МУ^АЦИОННОЕ АНАЛИЗ

МУ^АЦИОННОЕ АНАЛИЗ

zed

Альфа·Банк

JUG.RU

МОЛЧАНОВ НИКОЛАЙ

github.com/kroniak

Flurl is a modern, fluent, asynchronous, testable, portable, buzzword-laden URL builder and HTTP client library for .NET.

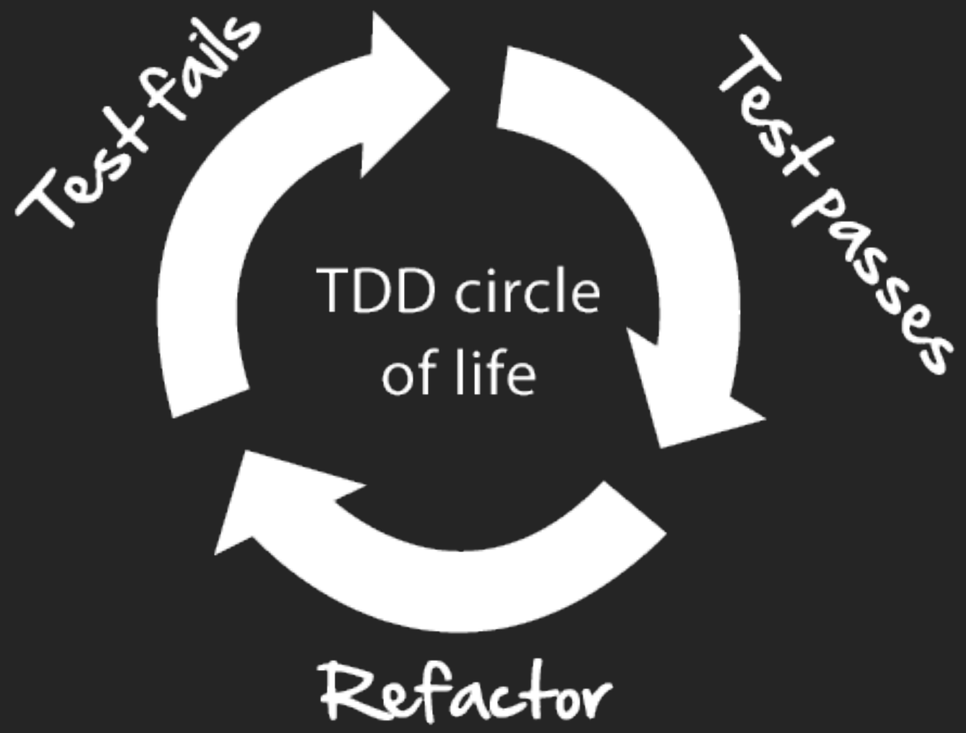
Code It

```
// Flurl will use 1 HttpClient instance per host
var person = await "https://api.com"
    .AppendPathSegment("person")
    .SetQueryParams(new { a = 1, b = 2 })
    .WithOAuthBearerToken("my_oauth_token")
    .PostJsonAsync(new
    {
        first_name = "Claire",
        last_name = "Underwood"
    })
    .ReceiveJson<Person>();
```

Test It

```
// fake & record all http calls in the test subject
using (var httpTest = new HttpTest()) {
    // arrange
    httpTest.RespondWith(200, "OK");
    // act
    await sut.CreatePersonAsync();
    // assert
    httpTest.ShouldHaveCalled("https://api.com/*")
        .WithVerb(HttpMethod.Post)
        .WithContentType("application/json");
}
```

Flurl is a modern, fluent, asynchronous, testable, portable, buzzword-laden URL builder and HTTP client library for .NET.

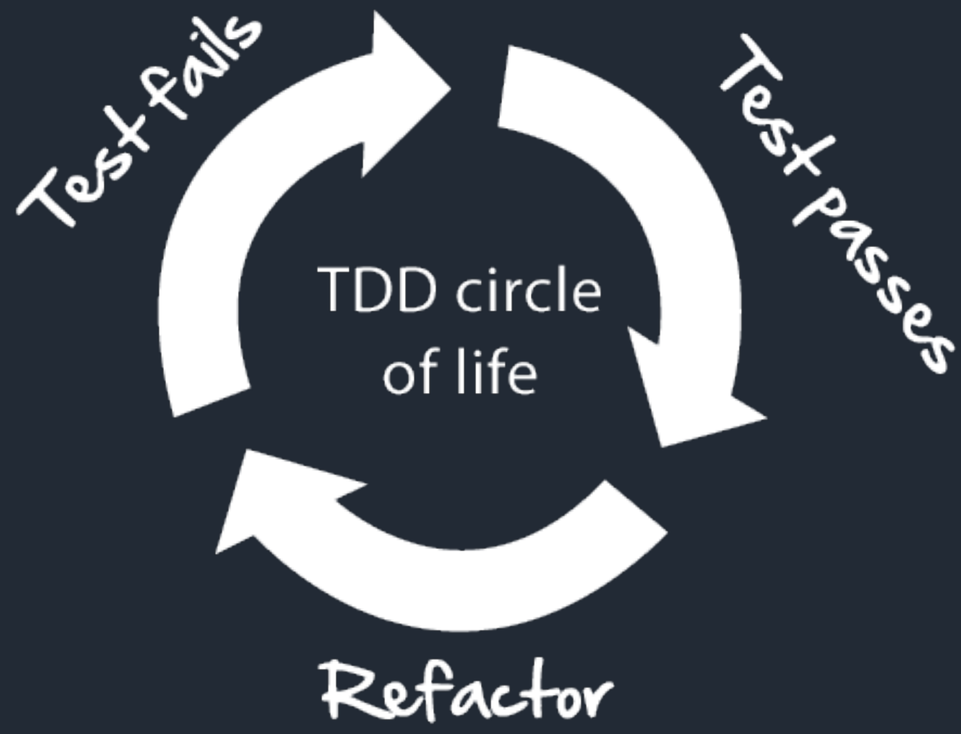


ECWID CLIENT LIBRARY


```
var result =
    await new EcwidClient()
          .Configure(someid, "somekey")
          .Orders
          .PaymentStatuses("PAID")
          .PaymentStatuses("REFUNDED")
          .FulfillmentStatuses("DELIVERED")
          .CreatedFrom(DateTime.Today)
          .Updated("2016-05-01","2016-05-02")
          .Customer("test@test.ts")
          .Keywords("cool phone")
          .TotalFrom(100.00)
          .PaymentMethod("PayPal")
          .Limit(10)
          .Offset(10)
          .GetAsync();
```

ECWID CLIENT LIBRARY

```
var result =
   await new EcwidClient()
         .Configure(someid, "somekey")
         .Orders
         .PaymentStatuses("PAID")
         .PaymentStatuses("REFUNDED")
         .FulfillmentStatuses("DELIVERED")
         .CreatedFrom(DateTime.Today)
         .Updated("2016-05-01","2016-05-02")
         .Customer("test@test.ts")
         .Keywords("cool phone")
         .TotalFrom(100.00)
         .PaymentMethod("PayPal")
         .Limit(10)
         .Offset(10)
         .GetAsync();
```



ПЛАН

ПЛАН

- тестирование и парадигмы тестирования

ПЛАН

- тестирование и парадигмы тестирования
- про принцип мутационного тестирования

ПЛАН

- тестирование и парадигмы тестирования
- про принцип мутационного тестирования
- виды мутантов

ПЛАН

- тестирование и парадигмы тестирования
- про принцип мутационного тестирования
- виды мутантов
- как применить в .NET

ПЛАН

- тестирование и парадигмы тестирования
- про принцип мутационного тестирования
- виды мутантов
- как применить в .NET
- когда использовать мутационное тестирование

КТО ЛЮБИТ ПИСАТЬ ТЕСТЫ?

ВИДЫ ТЕСТИРОВАНИЯ

- функциональное тестирование

ВИДЫ ТЕСТИРОВАНИЯ

- функциональное тестирование
- unit-тестирование

ВИДЫ ТЕСТИРОВАНИЯ

- функциональное тестирование
- unit-тестирование
- автотестирование интеграционное

ВИДЫ ТЕСТИРОВАНИЯ

- функциональное тестирование
- unit-тестирование
- автотестирование интеграционное
- автотестирование UI и e2e

ТЕСТИРОВАНИЕ ЭТО ПРОВЕРКА СООТВЕТСТВИЯ ТРЕБОВАНИЯМ

ТЕСТИРОВАНИЕ ЭТО ПРОВЕРКА СООТВЕТСТВИЯ ТРЕБОВАНИЯМ

ORACLE

> 40 000 tests

> 5 hour testing

ЕСТЬ ТУТ ПЕРФЕКЦИОНИСТЫ?

ПОКРЫТИЕ > 70% ?

Tesla Model Autopilot

https://www.tesla.com/blog/tragic-loss

Boeing 737 MAX 8

http://avherald.com/h?article=4c534c4a/0052

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ?

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

- code-review кода тестов

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

- code-review кода тестов
- code-coverage

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

- code-review кода тестов
- code-coverage
- test after each build

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

- code-review кода тестов
- code-coverage
- test after each build
- внешний аудит

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

- code-review кода тестов
- code-coverage
- test after each build
- внешний аудит
- рандомный набор данных

ПОКРЫТИЕ 100
И
ОШИБКИ ЕСТЬ

TEST COVERAGE IS DEAD

LONG LIVE
THE MUTATION SCORE

МУТАЦИОННОЕ ТЕСТИРОВАНИЕ

” Мутационное тестирование - это преднамеренное внесение в код специальных изменений - мутаций с целью проверки программного обеспечения ”

” **Мутационное тестирование** - это преднамеренное внесение в код **специальных изменений - мутаций** с целью проверки программного обеспечения ”

» **Мутационное тестирование** - это преднамеренное внесение в код специальных изменений - мутаций с целью **проверки программного обеспечения** «

” Мутационный анализ - это мероприятие для вычисления опасных изменений кода и рефакторинг кода для минимизации рисков ”

BDD => TDD => MDD

” MDD (major depressive disorder) – основное депрессивное расстройство личности ”

— Mental and behavioral disorders (F00–F99 & 290–319)

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

КАК ПРОТЕСТИРОВАТЬ ТЕСТЫ

=>

JUnit 5

nunit

СОПРОТИВЛЕНИЕ

2 способа

СОПРОТИВЛЕНИЕ

2 способа

- типы и компиляция

СОПРОТИВЛЕНИЕ

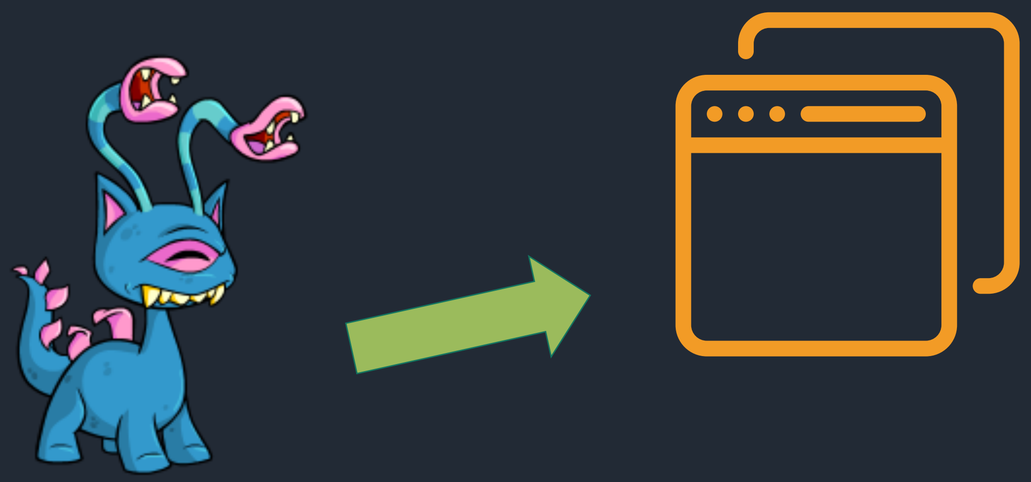
2 способа

- типы и компиляция
- наборы тестов

ЧТО ЛУЧШЕ?

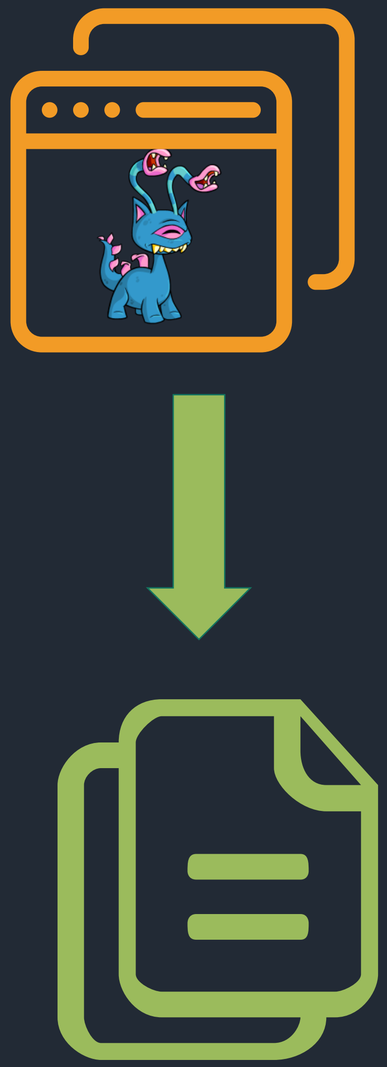
ВЫЖИВАНИЕ

ВЫЖИВАНИЕ



ВЫЖИВАНИЕ

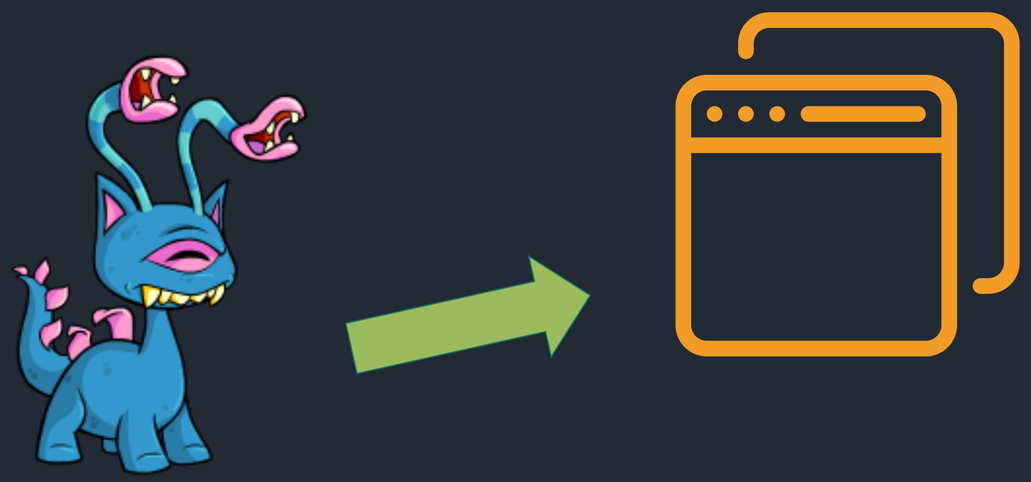
ВЫЖИВАНИЕ



выживший мутант

ВЫЖИВАНИЕ

ВЫЖИВАНИЕ



ВЫЖИВАНИЕ

ВЫЖИВАНИЕ

мертвый мутант

$$\exists\ t \in T:\ t(S_M) \neq t(S)$$

```
public static bool Method(bool a, bool b)
{
    if (a && b)
        return true;
    else
        return false;
}
```

```
public static bool Method(bool a, bool b)
{
    if (a && b)
        return true;
    else
        return false;
}
```

```
[Theory]
[InlineData(false, false)]
public void Method_Data_ReturnFalse(bool a, bool b)
{
    // Act
    var result = SampleClass.Method(a, b);

    // Assert
    Assert.False(result);
}
```

```
public static bool Method(bool a, bool b)
{
    if (a && b)
        return true;
    else
        return false;
}
```

```
[Theory]
[InlineData(false, false)]
public void Method_Data_ReturnFalse(bool a, bool b)
{
    // Act
    var result = SampleClass.Method(a, b);

    // Assert
    Assert.False(result);
}
```

```
public static bool Method(bool a, bool b)
{
    if (a && b)    (a || b)
        return true;
    else
        return false;
}
```

```
[Theory]
[InlineData(false, false)]
public void Method_Data_ReturnFalse(bool a, bool b)
{
    // Act
    var result = SampleClass.Method(a, b);

    // Assert
    Assert.False(result);
}
```

```
public static bool Method(bool a, bool b)
{
    if (a && b)    (a || b)
        return true;
    else
        return false;
}
```

```
[Theory]
[InlineData(false, false)]
[InlineData(true, false)]
[InlineData(false, true)]
public void Method_Data_ReturnFalse(bool a, bool b)
{
    // Act
    var result = SampleClass.Method(a, b);

    // Assert
    Assert.False(result);
}
```

