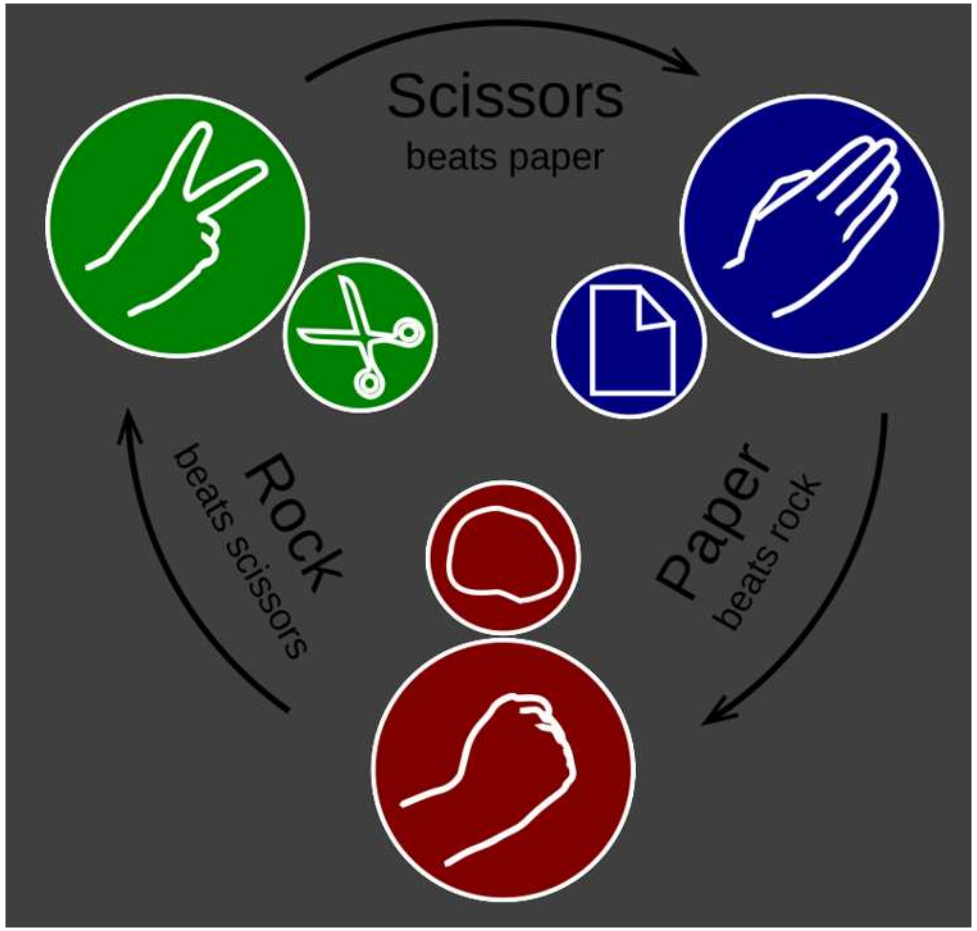
