0 / 0 / 1	7
Defenders	**Tests**	**Mutants Killed**	**Duels Won / Lost / Ongoing**	**Total Points**
user1	2	1	0 / 0 / 1	3
Defending Team	2	1	0 / 0 / 1	3

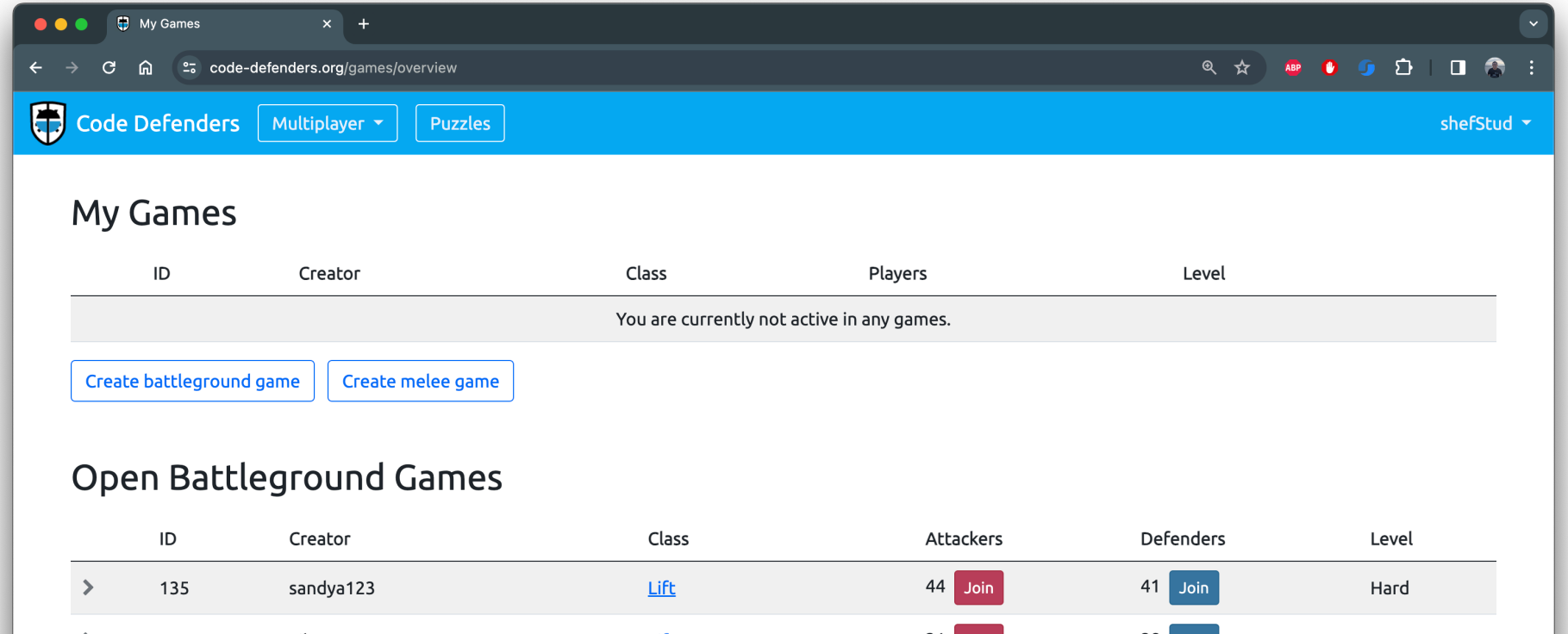
Close

Today's Practical – Sign up

Go to **https://code-defenders.org/**

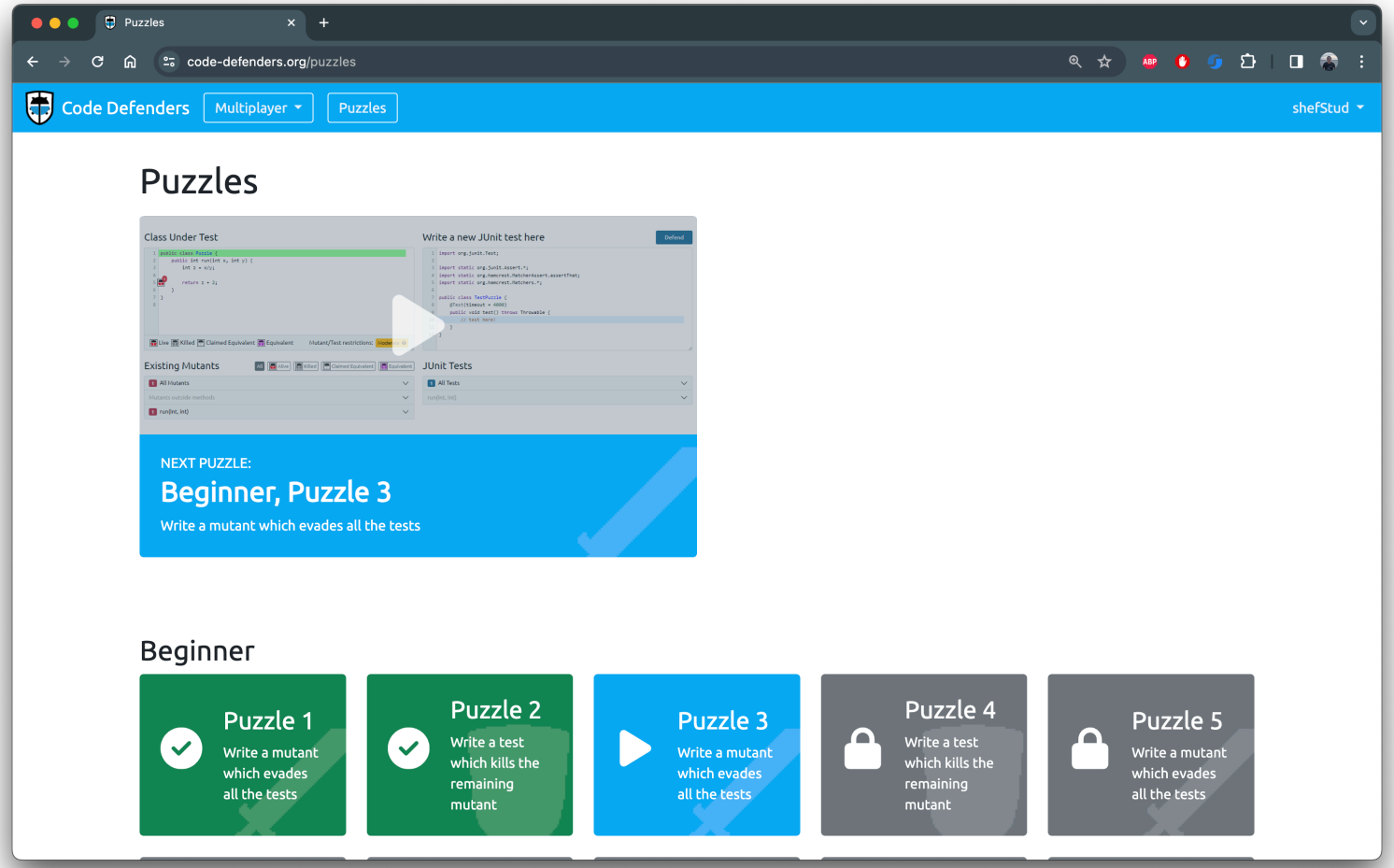