3 условия:

1. Покрытие мутанта тестом

3 условия:

1. Покрытие мутанта тестом
2. Полные входные данные

3 условия:

1. Покрытие мутанта тестом
2. Полные входные данные
3. Выходные параметры проверены

ВИДЫ МУТАЦИЙ

ВИДЫ МУТАЦИЙ

ООП мутации

ВИДЫ МУТАЦИЙ

ООП мутации

1. Мутации модификаторов доступа

ВИДЫ МУТАЦИЙ

ООП мутации

1. Мутации модификаторов доступа
2. Мутации наследования

ВИДЫ МУТАЦИЙ

ООП мутации

1. Мутации модификаторов доступа
2. Мутации наследования
3. Полиморфные мутации

ВИДЫ МУТАЦИЙ

ООП мутации

1. Мутации модификаторов доступа
2. Мутации наследования
3. Полиморфные мутации
4. Мутации перегрузки методов и их аргументов

ВИДЫ МУТАЦИЙ

ООП мутации

1. Мутации модификаторов доступа
2. Мутации наследования
3. Полиморфные мутации
4. Мутации перегрузки методов и их аргументов
5. Общие мутации

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции - => +

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения

```
>     =>    =>
>     =>    <
```

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения
3. Мутации логических операторов `&&` `=` `||`

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения
3. Мутации логических операторов
4. Мутации bool типа

```
true  =>  false
```

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения
3. Мутации логических операторов
4. Мутации bool типа
5. Унарные операции

```
++i    =>    i++
++i    =>    --i
```

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения
3. Мутации логических операторов
4. Мутации bool типа
5. Унарные операции
6. Строковые мутации

“” => “asd”

“asd” => “”

ВИДЫ МУТАЦИЙ

Общие мутации

1. Арифметические операции
2. Мутации сравнения
3. Мутации логических операторов
4. Мутации bool типа
5. Унарные операции
6. Строковые мутации
7. Мутации массивов и списков [0,1,2] => [0,0,2]

ВИДЫ МУТАЦИЙ

LINQ мутации

Last	=>	Fisrt
Any	=>	All
First	=>	Single
FirstOrDefault	=>	SingleOrDefault

СТАТИСТИКА

Flurl

9000 строк
85%

Ecwid

5000 строк
90%

СТАТИСТИКА

Flurl

9000 строк
85%

300 тестов
43 сек

Ecwid

5000 строк
90%

200 тестов
16 сек

СТАТИСТИКА

Flurl	Ecwid
9000 строк	5000 строк
85%	90%
300 тестов	200 тестов
43 сек	16 сек
363 мутанта	300 мутантов
30 минут	15 минут

СТАТИСТИКА ВЫЖИВШИХ МУТАЦИЙ

мертвый код

тесты

эквивалентные

ПРИМЕРЫ МУТАЦИЙ

1. Строковые мутации

```
bool ExceptionMethod (bool con) {
   if (!con)
      throw new ArgumentException ("USER_DB is not path pattern user_adminXXXX",
         nameof (con));

   return true;
}
```

```
bool ExceptionMethod (bool con) {
   if (!con)
      throw new ArgumentException ("USER_DB is not path pattern user_adminXXXX",
         nameof (con));

   return true;
}
```

```
bool ExceptionMethod (bool con) {
   if (!con)
      throw new ArgumentException ("USER_DB is not path pattern user_adminXXXX",
         nameof (con));

   return true;
}
```

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
   // Act
   var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

   // Assert
   Assert.Equal ("Something happened", ex.Message);
   Assert.Equal ("USER_DB is not path pattern user_adminXXXX (Parameter 'con')",
      ex.InnerException?.Message);
}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
    // Act
    var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

    // Assert
    Assert.Equal ("Something happened", ex.Message);


}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
    // Act
    var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

    // Assert
    Assert.Equal ("Something happened", ex.Message);


}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
    // Act
    var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

    // Assert
    Assert.Equal ("Something happened", ex.Message);


}
```

Что забыли?

```
bool ExceptionMethod (bool con) {
   if (!con)
      throw new ArgumentException ("USER_DB is not path pattern user_adminXXXX",
         nameof (con));

   return true;
}
```

```
bool ExceptionMethod (bool con) {
   if (!con)
      throw new ArgumentException ("USER_DB is not path pattern user_adminXXXX",
         nameof (con));
   return true;
}
```

“USER_DB…” => “”

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
string LibraryMethod (bool con) {
    try {

        if (ExceptionMethod (con))
            return "Successfully";

    } catch (ArgumentException e) {

        throw new Exception ("Something happened", e);
    }

    return string.Empty;
}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
    // Act
    var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

    // Assert
    Assert.Equal ("Something happened", ex.Message);


}
```

```
[Theory]
[InlineData (false)]
public void LibraryMethod_NegativeData_ThrowException (bool val) {
   // Act
   var ex = Assert.Throws<Exception> (() => SampleClass.LibraryMethod (val));

   // Assert
   Assert.Equal ("Something happened", ex.Message);
   Assert.Equal ("USER_DB is not path pattern user_adminXXXX (Parameter 'con')",
      ex.InnerException?.Message);
}
```

2. Мутации сравнения

ПРИМЕР 1

```
public string Encode (string s) {
    if (string.IsNullOrWhiteSpace (s)) return s;

    if (s.Length >  MAX_URL_LENGTH) {
        // ... other actions with s

        return s;
    }

    return s;
}
```

ПРИМЕР 1

```
public string Encode (string s) {
    if (string.IsNullOrWhiteSpace (s)) return s;

    if (s.Length >  MAX_URL_LENGTH) {
        // ... other actions with s

        return s;
    }

    return s;
}
```

ПРИМЕР 1

```
public string Encode (string s) {
    if (string.IsNullOrWhiteSpace (s)) return s;

    if (s.Length >  MAX_URL_LENGTH) {
        // ... other actions with s

        return s;
    }

    return s;
}
```

ПРИМЕР 1

```
public string Encode (string s) {
    if (string.IsNullOrWhiteSpace (s)) return s;

    if (s.Length >  MAX_URL_LENGTH) {
        // ... other actions with s

        return s;
    }

    return s;
}
```

ПРИМЕР 1

```
public string Encode (string s) {
    if (string.IsNullOrWhiteSpace (s)) return s;

    if (s.Length >= MAX_URL_LENGTH) {
        // ... other actions with s

        return s;
    }

    return s;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
[Theory]
[InlineData (100)]
[InlineData (101)]
public void Limit_Greater100_ReturnCorrectData (int value) {
   Assert.Equal (100, Limit (value));
}
```

ПРИМЕР 2

```
[Theory]
[InlineData (98)]
[InlineData (99)]
public void Limit_Smaller100_ReturnCorrectData (int value) {
   Assert.Equal (value, Limit (value));
}
```

ПРИМЕР 2

```
[Theory]
[InlineData (-1)]
public void Limit_Smaller0_ThrowException (int value) {
    Assert.Throws<ArgumentException> (() => Limit (value));
}
```

ПРИМЕР 2

```
[Theory]
[InlineData (100)]
[InlineData (101)]
public void Limit_Greater100_ReturnCorrectData (int value) {
   Assert.Equal (100, Limit (value));
}

[Theory]
[InlineData (98)]
[InlineData (99)]
public void Limit_Smaller100_ReturnCorrectData (int value) {
   Assert.Equal (value, Limit (value));
}

[Theory]
[InlineData (-1)]
public void Limit_Smaller0_ThrowException (int value) {
   Assert.Throws<ArgumentException> (() => Limit (value));
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 100) limit = 100;

    return limit;
}

[InlineData (100)]
[InlineData (101)]
[InlineData (98)]
[InlineData (99)]
[InlineData (-1)]
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if 1.(limit <= 0) 2.(limit > 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if 1.(limit >= 100) 2.(limit < 100) limit = 100;

    return limit;
}

[InlineData (100)]
[InlineData (101)]
[InlineData (98)]
[InlineData (99)]
[InlineData (-1)]
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if 1.(limit <= 0) 2.(limit > 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if 1.(limit >= 100) 2.(limit < 100) limit = 100;

    return limit;
}

[InlineData (100)]
[InlineData (101)]
[InlineData (98)]
[InlineData (99)]
[InlineData (-1)]
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if 1.(limit <= 0) 2.(limit > 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if 1.(limit >= 100) 2.(limit < 100) limit = 100;

    return limit;
}

[InlineData (100)]
[InlineData (101)]
[InlineData (98)]
[InlineData (99)]
[InlineData (-1)]
```

ПРИМЕР 2

```
[Theory]
[InlineData (100)]
[InlineData (101)]
public void Limit_Greater100_ReturnCorrectData (int value) {
   Assert.Equal (100, Limit (value));
}

[Theory]
[InlineData (98)]
[InlineData (99)]
public void Limit_Smaller100_ReturnCorrectData (int value) {
   Assert.Equal (value, Limit (value));
}

[Theory]
[InlineData (-1)]
public void Limit_Smaller0_ThrowException (int value) {
   Assert.Throws<ArgumentException> (() => Limit (value));
}
```

Добавляем

```
[InlineData (0)]
```

ПРИМЕР 2

```
[Theory]
[InlineData (100)]
[InlineData (101)]
public void Limit_Greater100_ReturnCorrectData (int value) {
   Assert.Equal (100, Limit (value));
}

[Theory]
[InlineData (0)]
[InlineData (98)]
[InlineData (99)]
public void Limit_Smaller100_ReturnCorrectData (int value) {
   Assert.Equal (value, Limit (value));
}

[Theory]
[InlineData (-1)]
public void Limit_Smaller0_ThrowException (int value) {
   Assert.Throws<ArgumentException> (() => Limit (value));
}
```

Добавляем

```
[InlineData (0)]
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if 1.(limit <= 0) 2.(limit > 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if 1.(limit >= 100) 2.(limit < 100) limit = 100;

    return limit;
}
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if 1.(limit <= 0) 2.(limit > 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if 1.(limit >= 100) 2.(limit < 100) limit = 100;

    return limit;
}
```

Эквивалентная мутация – та, которую нельзя убить тестом

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 99) limit = 100;

    return limit;
}

[InlineData (100)]
[InlineData (101)]
[InlineData (0)]
[InlineData (98)]
[InlineData (99)]
[InlineData (-1)]
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 99) limit = 100;

    return limit;
}
```

```
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit > 0'
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit <= 0'

[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit < 99'
[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit >= 99'
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 99) limit = 100;

    return limit;
}
```

Что забыли?

```
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit > 0'
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit <= 0'

[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit < 99'
[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit >= 99'
```

ПРИМЕР 2

```
public int Limit (int limit) {
    if (limit < 0)
        throw new ArgumentException ("Limit must be greater than 0", nameof (limit));

    if (limit > 99) limit = 100;

    return limit;
}
```

```
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit > 0'
[Killed] Binary expression mutation on line 58: 'limit < 0' ==> 'limit <= 0'
[Survived] String mutation on line 59: '"Limit must be greater than 0"' ==> '""'
[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit < 99'
[Killed] Binary expression mutation on line 61: 'limit > 99' ==> 'limit >= 99'
```

ПРИМЕР 3

```
void Some<T> (object arg, IEnumerable<T> list) {
    // ...
    var result = new List<T> ();
    // ...

    if (result.Count > 0) {
        // ...
    }
    // ...
}
```

ПРИМЕР 3

```
void Some<T> (object arg, IEnumerable<T> list) {
    // ...
    var result = new List<T> ();
    // ...

    if (result.Count > 0) {       (result.Count >= 0)
        // ...
    }
    // ...
}
```

ПРИМЕР 3

```
void Some<T> (object arg, IEnumerable<T> list) {
    // ...
    var result = new List<T> ();
    // ...

    if (result.Any ()) {
        // ...
    }
    // ...
}
```

ПРИМЕР 3

```
void Some<T> (object arg, IEnumerable<T> list) {
    // ...
    var result = new List<T> ();
    // ...

    if (result.FirstOrDefault () != null) {
        // ...
    }
    // ...
}
```

ПРИМЕР 4

```
async Task<string> GetValueAsync (string url) {
    var values = await GetFromUrlAsync (url);

    // ... other work

    return values.FirstOrDefault ();
}


[Fact]
async Task GetValue_ReturnCorrectDataAsync () {
    //... mock data
    var result = await client.GetValueAsync ("some url");

    Assert.Equal ("some data", result);
    //... some other assertion
}
```

ПРИМЕР 4

```
async Task<string> GetValueAsync (string url) {
    var values = await GetFromUrlAsync (url);

    // ... other work

    return values.LastOrDefault ();
}


[Fact]
async Task GetValue_ReturnCorrectDataAsync () {
    //... mock data
    var result = await client.GetValueAsync ("some url");

    Assert.Equal ("some data", result);
    //... some other assertion
}
```

3. Эквивалентная мутация

```
public int ForMethod () {
    var index = 0;

    while (true) {

        index++;
        if (index == 10) {
            break;
        }
    }

    return index;
}
```

```
public int ForMethod () {
   var index = 0;

   while (true) {

      index++;
      if (index == 10) {
         break;
      }
   }

   return index;
}
```

(index >= 10)

(index > 10)

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;


   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   if (container.All (c => c.SomeField != data.SomeField)) return;
                           c.SomeField == data.SomeField

   container.Remove (data); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_NotExist_NotRemoved() {
   // Arrange
   var expected = container.Count;
   var data = new Value(4);

   // Act
   Remove(container, data);

   // Assert
   Assert.Equal(expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.FirstOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.FirstOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_Exist_Removed () {
   // Arrange
   var expected = container.Count;
   var data = new Value (container[1].SomeField);

   // Act
   Remove (container, data);

   // Assert
   Assert.Equal (expected - 1, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.FirstOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}

[Fact]
public void Remove_Exist_Removed () {
   // Arrange
   var expected = container.Count;
   var data = new Value (container[1].SomeField);

   // Act
   Remove (container, data);

   // Assert
   Assert.Equal (expected - 1, container.Count);
   Assert.False (container.Any (c => c.SomeField == data.SomeField));
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.FirstOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.SingleOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.SingleOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}

[Fact]
public void RemoveThird_ExistDouble_ThrowsException () {
   // Arrange
   var expected = container.Count;
   var data = new Value (container[1].SomeField);

   // Act
   Assert.Throws<InvalidOperationException> (() => RemoveThird (container, data));

   // Assert
   Assert.Equal (expected, container.Count);
}
```

ПРИМЕР

```
public void Remove (IList<Value> container, Value data) {
   if (data == null) return;

   var deleteT = container.SingleOrDefault (c => c.SomeField == data.SomeField);
   if (deleteT == null) return;

   container.Remove (deleteT); // очень много связанной логики и т.п.
}

[Fact]
public void RemoveThird_ExistDouble_ThrowsException () {
   // Arrange
   var expected = container.Count;
   var data = new Value (container[1].SomeField);

   // Act
   Assert.Throws<InvalidOperationException> (() => RemoveThird (container, data));

   // Assert
   Assert.Equal (expected, container.Count);
}
```

ЭКВИВАЛЕНТНАЯ МУТАЦИЯ

II

V

РЕФАКТОРИНГ


STRYKER
KILL THE MUTANTS
Powered by
InfoSupport

https://stryker-mutator.io/

Особенности

1. Покрыт тестами, в том числе мутационными

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core
3. Только общие мутации и LINQ мутации

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core
3. Только общие мутации и LINQ мутации
4. Поддержка конфигурации

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core
3. Только общие мутации и LINQ мутации
4. Поддержка конфигурации
5. Отключение части мутаторов

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core
3. Только общие мутации и LINQ мутации
4. Поддержка конфигурации
5. Отключение части мутаторов
6. Ограничение на проекты или на файлы и методы

STRYKER

Особенности

1. Покрыт тестами, в том числе мутационными
2. Работает сейчас нормально только для .NET Core
3. Только общие мутации и LINQ мутации
4. Поддержка конфигурации
5. Отключение части мутаторов
6. Ограничение на проекты или на файлы и методы
7. Флаг красного билда

STRYKER

1. 