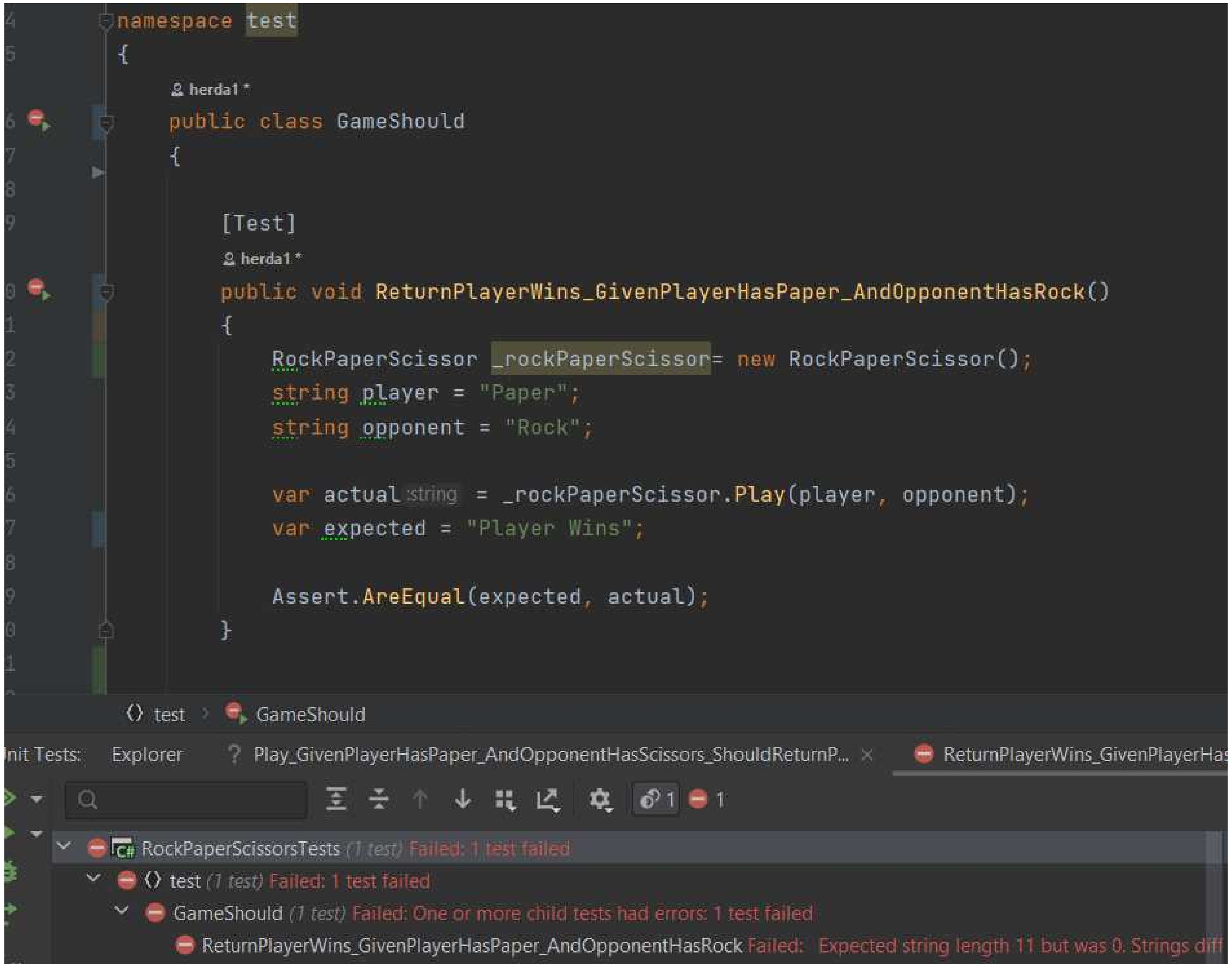
Test Driven Development