Click on **Log in or Sign up** to create an account and sign in

You should then see this page:



Today's Practical – Puzzles

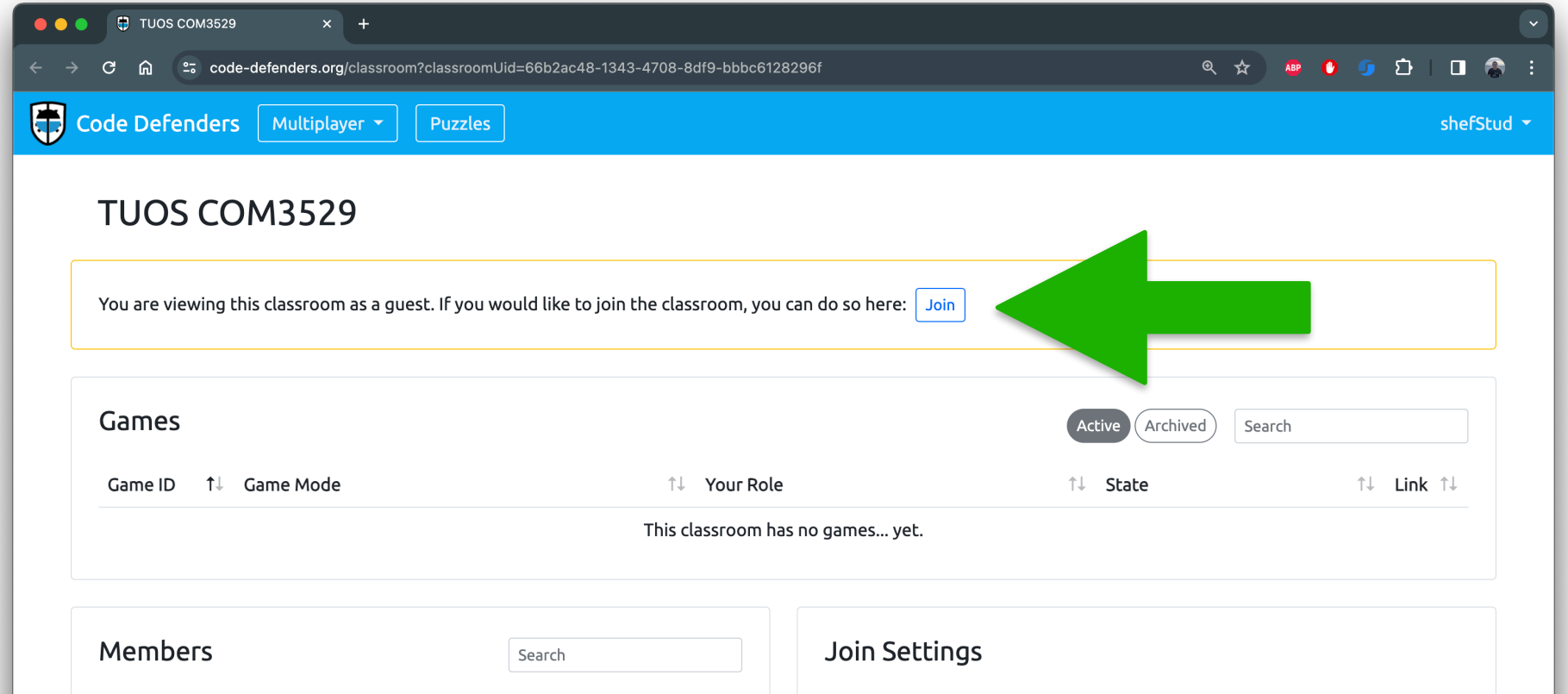
Next, go to **Puzzles** and complete **as many puzzles as you can in 15 minutes** (puzzle tiles turn green when completed):