STRYKER

1. 

2. 

STRYKER

1. 
2. 
3.  *SharpSyntaxTree*

STRYKER

1. 

2. 

3.  *SharpSyntaxTree*

4. 

STRYKER

1. 
2. 
3. 
4. 
5.  *Mutate recursion*

STRYKER

1. 
2. 
3. 
4. 
5. 
6.  *CSharpCompilation*

STRYKER

1. 
2. 
3. 
4. 
5. 
6. 
7. 

Инъекция в код

STRYKER

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 

TPL n-threads

МУТАЦИЯ

```
interface IMutator {
   IEnumerable<Mutation> Mutate (SyntaxNode node);
}

abstract class MutatorBase<T> where T : SyntaxNode {
   abstract IEnumerable<Mutation> ApplyMutations (T node);

   public IEnumerable<Mutation> Mutate (SyntaxNode node) {
      if (node is T tNode) return ApplyMutations (tNode);
      else return Enumerable.Empty<Mutation> ();
   }
}
```

МУТАЦИЯ

```
interface IMutator {
   IEnumerable<Mutation> Mutate (SyntaxNode node);
}

abstract class MutatorBase<T> where T : SyntaxNode {
   abstract IEnumerable<Mutation> ApplyMutations (T node);

   public IEnumerable<Mutation> Mutate (SyntaxNode node) {
      if (node is T tNode) return ApplyMutations (tNode);
      else return Enumerable.Empty<Mutation> ();
   }
}
```

МУТАЦИЯ

```
interface IMutator {
   IEnumerable<Mutation> Mutate (SyntaxNode node);
}

abstract class MutatorBase<T> where T : SyntaxNode {
   abstract IEnumerable<Mutation> ApplyMutations (T node);

   public IEnumerable<Mutation> Mutate (SyntaxNode node) {
      if (node is T tNode) return ApplyMutations (tNode);
      else return Enumerable.Empty<Mutation> ();
   }
}
```

МУТАЦИЯ

```
private readonly Dictionary<SyntaxKind, SyntaxKind> kindsToMutate { get; }

public BooleanMutator () {
   kindsToMutate = new Dictionary<SyntaxKind, SyntaxKind> {
            {
               SyntaxKind.TrueLiteralExpression,
               SyntaxKind.FalseLiteralExpression },
            {
               SyntaxKind.FalseLiteralExpression,
               SyntaxKind.TrueLiteralExpression }
      };
}

public override IEnumerable<Mutation> ApplyMutations (LiteralExpressionSyntax node) {
   if (kindsToMutate.ContainsKey (node.Kind ())) {
      yield return new Mutation () {
         OriginalNode = node,
         ReplacementNode = SyntaxFactory.LiteralExpression (kindsToMutate[node.Kind ()]),
         DisplayName = "Boolean mutation",
         Type = Mutator.Boolean
      };
   }
}
```

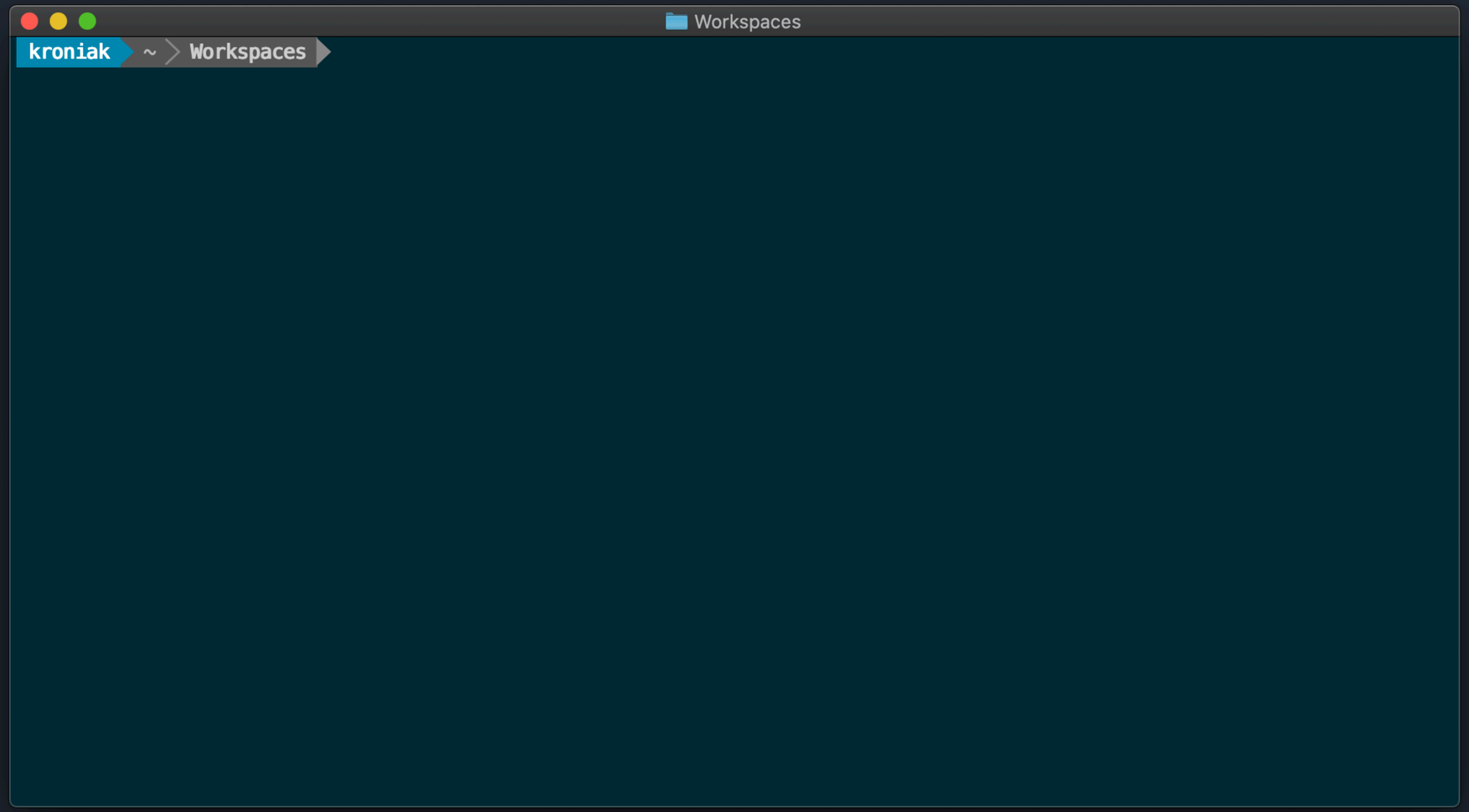
МУТАЦИЯ

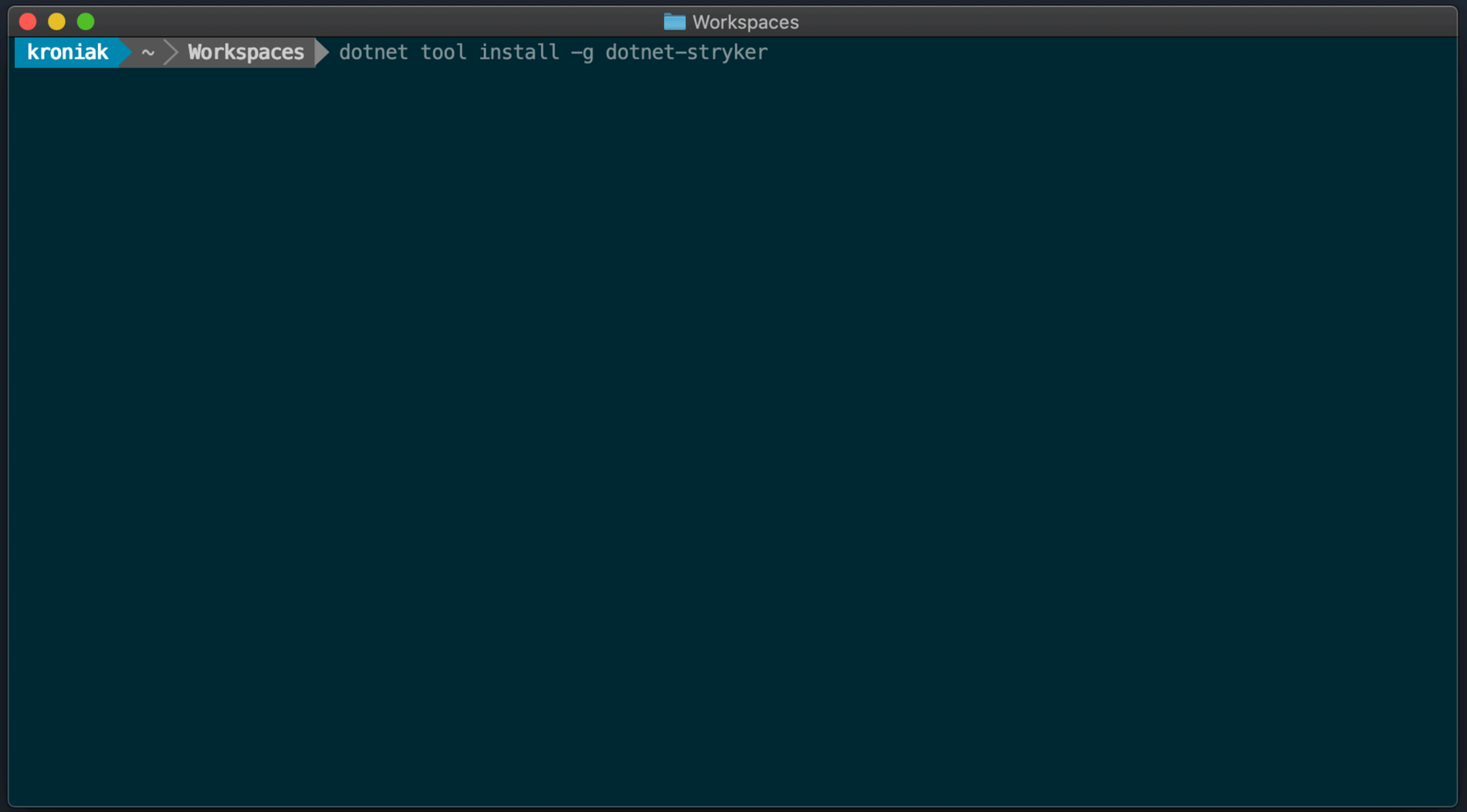
```
private readonly Dictionary<SyntaxKind, SyntaxKind> kindsToMutate { get; }

public BooleanMutator () {
   kindsToMutate = new Dictionary<SyntaxKind, SyntaxKind> {
            {
                SyntaxKind.TrueLiteralExpression,
                SyntaxKind.FalseLiteralExpression },
            {
                SyntaxKind.FalseLiteralExpression,
                SyntaxKind.TrueLiteralExpression }
        };
}

public override IEnumerable<Mutation> ApplyMutations (LiteralExpressionSyntax node) {
   if (kindsToMutate.ContainsKey (node.Kind ())) {
      yield return new Mutation () {
         OriginalNode = node,
         ReplacementNode = SyntaxFactory.LiteralExpression (kindsToMutate[node.Kind ()]),
         DisplayName = "Boolean mutation",
         Type = Mutator.Boolean
      };
   }
}
```