Herda Akshija

Case Study: Rock – Paper – Scissors Kata



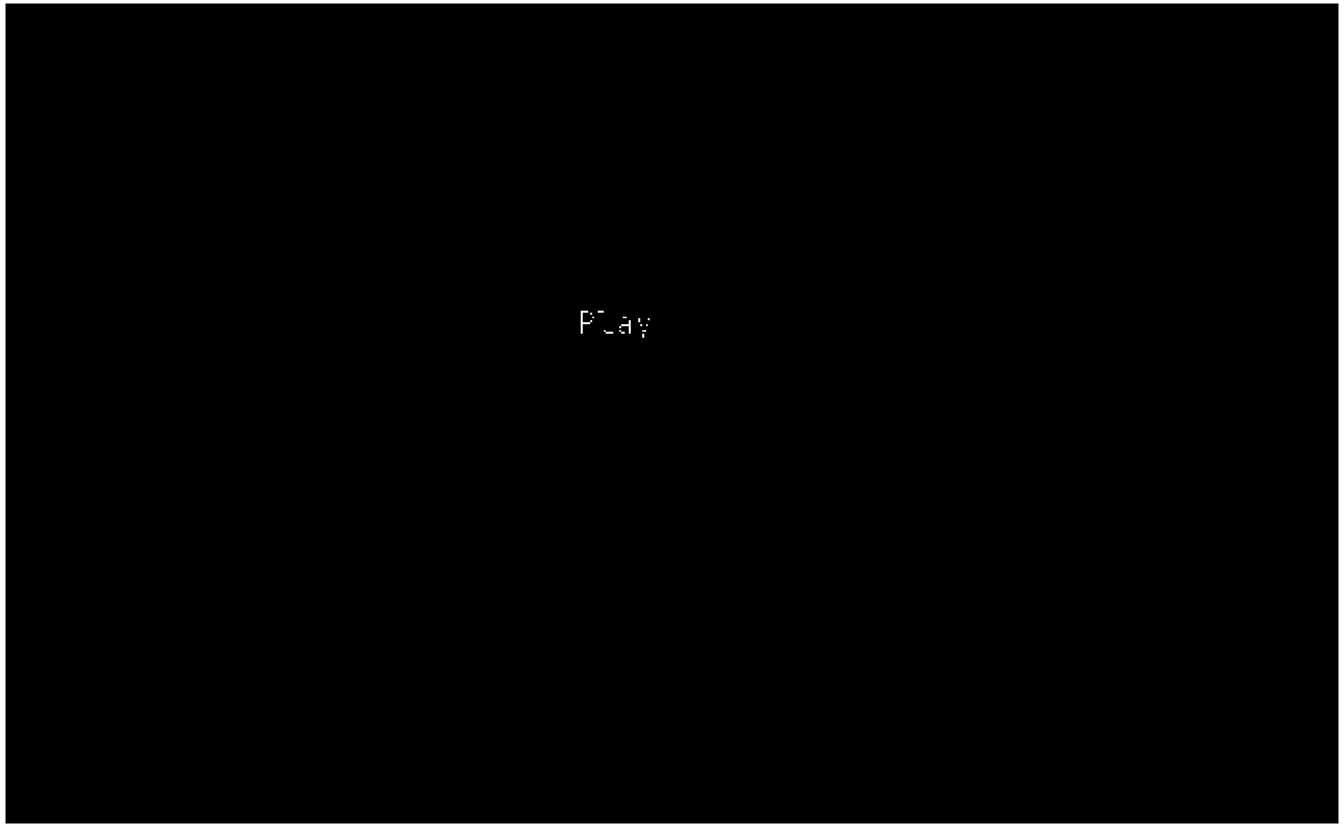
1. Start by writing a failing test



```
namespace test
{
    herda1 *
    public class GameShould
    {

        [Test]
        herda1 *
        public void ReturnPlayerWins_GivenPlayerHasPaper_AndOpponentHasRock()
        {
            RockPaperScissor _rockPaperScissor= new RockPaperScissor();
            string player = "Paper";
            string opponent = "Rock";

            var actual :string = _rockPaperScissor.Play(player, opponent);
            var expected = "Player Wins";

            Assert.AreEqual(expected, actual);
        }
```