Today's Practical – Classroom

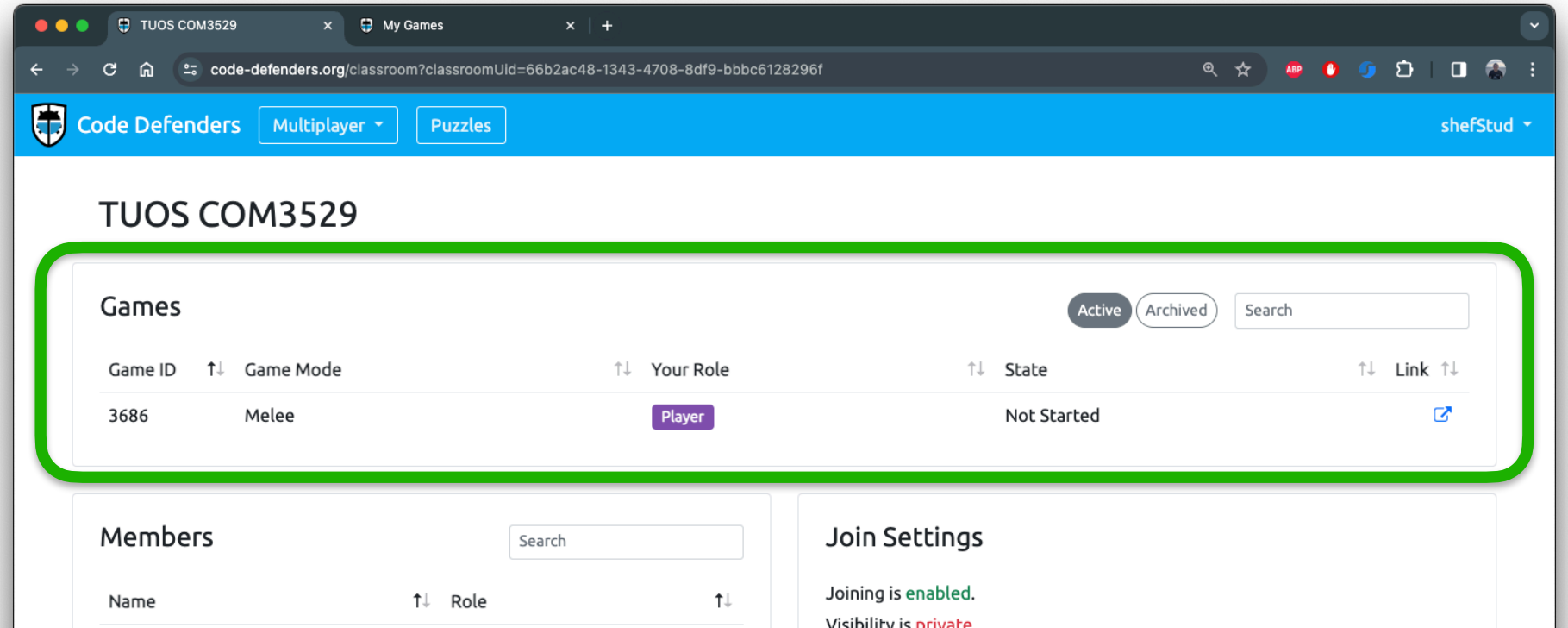
Next, join the Code Defenders classroom at:

https://bit.ly/COM3529-24-CodeDefenders



Today's Practical – Classroom

Once you have joined the classroom, wait to be added to a game and click on the game link when its state changes to **Running: XXmin left**.