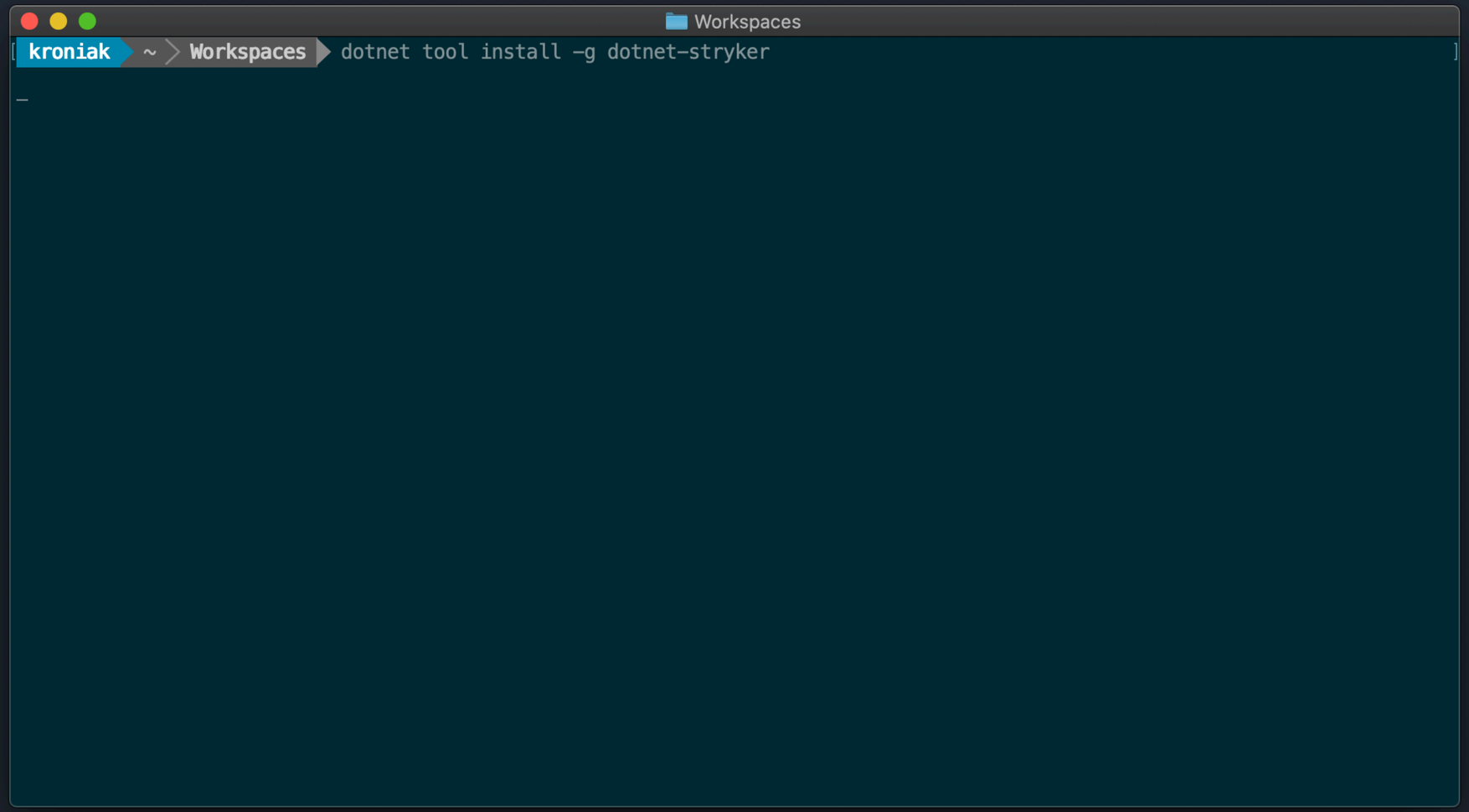
МУТАЦИЯ

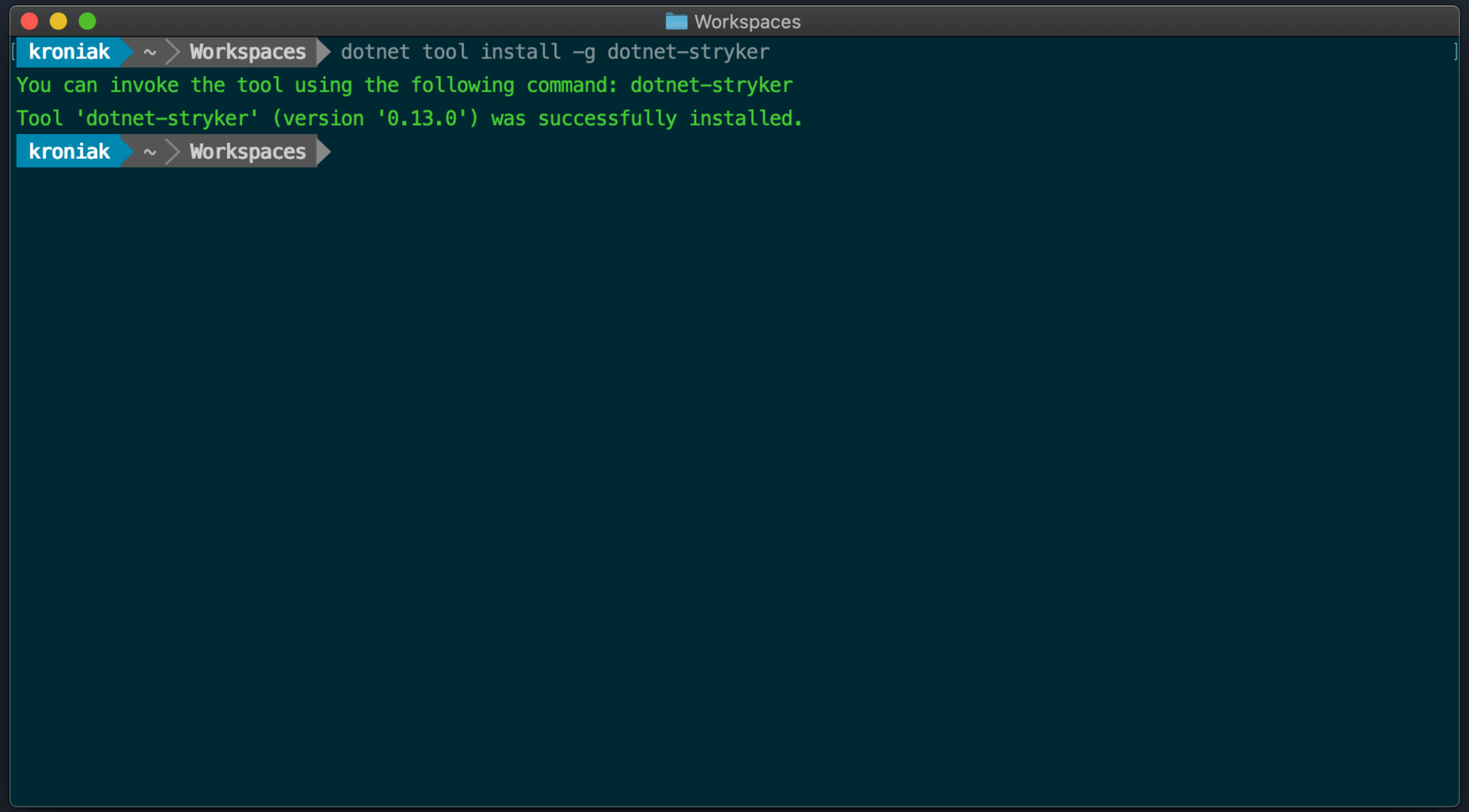
```
private readonly Dictionary<SyntaxKind, SyntaxKind> kindsToMutate { get; }

public BooleanMutator () {
   kindsToMutate = new Dictionary<SyntaxKind, SyntaxKind> {
            {
               SyntaxKind.TrueLiteralExpression,
               SyntaxKind.FalseLiteralExpression },
            {
               SyntaxKind.FalseLiteralExpression,
               SyntaxKind.TrueLiteralExpression }
         };
}

public override IEnumerable<Mutation> ApplyMutations (LiteralExpressionSyntax node) {
   if (kindsToMutate.ContainsKey (node.Kind ())) {
      yield return new Mutation () {
         OriginalNode = node,
         ReplacementNode = SyntaxFactory.LiteralExpression (kindsToMutate[node.Kind ()]),
         DisplayName = "Boolean mutation",
         Type = Mutator.Boolean
      };
   }
}
```