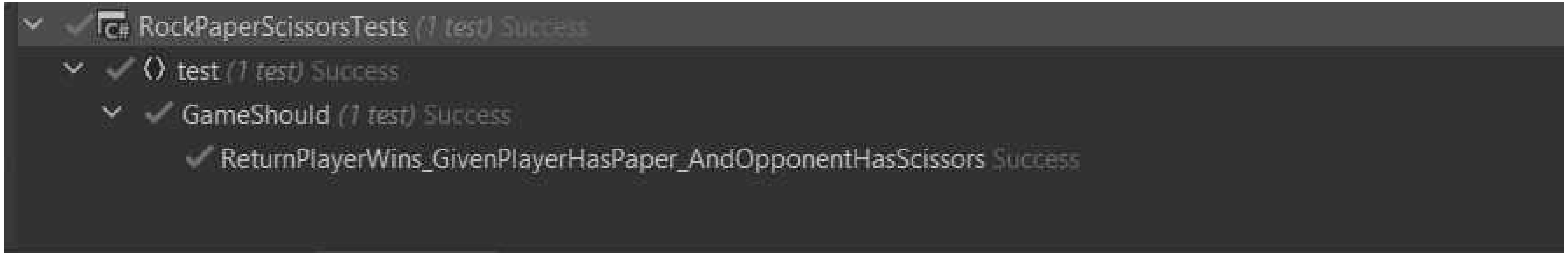
```
test › GameShould
nit Tests:   Explorer   ? Play_GivenPlayerHasPaper_AndOpponentHasScissors_ShouldReturnP... ×   ReturnPlayerWins_GivenPlayerHas
1   1
RockPaperScissorsTests (1 test) Failed: 1 test failed
  test (1 test) Failed: 1 test failed
    GameShould (1 test) Failed: One or more child tests had errors: 1 test failed
      ReturnPlayerWins_GivenPlayerHasPaper_AndOpponentHasRock Failed:  Expected string length 11 but was 0. Strings diff
```

Make sure that is failing for the right reason



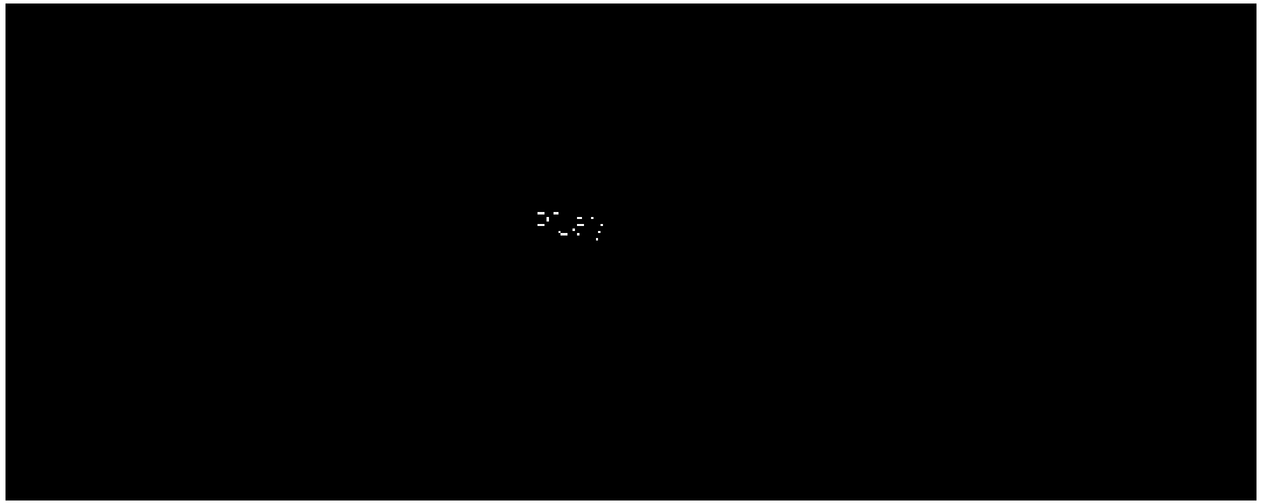
Naming

- Describe a business feature or behavior.
  - Read the test name as a full sentence with the test class name



```
RockPaperScissorsTests (1 test) Success
  test (1 test) Success
    GameShould (1 test) Success
      ReturnPlayerWins_GivenPlayerHasPaper_AndOpponentHasScissors Success
```