Today's Practical – Game on!

Game 3686 (Player)

code-defenders.org/meleegame?gameId=3686

Code Defenders | Multiplayer | Puzzles | shefStud

Game #3686 Melee Player

1h | Scoreboard | Timeline | Gradle Export | Feedback | Editor Mode: default | Chat 0

Create a mutant here

Enemy Coverage | Reset | Attack

```java
public class Lift {

    private int topFloor;
    private int currentFloor = 0; // default
    private int capacity = 10;    // default
    private int numRiders = 0;    // default

    public Lift(int highestFloor) {
        topFloor = highestFloor;
    }

    public Lift(int highestFloor, int maxRiders) {
        this(highestFloor);
        capacity = maxRiders;
    }

    public int getTopFloor() {
        return topFloor;
    }

    public int getCurrentFloor() {
        return currentFloor;
    }

    public int getCapacity() {
        return capacity;
    }

    public int getNumRiders() {
        return numRiders;
    }

    public boolean isFull() {
        return numRiders == capacity;
```

Live | Killed | Claimed Equivalent | Equivalent

Mutant/Test restrictions: Moderate

Write a new JUnit test here

Clone previous test | Defend

```java
import org.junit.Test;

import static org.junit.Assert.*;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;

public class TestLift {
    @Test(timeout = 4000)
    public void test() throws Throwable {
        // test here!
    }
}
```

Existing Mutants

All | Alive | Killed | Claimed Equivalent | Equivalent

- All Mutants
- Mutants outside methods
- Lift(int)
- Lift(int, int)

JUnit Tests

- All Tests
- Lift(int)
- Lift(int, int)
- getTopFloor()

You will play both as **attacker** (creating mutants) and as **defender** (creating tests) at the same time. **Have fun!**

Lift Class

```java
public class Lift {

    private int topFloor;
    private int currentFloor = 0; // default
    private int capacity = 10;    // default
    private int numRiders = 0;    // default

    public Lift(int highestFloor) {
        topFloor = highestFloor;
    }

    public Lift(int highestFloor, int maxRiders) {
        this(highestFloor);
        capacity = maxRiders;
    }

    public int getTopFloor() {
        return topFloor;
    }

    public int getCurrentFloor() {
        return currentFloor;
    }

    public int getCapacity() {
        return capacity;
    }

    public int getNumRiders() {
        return numRiders;
    }

    public boolean isFull() {
        return numRiders == capacity;
    }

    public void addRiders(int numEntering) {
        if (numRiders + numEntering <= capacity) {
            numRiders = numRiders + numEntering;
        } else {
            numRiders = capacity;
        }
    }

    public void goUp() {
        if (currentFloor < topFloor)
            currentFloor++;
    }

    public void goDown() {
        if (currentFloor > 0)
            currentFloor--;
    }

    public void call(int floor) {
        if (floor >= 0 && floor <= topFloor) {
            while (floor != currentFloor) {
                if (floor > currentFloor)
                    goUp();
                else
                    goDown();
            }
        }
    }
}
```

Example Test for Lift Class

```
@Test
public void testEmptyLift() {
    Lift l = new Lift(5, 10);
    l.call(3);
    assertFalse(l.isFull());
    assertEquals(3, l.getCurrentFloor());
}
```

Stack Class

```java
public class Stack<T> {
    private int capacity = 10;
    private int pointer  = 0;
    private T[] objects = (T[]) new Object[capacity];

    public void push(T o) {
        if(pointer >= capacity)
            throw new IllegalStateException("Stack exceeded capacity!");
        objects[pointer++] = o;
    }

    public T pop() {
        if(pointer <= 0)
            throw new IllegalStateException("Stack empty");
        return objects[--pointer];
    }

    public boolean isEmpty() {
        return pointer <= 0;
    }
}
```

Example Tests for Stack

```java
@Test
public void testStackException() {
    Stack<Integer> s = new Stack<Integer>();
    assertTrue(s.isEmpty());
    try {
        s.pop();
        fail("pop on an empty list should fail");
    } catch (IllegalStateException e) {
        // It's all good, exception was expected
    }
}
```

```java
@Test
public void testEmptyStack() {
    Stack s = new Stack<String>();
    s.push("Hello");
    assertFalse(s.isEmpty());
}
```