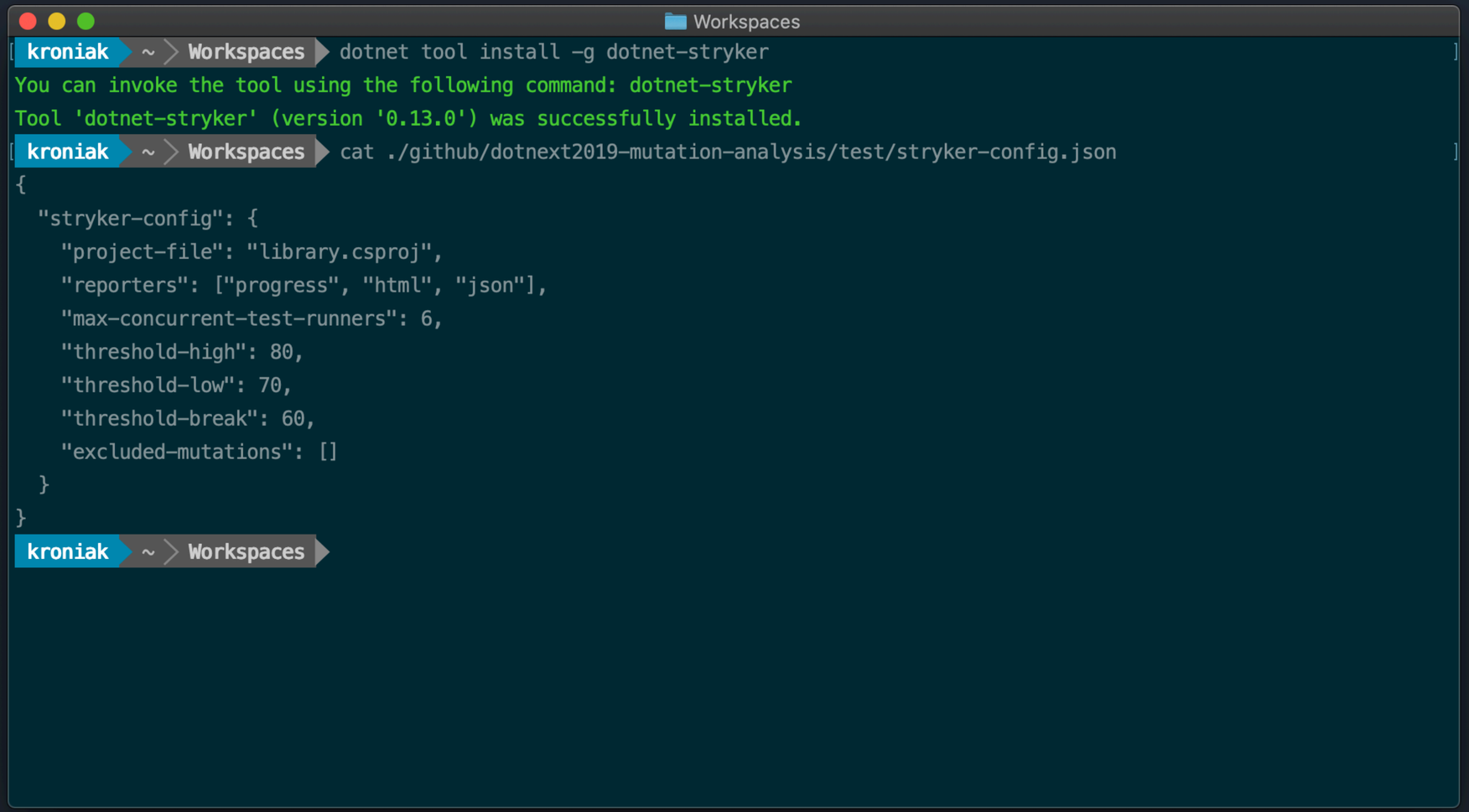
КАК РАБОТАЕТ

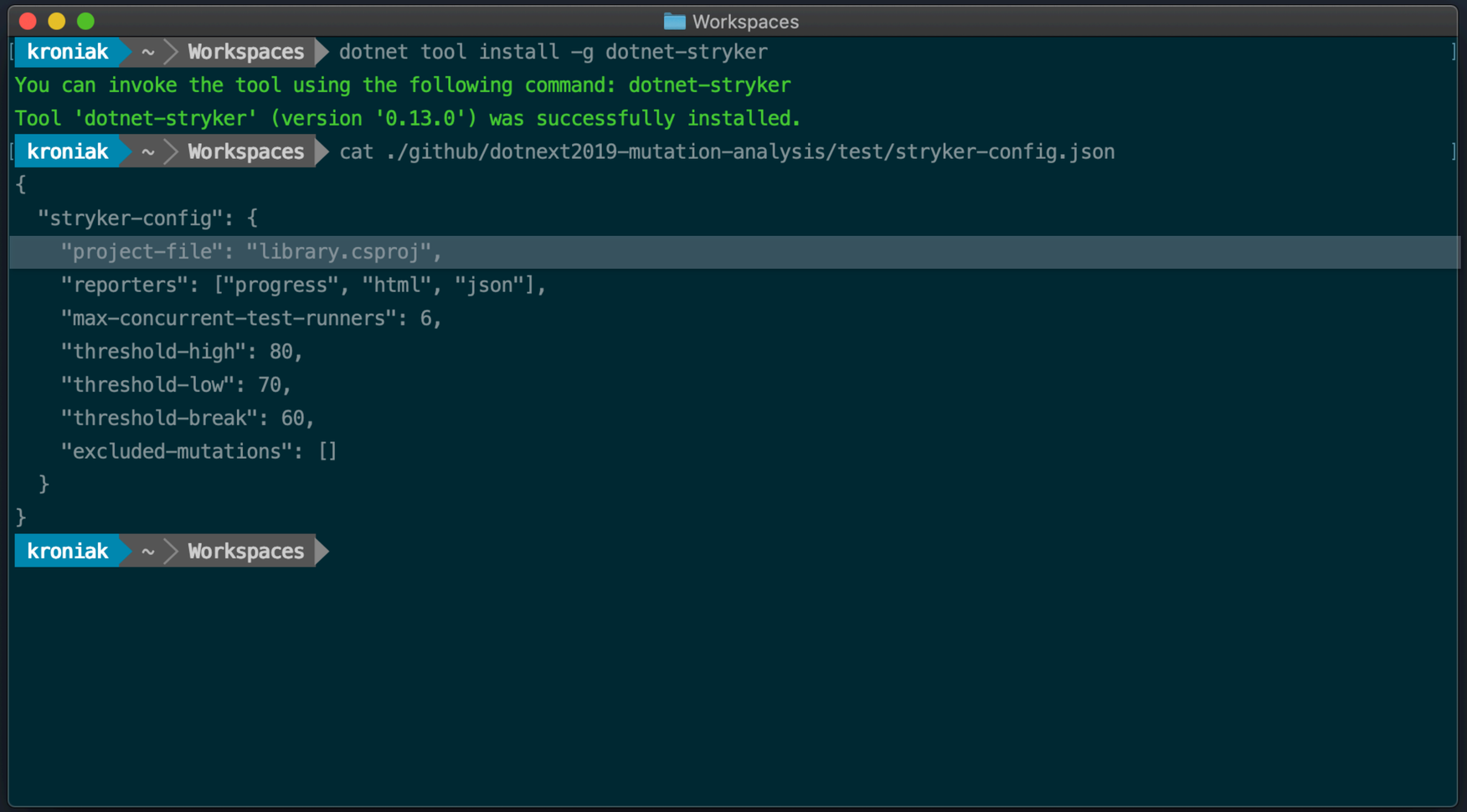

Workspaces
kroniak ~ Workspaces

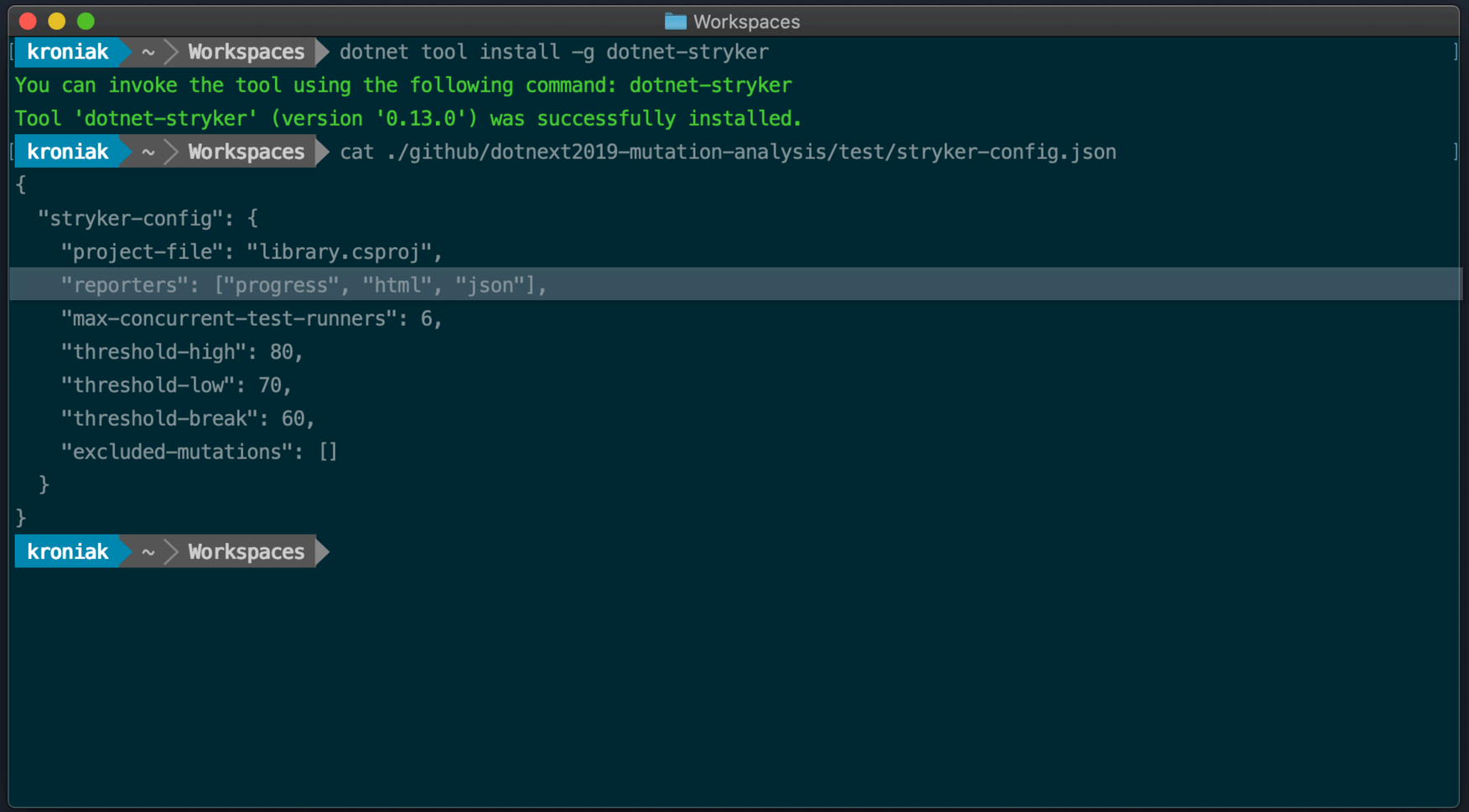

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker

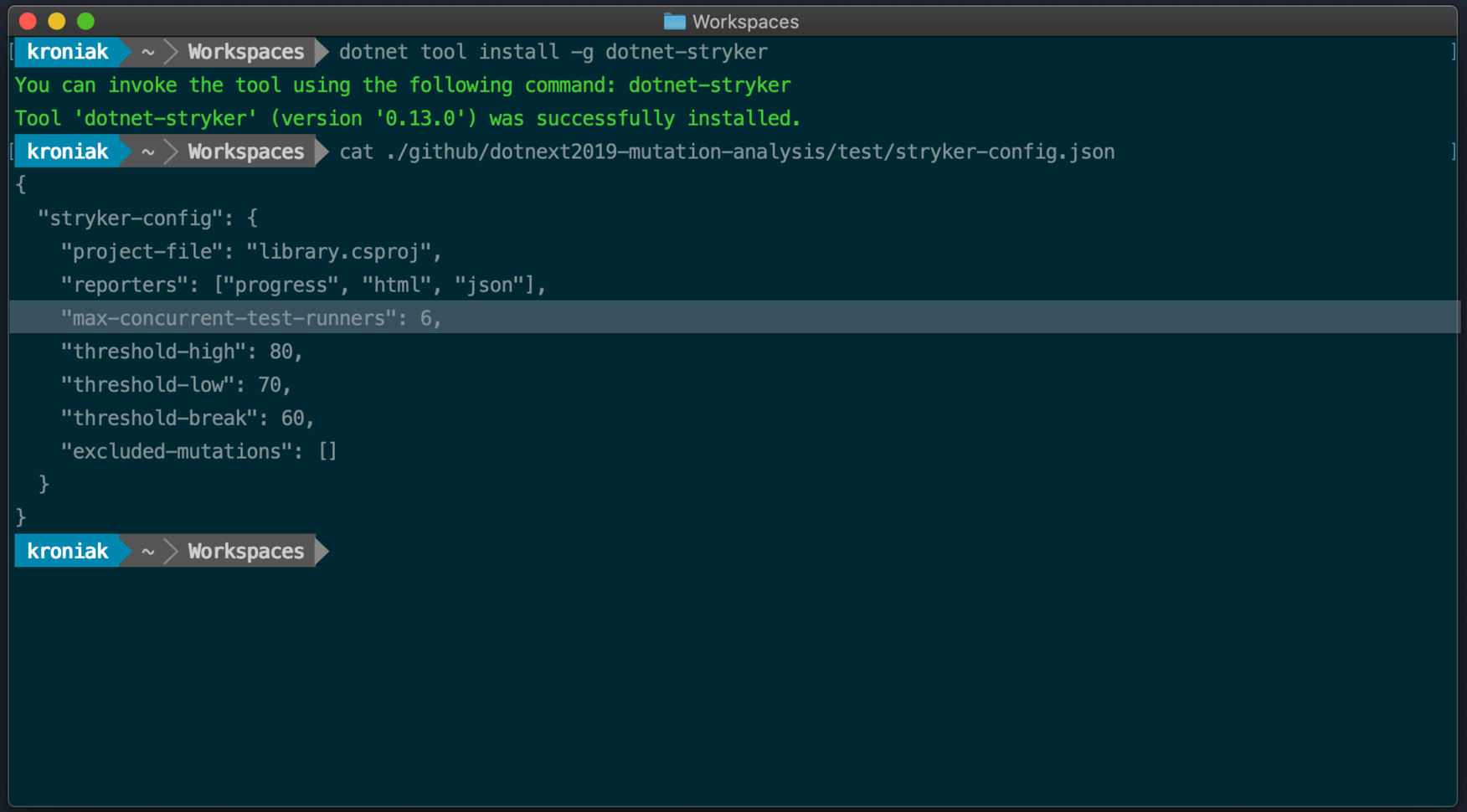

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker

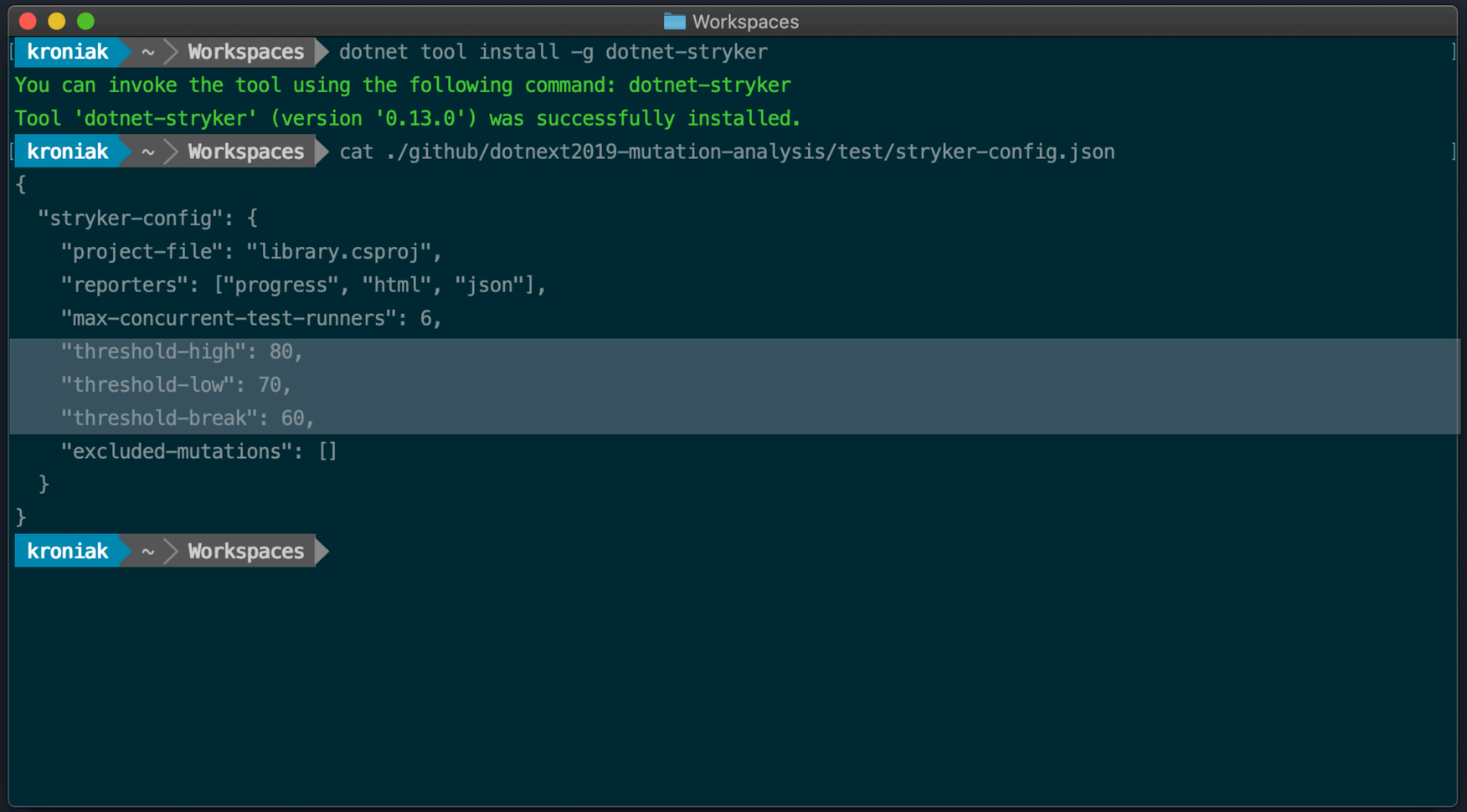

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces

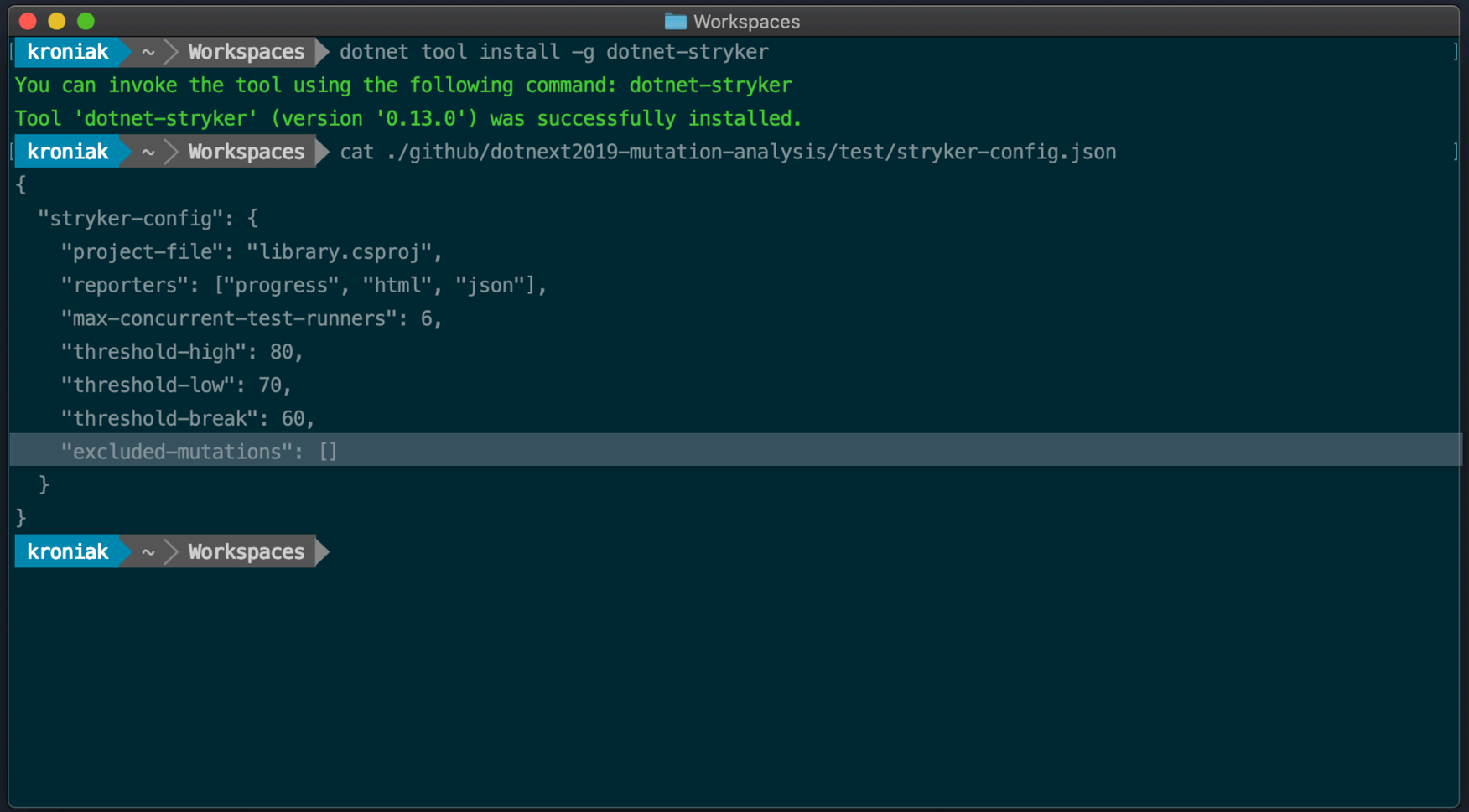

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

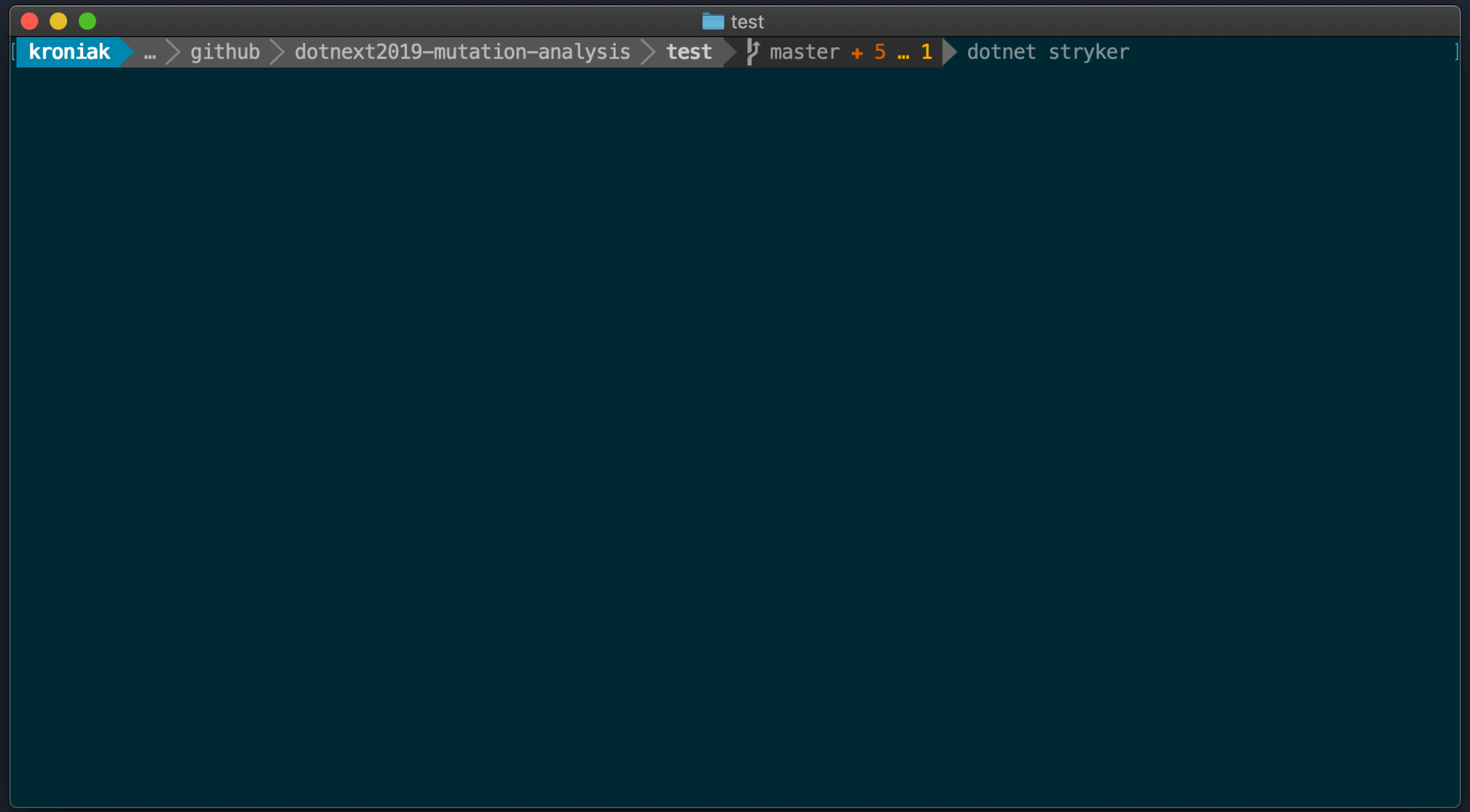

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

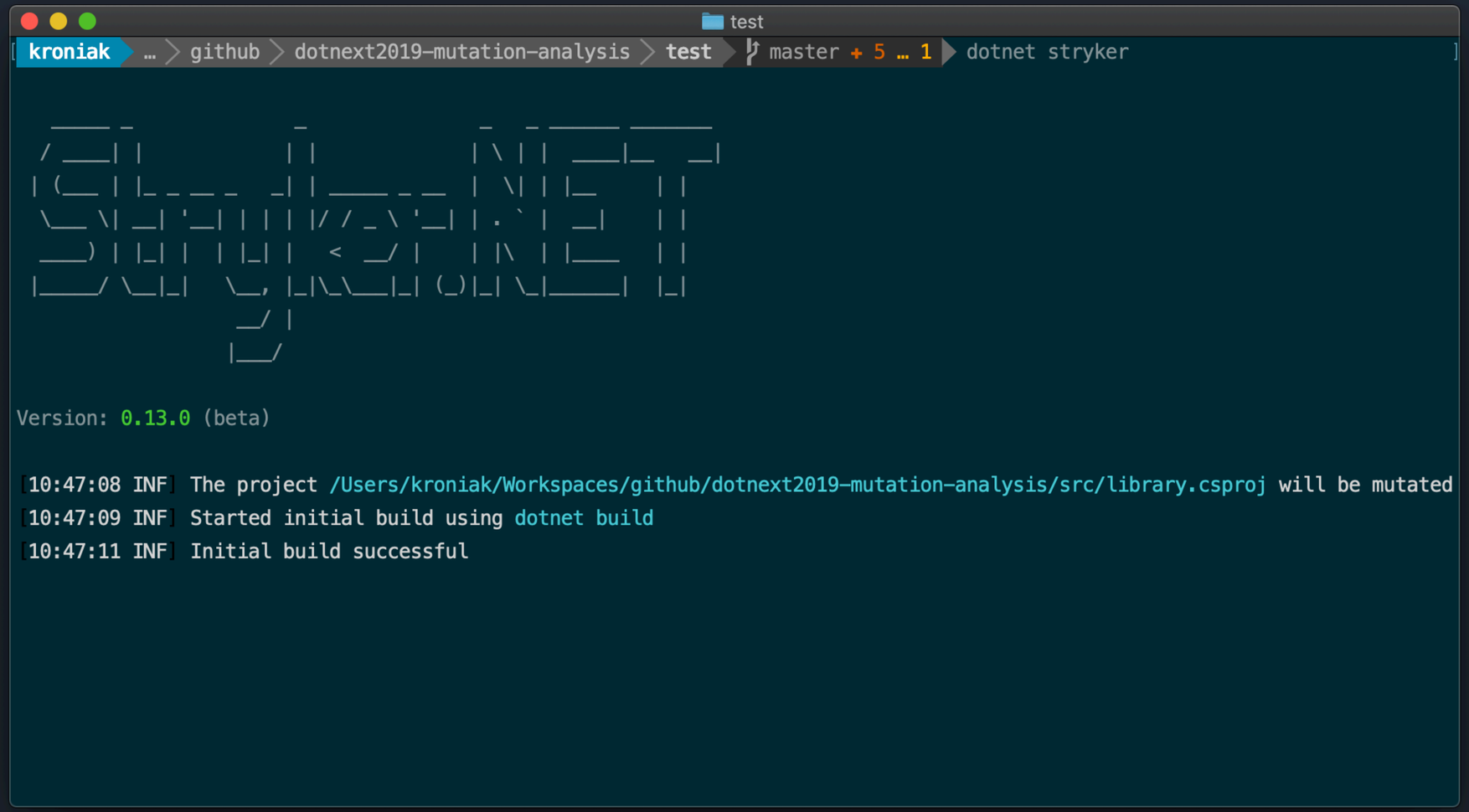

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

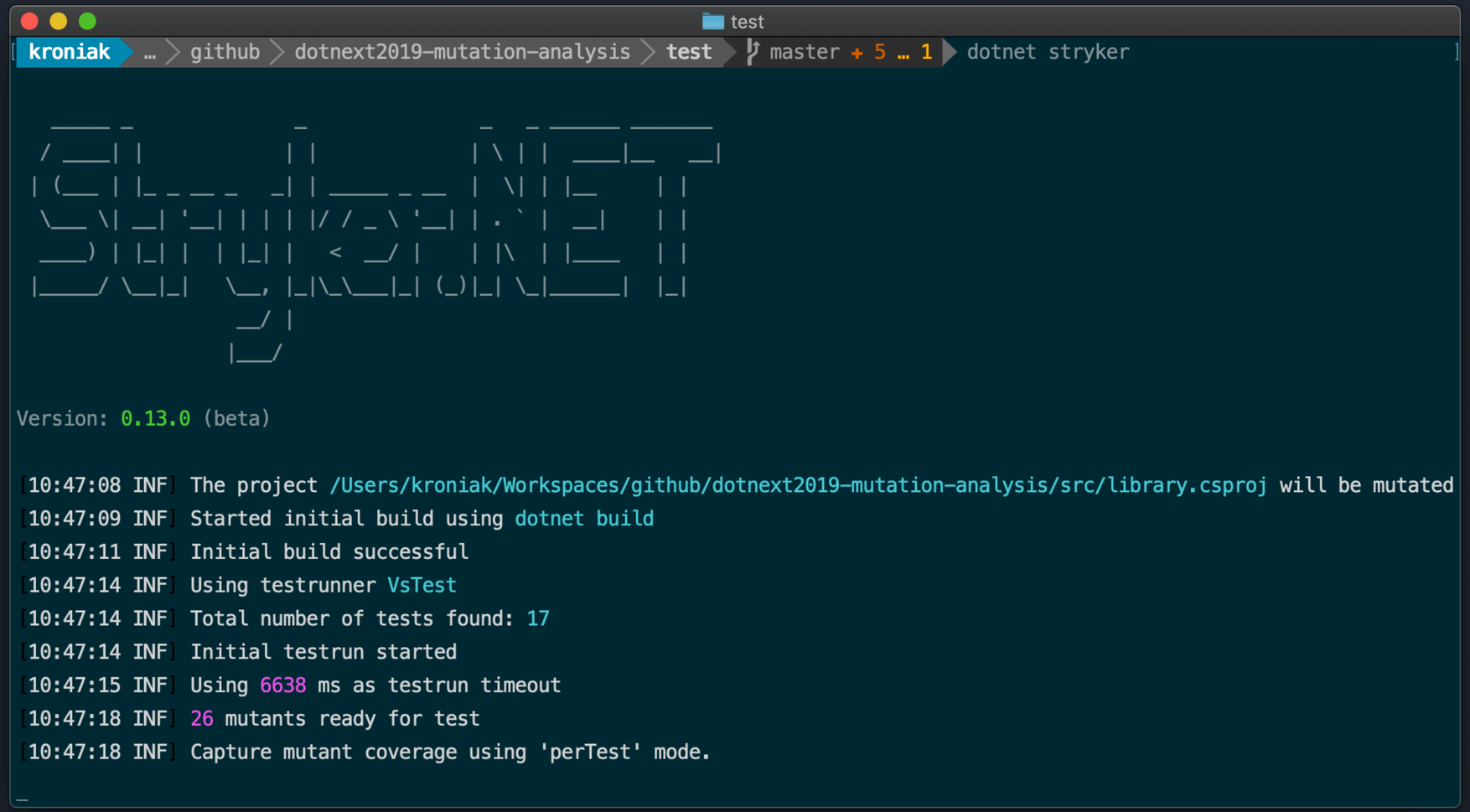