2. Then make it pass



From red to green

FAKE IT

OBVIOUS IMPLEMENTATION

TRIANGULATION

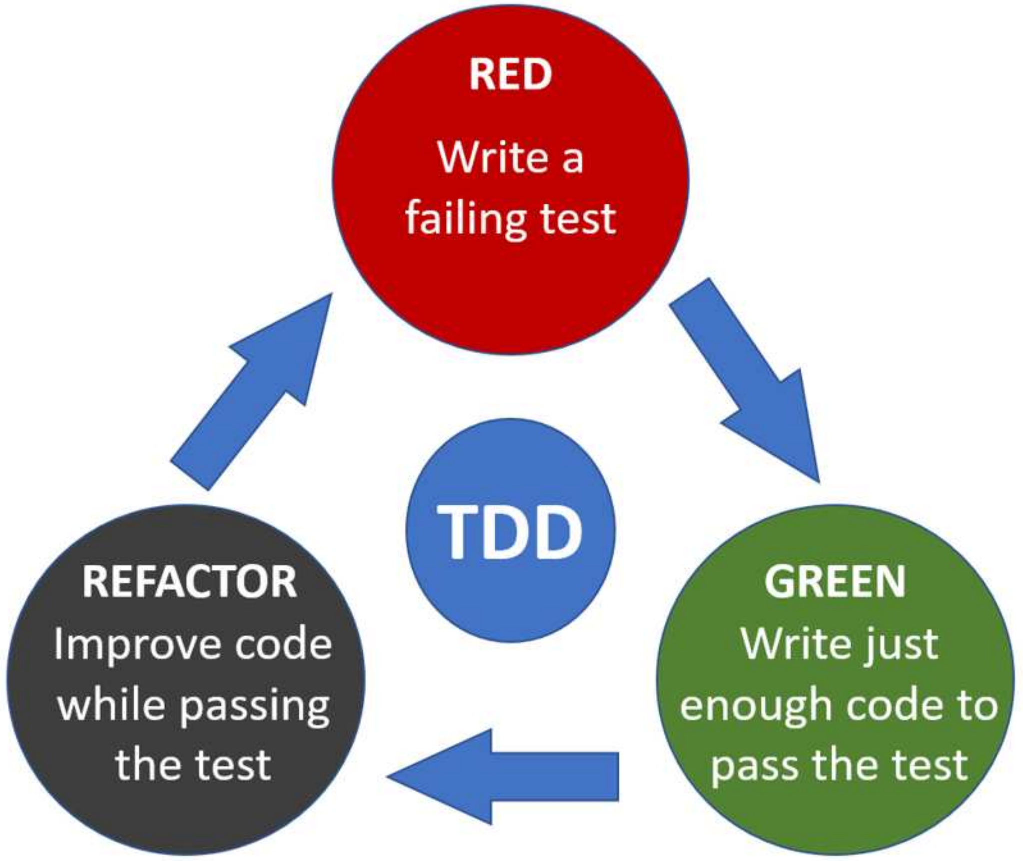
Input	Expected Output	Strategy	Implementation
Player: Paper Opponent: Rock	Player Wins	Fake it	Return “Player Wins”

Player: Paper
Opponent: Scissor
	Player Loses	Obvious implementation	if (opponent == "Scissor")
			    return "Player Loses";

Player: Paper
Opponent: Paper
	Tie	Obvious implementation	if (opponent == "Paper")
			    return “Tie";

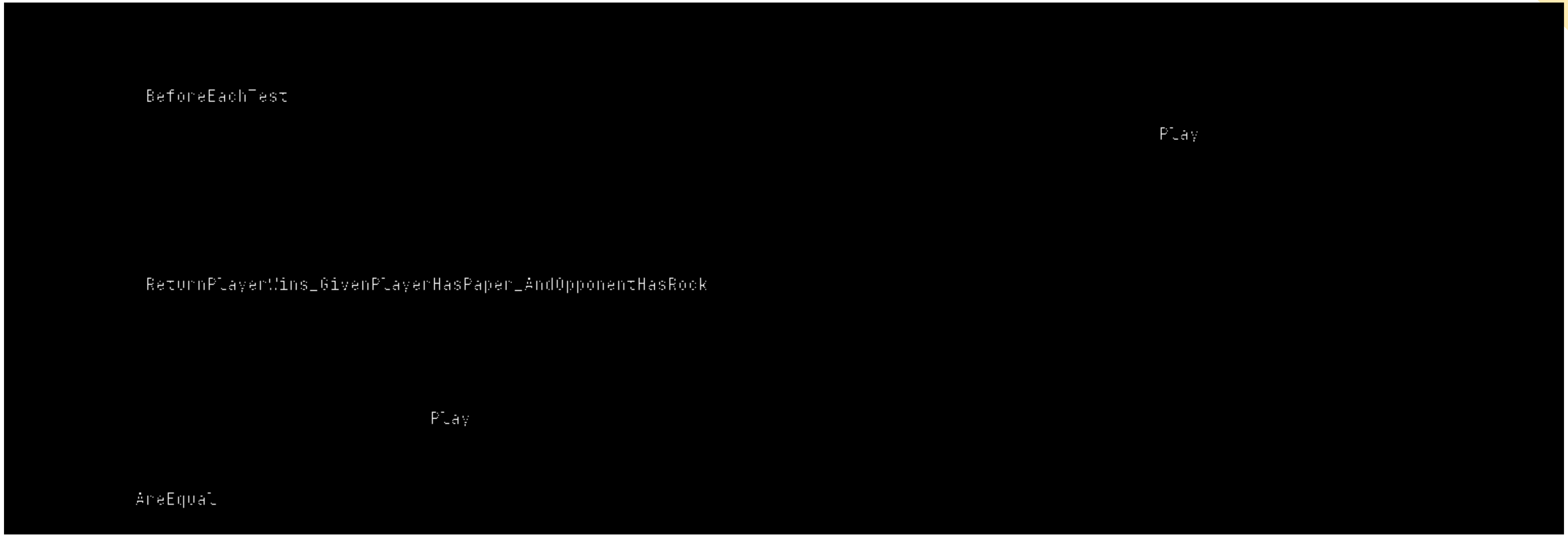
Player: Rock
Player: Rock
	Tie	Obvious implementation
		+ triangulation
			if (opponent == player)
			    return "Tie";