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

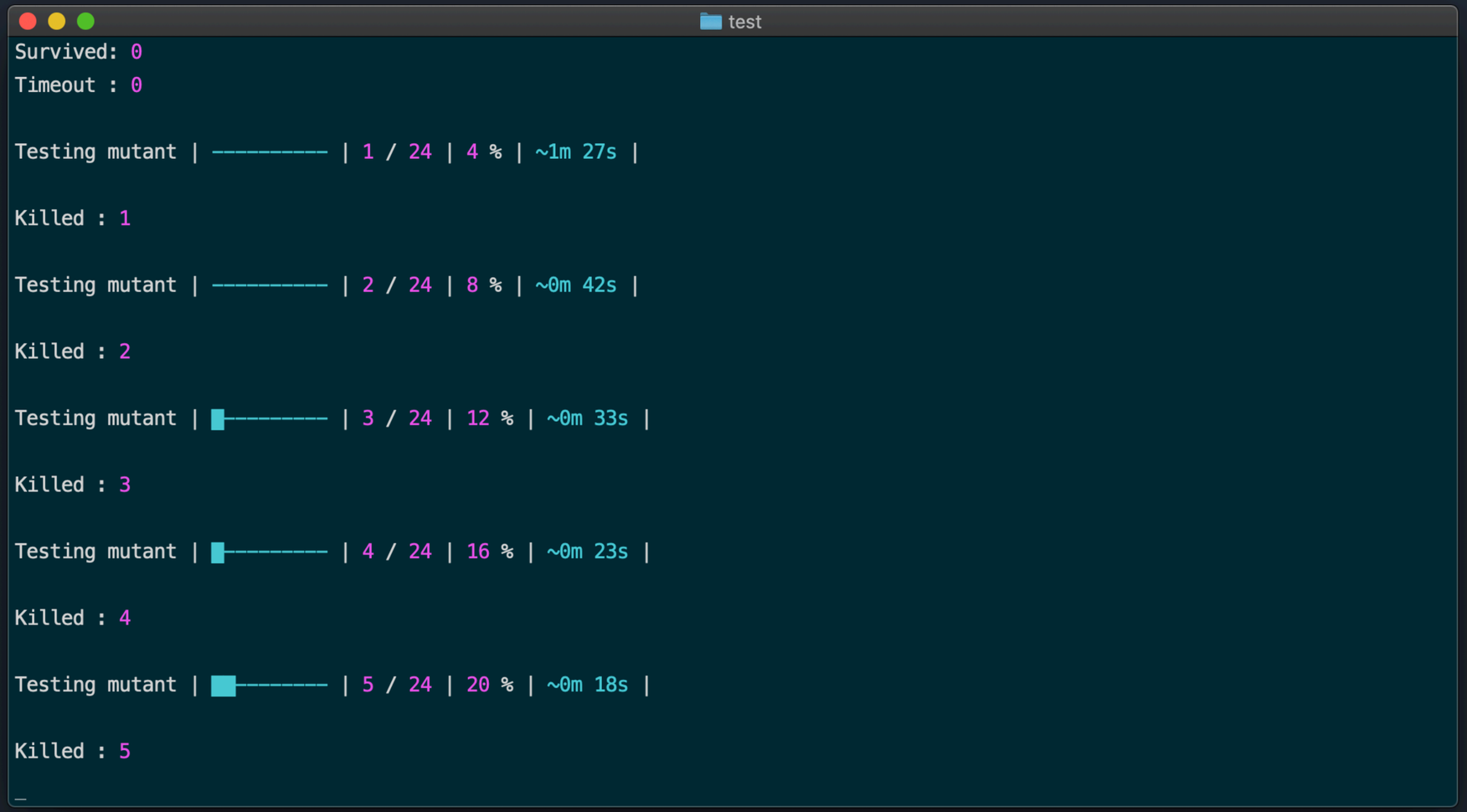

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

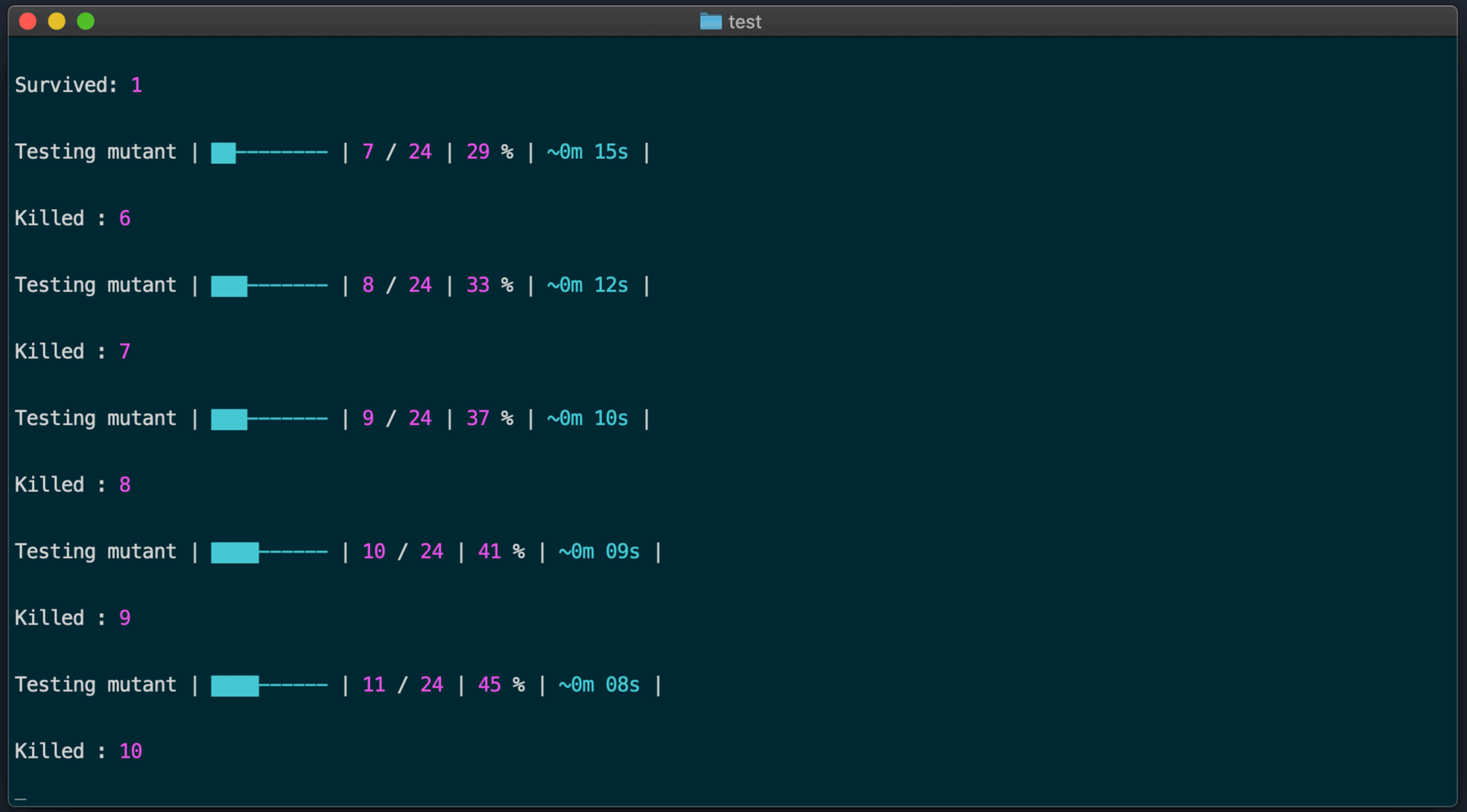

Workspaces
kroniak ~ Workspaces dotnet tool install -g dotnet-stryker
You can invoke the tool using the following command: dotnet-stryker
Tool 'dotnet-stryker' (version '0.13.0') was successfully installed.
kroniak ~ Workspaces cat ./github/dotnext2019-mutation-analysis/test/stryker-config.json
{
"stryker-config": {
"project-file": "library.csproj",
"reporters": ["progress", "html", "json"],
"max-concurrent-test-runners": 6,
"threshold-high": 80,
"threshold-low": 70,
"threshold-break": 60,
"excluded-mutations": []
}
}
kroniak ~ Workspaces

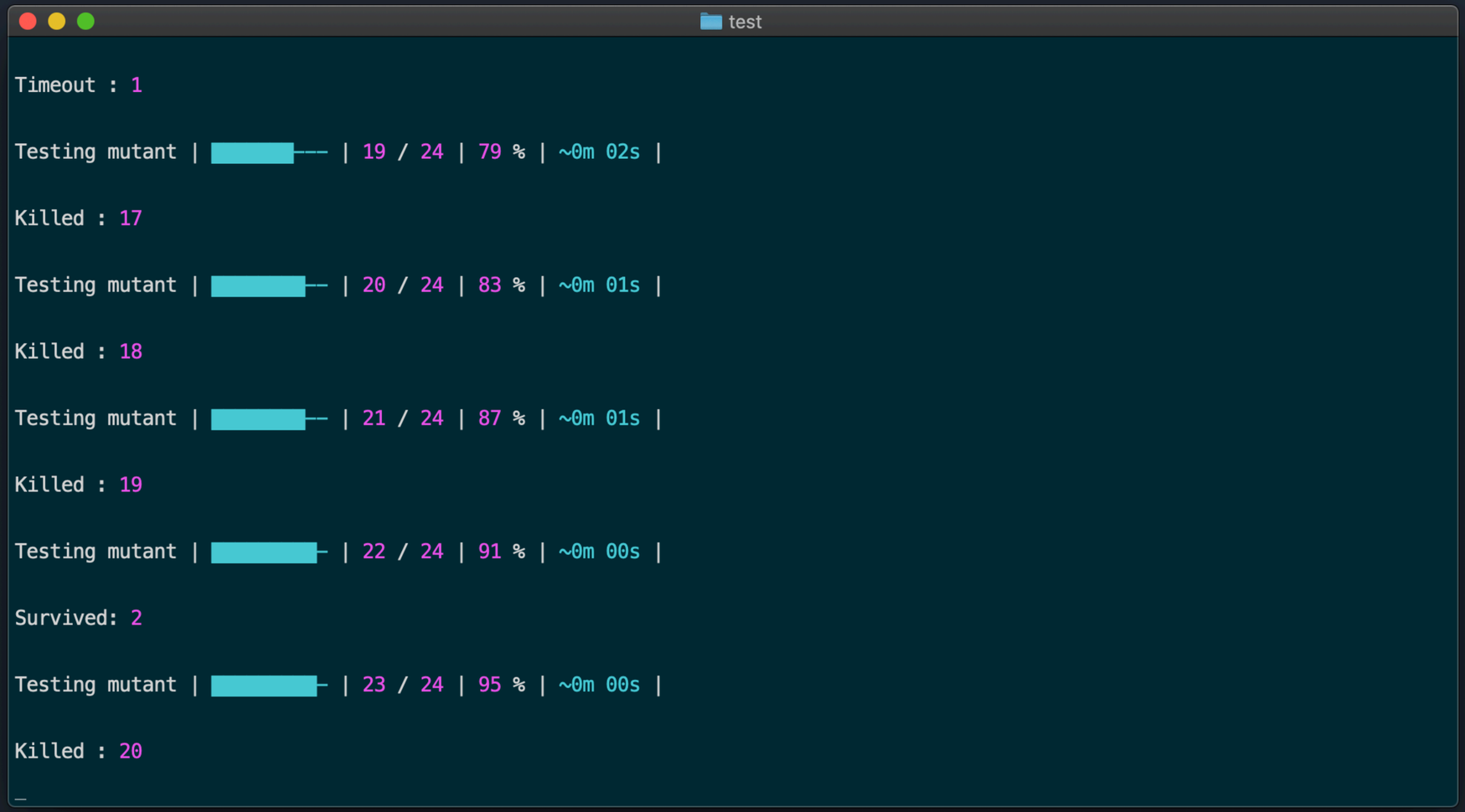

test
kroniak … github dotnext2019-mutation-analysis test master + 5 … 1 dotnet stryker

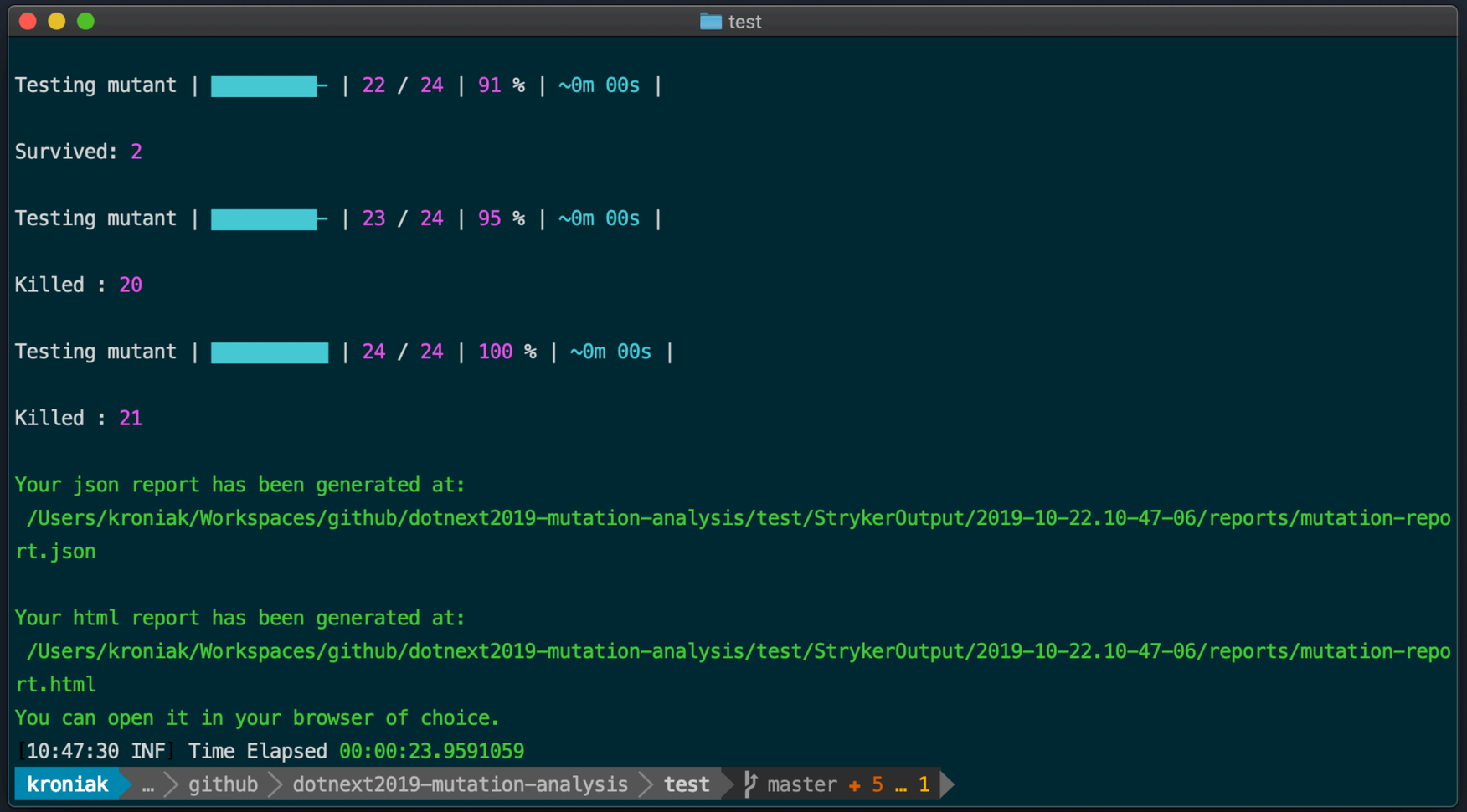

test
kroniak … github dotnext2019-mutation-analysis test master + 5 … 1 dotnet stryker
Version: 0.13.0 (beta)
[10:47:08 INF] The project /Users/kroniak/Workspaces/github/dotnext2019-mutation-analysis/src/library.csproj will be mutated
[10:47:09 INF] Started initial build using dotnet build
[10:47:11 INF] Initial build successful

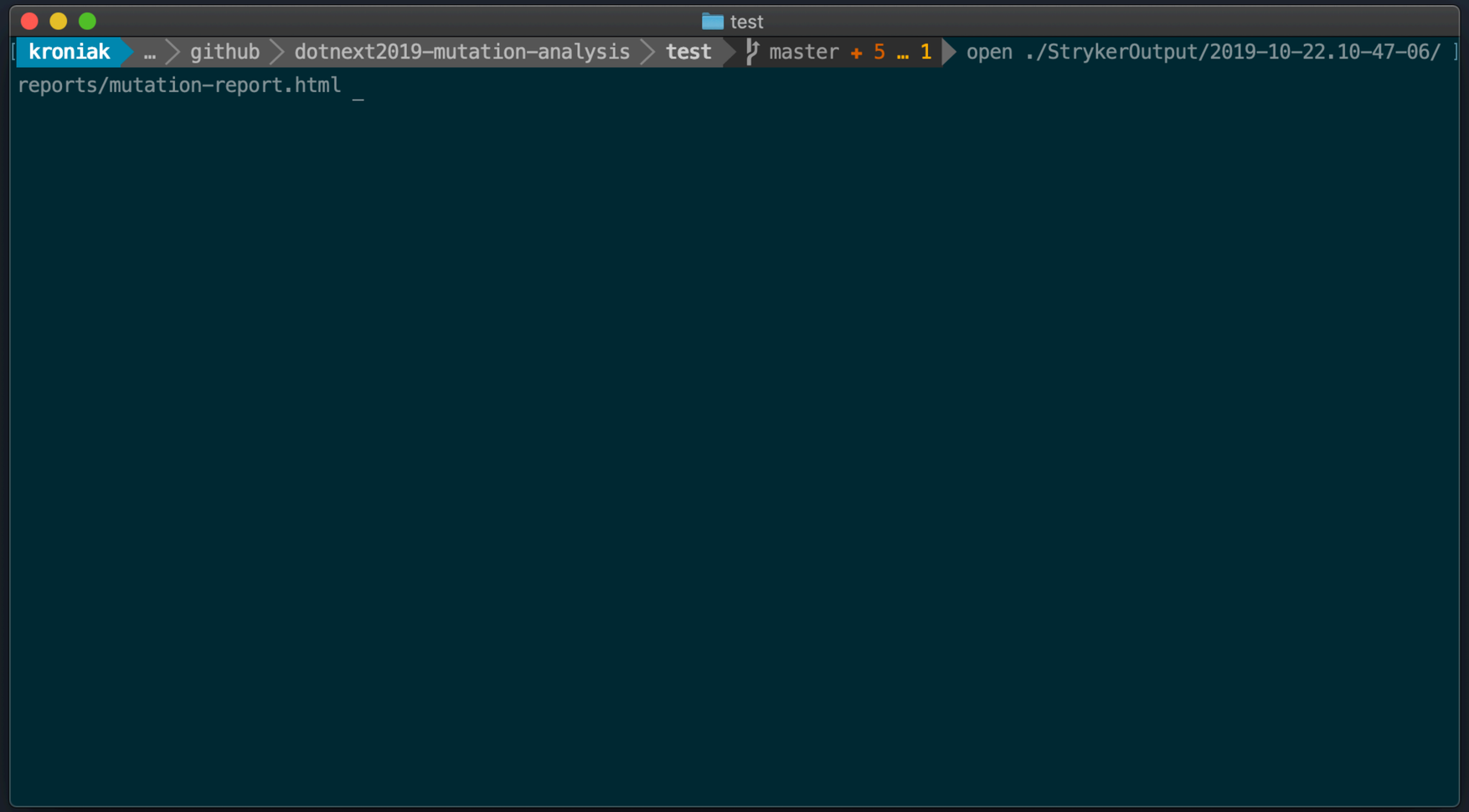

test
kroniak … github dotnext2019-mutation-analysis test master + 5 … 1 dotnet stryker
Version: 0.13.0 (beta)
[10:47:08 INF] The project /Users/kroniak/Workspaces/github/dotnext2019-mutation-analysis/src/library.csproj will be mutated
[10:47:09 INF] Started initial build using dotnet build
[10:47:11 INF] Initial build successful
[10:47:14 INF] Using testrunner VsTest
[10:47:14 INF] Total number of tests found: 17
[10:47:14 INF] Initial testrun started
[10:47:15 INF] Using 6638 ms as testrun timeout
[10:47:18 INF] 26 mutants ready for test
[10:47:18 INF] Capture mutant coverage using 'perTest' mode.


test
Survived: 0
Timeout : 0
Testing mutant | ---------- | 1 / 24 | 4 % | ~1m 27s |
Killed : 1
Testing mutant | ---------- | 2 / 24 | 8 % | ~0m 42s |
Killed : 2
Testing mutant | ---------- | 3 / 24 | 12 % | ~0m 33s |
Killed : 3
Testing mutant | ---------- | 4 / 24 | 16 % | ~0m 23s |
Killed : 4
Testing mutant | ---------- | 5 / 24 | 20 % | ~0m 18s |
Killed : 5


test
Survived: 1
Testing mutant | ▇▇-------- | 7 / 24 | 29 % | ~0m 15s |
Killed : 6
Testing mutant | ▇▇▇------- | 8 / 24 | 33 % | ~0m 12s |
Killed : 7
Testing mutant | ▇▇▇------- | 9 / 24 | 37 % | ~0m 10s |
Killed : 8
Testing mutant | ▇▇▇▇------ | 10 / 24 | 41 % | ~0m 09s |
Killed : 9
Testing mutant | ▇▇▇▇------ | 11 / 24 | 45 % | ~0m 08s |
Killed : 10
_


test
Timeout : 1
Testing mutant | ▇▇▇▇▇▇▇▇--- | 19 / 24 | 79 % | ~0m 02s |
Killed : 17
Testing mutant | ▇▇▇▇▇▇▇▇▇-- | 20 / 24 | 83 % | ~0m 01s |
Killed : 18
Testing mutant | ▇▇▇▇▇▇▇▇▇-- | 21 / 24 | 87 % | ~0m 01s |
Killed : 19
Testing mutant | ▇▇▇▇▇▇▇▇▇▇- | 22 / 24 | 91 % | ~0m 00s |
Survived: 2
Testing mutant | ▇▇▇▇▇▇▇▇▇▇- | 23 / 24 | 95 % | ~0m 00s |
Killed : 20
_


test
Testing mutant | ▇▇▇▇▇▇▇▇▇- | 22 / 24 | 91 % | ~0m 00s |
Survived: 2
Testing mutant | ▇▇▇▇▇▇▇▇▇- | 23 / 24 | 95 % | ~0m 00s |
Killed : 20
Testing mutant | ▇▇▇▇▇▇▇▇▇▇ | 24 / 24 | 100 % | ~0m 00s |
Killed : 21
Your json report has been generated at:
/Users/kroniak/Workspaces/github/dotnext2019-mutation-analysis/test/StrykerOutput/2019-10-22.10-47-06/reports/mutation-report.json
Your html report has been generated at:
/Users/kroniak/Workspaces/github/dotnext2019-mutation-analysis/test/StrykerOutput/2019-10-22.10-47-06/reports/mutation-report.html
You can open it in your browser of choice.
[10:47:30 INF] Time Elapsed 00:00:23.9591059
kroniak … github dotnext2019-mutation-analysis test master + 5 … 1


test
kroniak … github dotnext2019-mutation-analysis test master + 5 … 1 open ./StrykerOutput/2019-10-22.10-47-06/
reports/mutation-report.html _