Red, Green, Refactor

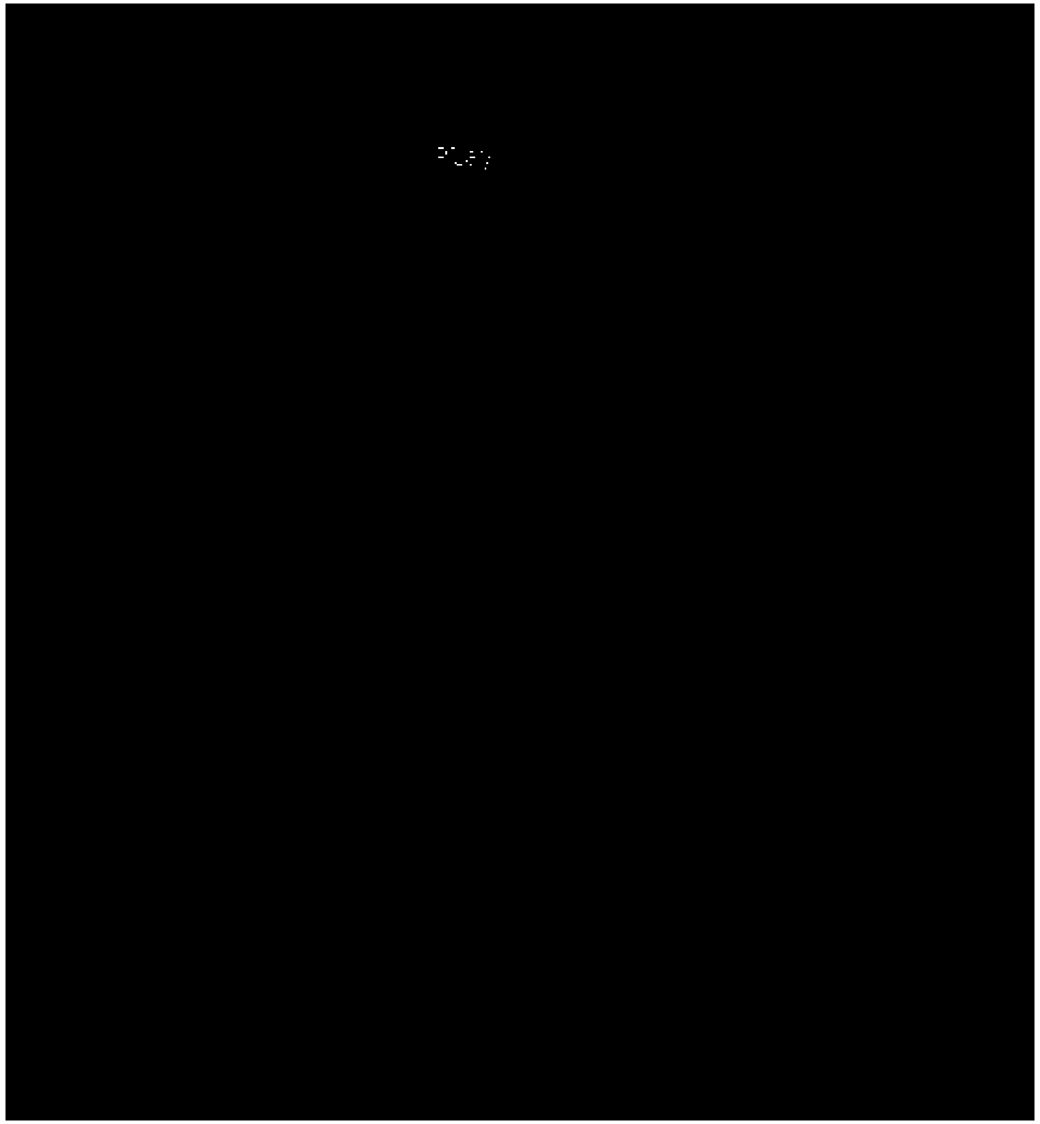

RED
Write a failing test
TDD
GREEN
Write just enough code to pass the test
REFACTOR
Improve code while passing the test

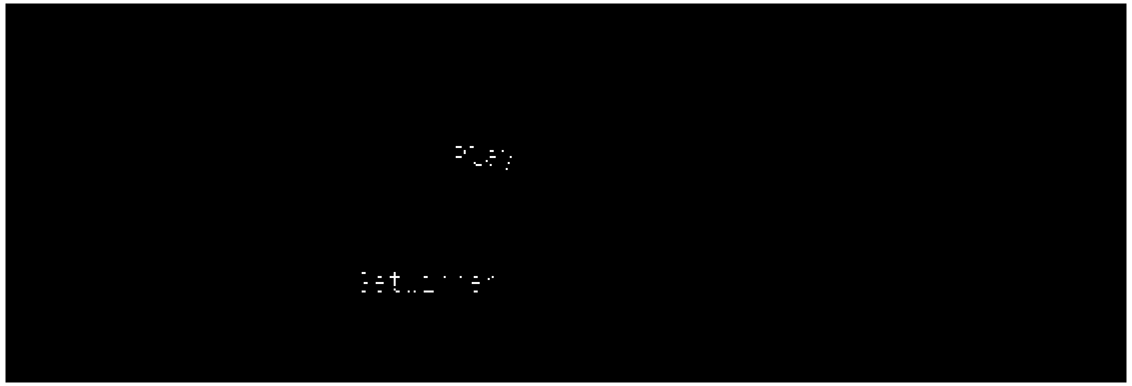
3. When all Green -> Refactor

- The Rule of 3



```
BeforeEachTest
                                                                  Play




ReturnPlayerWins_GivenPlayerHasPaper_AndOpponentHasRock


                  Play

AreEqual
```





Thank you!

Contact:
herda.akshija@bouvet.no