All files - Stryker.NET Report

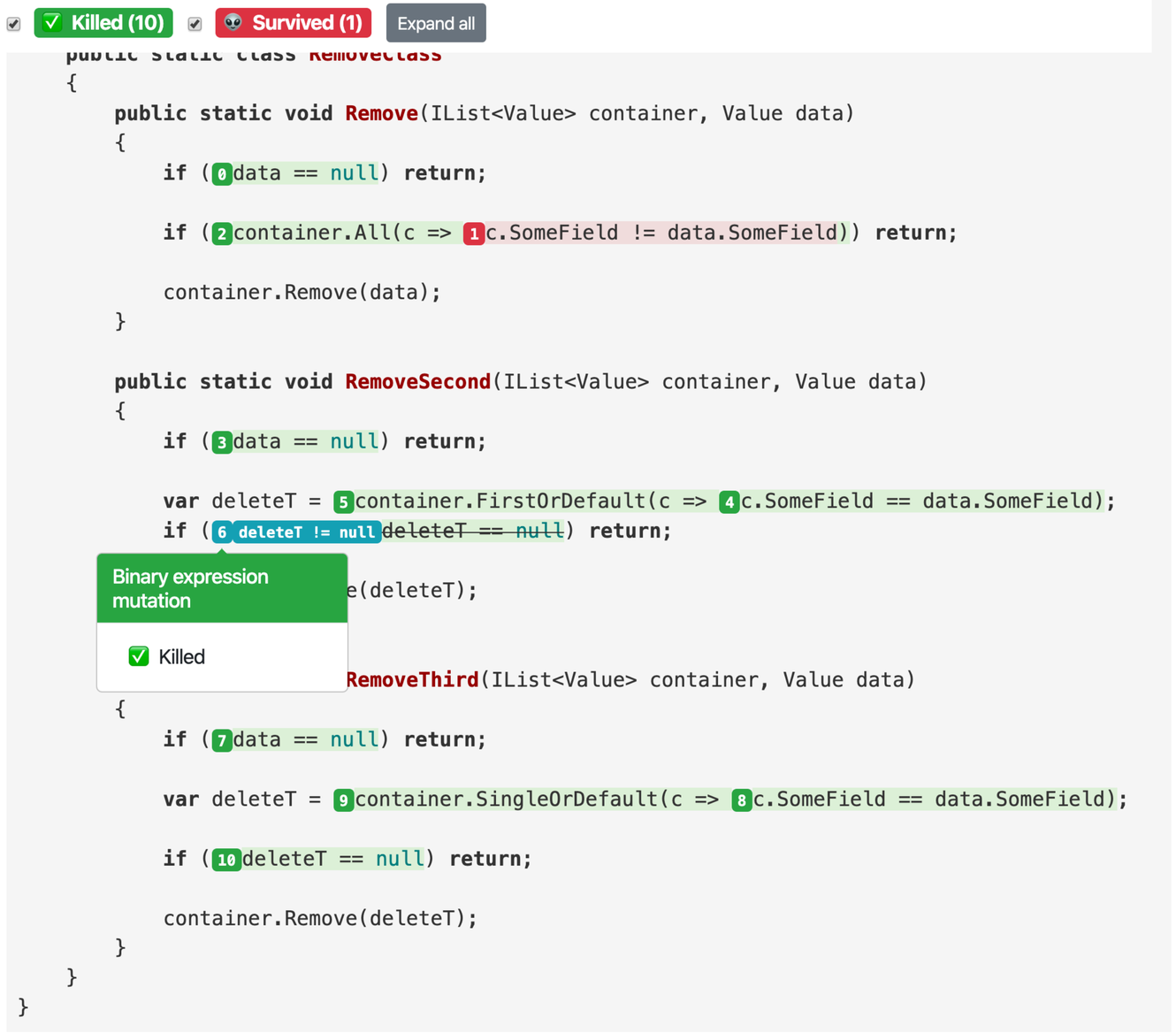
All files

File / Directory	Mutation score		# Killed	# Survived	# Timeout	# No coverage	# Runtime errors	# Compile errors	Total detected	Total undetected	Total mutants
All files	84.62%	84.62	21	2	1	2	0	0	22	4	26
RemoveClass.cs	90.91%	90.91	10	1	0	0	0	0	10	1	11
SampleClass.cs	80.00%	80.00	11	1	1	2	0	0	12	3	15

Killed (10) Survived (1) Expand all

```
    public static class RemoveClass
    {
        public static void Remove(IList<Value> container, Value data)
        {
            if (data == null) return;

            if (container.All(c => 1c.SomeField != data.SomeField)) return;

            container.Remove(data);
        }

        public static void RemoveSecond(IList<Value> container, Value data)
        {
            if (data == null) return;

            var deleteT = container.FirstOrDefault(c => c.SomeField == data.SomeField);
            if (deleteT == null) return;

            container.Remove(deleteT);
        }

        public static void RemoveThird(IList<Value> container, Value data)
        {
            if (data == null) return;

            var deleteT = container.SingleOrDefault(c => c.SomeField == data.SomeField);

            if (deleteT == null) return;

            container.Remove(deleteT);
        }
    }
}
```


Killed (10)
Survived (1)
Expand all
public static class RemoveClass
{
public static void Remove(IList<Value> container, Value data)
{
if (0data == null) return;
if (2container.All(c => 1c.SomeField == data.SomeField c.SomeField != data.SomeField)) r
container.Remov
}
Binary expression mutation
Survived
public static void
lue> container, Value data)
{
if (3data == null) return;
var deleteT = 5container.FirstOrDefault(c => 4c.SomeField == data.SomeField);
if (6deleteT != null deleteT == null) return;
container.Remove(deleteT);
}
public static void RemoveThird(IList<Value> container, Value data)
{
if (7data == null) return;
var deleteT = 9container.SingleOrDefault(c => 8c.SomeField == data.SomeField);
if (10deleteT == null) return;
container.Remove(deleteT);
}
}
}


Killed (10)
Survived (1)
Expand all
public static class RemoveClass
{
public static void Remove(IList<Value> container, Value data)
{
if (0data == null) return;
if (2container.All(c => 1c.SomeField != data.SomeField)) return;
container.Remove(data);
}
public static void RemoveSecond(IList<Value> container, Value data)
{
if (3data == null) return;
var deleteT = 5container.FirstOrDefault(c => 4c.SomeField == data.SomeField);
if (6 deleteT != null deleteT == null) return;
Binary expression mutation
Killed
e(deleteT);
RemoveThird(IList<Value> container, Value data)
{
if (7data == null) return;
var deleteT = 9container.SingleOrDefault(c => 8c.SomeField == data.SomeField);
if (10deleteT == null) return;
container.Remove(deleteT);
}
}
}

ЗАПУСК НА ТОП 5

ТОП 5

1. Json.NET

ТОП 5

1. Json.NET .....

ТОП 5

1. Json.NET ..... FAIL

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ .....

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ ..... FAIL

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ ..... FAIL
3. EntityFrameworkCore

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ ..... FAIL
3. EntityFrameworkCore .....

ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ ..... FAIL
3. EntityFrameworkCore ..... FAIL

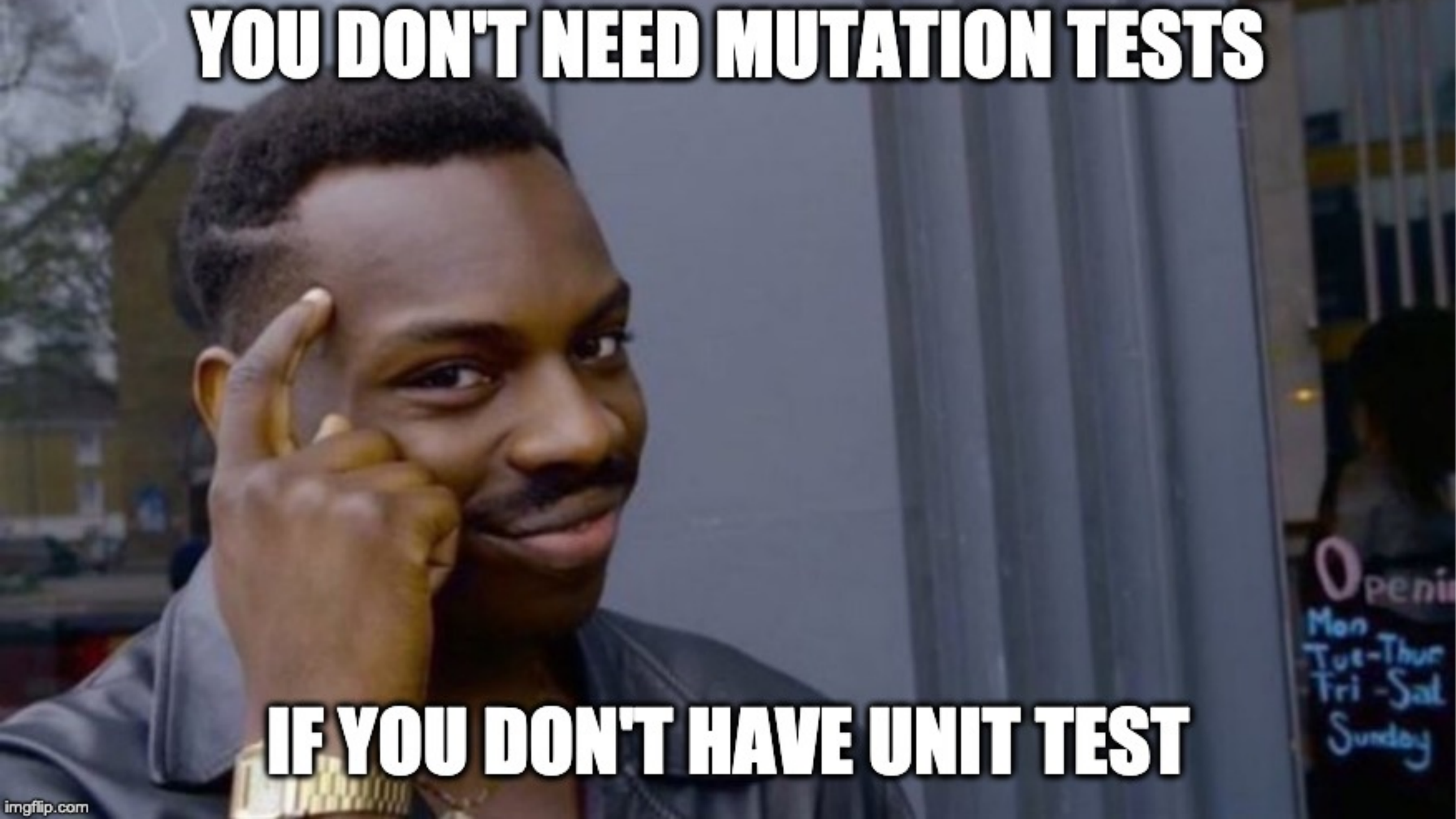
ТОП 5

1. Json.NET ..... FAIL
2. System.LINQ ..... FAIL
3. EntityFrameworkCore ..... FAIL
4. MongoDB Driver ..... FAIL
5. Postgresql Driver ..... FAIL
6. ... other projects ..... FAIL
7. Flurl ..... SUCCESS

КРИТЕРИИ ПРИМЕНИМОСТИ

КРИТЕРИИ ПРИМЕНИМОСТИ

1. Покрытие кода тестами >70%


YOU DON'T NEED MUTATION TESTS
IF YOU DON'T HAVE UNIT TEST
imgflip.com

КРИТЕРИИ ПРИМЕНИМОСТИ

1. Покрытие кода тестами >70%
2. Скорость работы тестов

КРИТЕРИИ ПРИМЕНИМОСТИ

1. Покрытие кода тестами >70%
2. Скорость работы тестов
3. Чистые маленькие методы

КРИТЕРИИ ПРИМЕНИМОСТИ

1. Покрытие кода тестами >70%
2. Скорость работы тестов
3. Чистые маленькие методы
4. .NET Core

КРИТЕРИИ ПРИМЕНИМОСТИ

1. Покрытие кода тестами >70%
2. Скорость работы тестов
3. Чистые маленькие методы
4. .NET Core
5. Сила духа и авантюризм

ВЫВОДЫ

ВЫВОДЫ

MDD
классно подходит при работе

ВЫВОДЫ

MDD
классно подходит при работе
с парадигмой TDD
при написании НОВЫХ тестов

ВЫВОДЫ

MDD
классно подходит при работе
с парадигмой TDD
при написании НОВЫХ тестов
и
однократного
аудита текущей системы

ВЫВОДЫ

для поиска
неочевидных
проблемных мест

ВЫВОДЫ

для поиска
неочевидных
проблемных мест
и
оптимизации юнит тестов

ВЫВОДЫ

MDD подходит
для
аудита кода
внешних разработчиков

ВЫВОДЫ

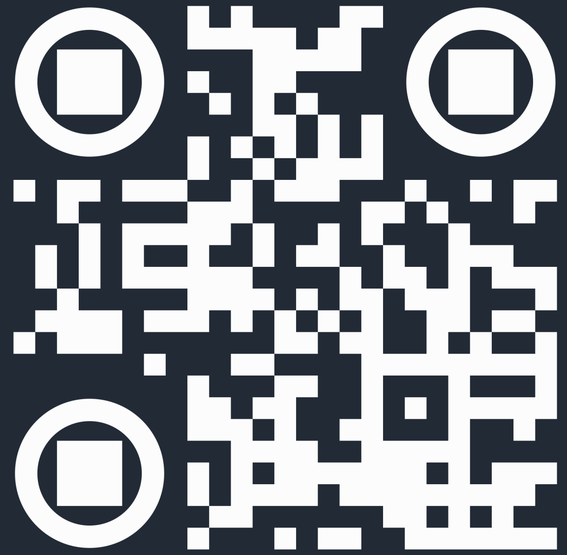
MDD
подходит
для
проверки себя
и
внутренних разработчиков

ВЫВОДЫ

MDD
подходит
для
погружения новичков
в проект

библиотека

http://flurl.dev



примеры кода и ссылки на книги

https://github.com/kroniak/dotnext2019-mutation-analysis

ССЫЛКИ

https://d-nb.info/1051432480/34

Assessing Test Quality by David Schuler

https://leanpub.com/mutationtesting

Mutation Testing by Filip van Laenen

https://1drv.ms/u/s!AswfoxlkvkXGgdVOQBvKmoGlQhPzmQ

http://tiny.cc/4d88ez

This presentation by me

CHANGE

Q & A