**План тестирования для игры «Морской бой», написанный на c# с использованием WinForm**

Для того, чтобы компьютер играл автоматически, были прописаны две функции:

1. Расстановка кораблей рандом(автоматом)
2. Удалить все корабли
3. Генерация выстрелов со стороны компьютера

Для человека были написаны следующие классы и функции:

1. Направления кораблей
   1. Функция Direction показывает в каком направлении могут стоять корабли: вертикальный или горизонтальный
2. Выстрелы
   1. Получаем округленные очки
3. Обрисовка поля
   1. Данный класс отвечает за расстановку кораблей (Получаем корабль, выставляем корабль на поле, изменяем направления корабля, и решения конфликтов, связанных с расстановкой кораблей)
4. Игра
   1. Данный класс создает двух игроков для дальнейшей игры, а также результаты данной игры
5. Параметры игры
   1. Создает корабли(флот)
6. Этапы игры
   1. Описывает этапы игры: начало (ожидает нажатия кнопки «Начать игру», добавление кораблей, сама игра(выстрелы), финиш (кто выиграл))
7. Поле
   1. Отлеживает обрисовку всего
8. Игрок
   1. Получает имя игроков
9. Корабли
   1. Получает размеры, позиции и направление кораблей
10. Игроки
    1. Создаем имя игрока
11. Выстрелы
    1. Обрисовываем на поле выстрелы по координатам
12. Корабли
    1. Отслеживаем позиции кораблей и обрисовываем их
13. Результаты выстрелов
    1. Выявляем победителя, проигравшего или ничью

**Тесты**

В AssemblyInfo прописываем данные о сборке и версии ассмблера

**Аттестационные тесты**

Тест 1: Проверяем класс выстрелов, для этого подаем на вход очки, результатом теста должно быть выведены те точки, на которых есть очки (в данном тесте это (2,5))

Ожидаемый результат: поле с синим кружком в точку 2,5

lic class ExtensionsSpecification

{

[Test]

public void GetRoundPoints\_ShouldReturnRound\_WhenAnyPoint()

{

var point = new Point(2, 5);

point.GetRoundPoints().Should().BeEquivalentTo(

new Point(1, 4), new Point(2, 4), new Point(3, 4),

new Point(1, 5), new Point(2, 5), new Point(3, 5),

new Point(1, 6), new Point(2, 6), new Point(3, 6));

}

}

}

Тест 2. Проверяем классы добавления и получения кораблей. Для этого создаем поле размером 10\*10, и пытаемся с помощью тестов добавить туда кораблики размера 1,3,4. Результат поле с расставленными кораблями

public class FieldAddingShipsSpecification

{

private Field field;

private Ship ship;

private Ship bigShip;

private Ship smallShip;

[SetUp]

public void SetUp()

{

field = new Field(10, 10);

ship = new Ship(3);

bigShip = new Ship(4);

smallShip = new Ship(1);

}

[Test]

public void GetShips\_ShouldReturnNothing\_WhenNotAdded()

{

field.GetShips().Should().BeEmpty();

}

[Test]

public void GetShips\_ShouldReturnAddedShips()

{

field.AddShip(ship);

field.AddShip(bigShip);

field.AddShip(smallShip);

field.GetShips().Should().BeEquivalentTo(ship, bigShip, smallShip);

}

[Test]

public void GetShips\_ShouldNotReturnDuplicates()

{

field.AddShip(ship);

field.AddShip(ship);

field.GetShips().Should().BeEquivalentTo(ship);

}

}

Тест 3.Для того, чтобы проверить расстановку кораблей автоматически, мы создаем поле, создаем наши флотилии, и начинаем добавлять корабли. Результатом данного теста должной быть поле с кораблями, расставленными автоматически.

public class FieldExtensionsSpecification

{

[Test]

public void ArrangeShipsAutomatically\_ShouldArrange()

{

var options = new GameOptions();

var field = options.CreateField();

var fleet = options.CreateFleet();

foreach (var ship in fleet)

field.AddShip(ship);

var result = field.ArrangeShipsAutomatically();

result.Should().BeTrue();

}

}

}

**Модульные тесты:**

**Позитивные:1-5,7,8,10,13,14,17-19,21-26,29**

**Негативные:6,9,11,12,15,16,20,27,28**

Для Кораблей:

[TestFixture]

public class FieldPuttingShipsSpecification

{

private Field field;

private Ship ship;

private Ship bigShip;

private Ship smallShip;

[SetUp]

public void SetUp()

{

ship = new Ship(3);

bigShip = new Ship(4);

smallShip = new Ship(1);

field = new Field(10, 10);

field.AddShip(ship);

field.AddShip(bigShip);

field.AddShip(smallShip);

}

// GetShipToPutOrNull

Тест 1 // первым предлагается к размещению самый большой корабль

Входные данные: поле

Ожидаемый результат: первый ставиться большой корабль

[Test]

public void GetShipToPutOrNull\_ShouldReturnBiggestShip\_WhenItIsNotPut()

{

field.GetShipToPutOrNull().Should().Be(bigShip);

}

Тест 2 // предлагается самый большой корабль из оставшихся, если часть кораблей размещено

Входные данные: большой корабль

Ожидаемый результат: корабль с позицией

[Test]

public void GetShipToPutOrNull\_ShouldReturnSecondBiggestShip\_WhenBiggestIsPut()

{

bigShip.Position = new Point(0, 0);

field.GetShipToPutOrNull().Should().Be(ship);

}

Тест 3 // кораблей нет, размещать нечего

Входные данные: поле 10\*10

Ожидаемый результат: просто отрисованное поле

[Test]

public void GetShipToPutOrNull\_ShouldReturnNull\_WhenNoShips()

{

var field = new Field(10, 10);

field.GetShipToPutOrNull().Should().BeNull();

}

Тест 4 // все корабли размещены

Входные данные: три корабля с позициями

Ожидаемый результат: ничего

[Test]

public void GetShipToPutOrNull\_ShouldReturnNull\_WhenAllShipsPut()

{

ship.Position = new Point(0, 0);

bigShip.Position = new Point(0, 0);

smallShip.Position = new Point(0, 0);

field.GetShipToPutOrNull().Should().BeNull();

}

// > PutShip

Тест 5 // просто разместить, проверить, что разместился в правильной точке

Входные данные: два корабля с позициями

Ожидаемый результат: позицию корабля

[Test]

public void PutShip\_ShouldSetShipPosition()

{

var point = new Point(2, 5);

field.PutShip(ship, point).Should().BeTrue();

ship.Position.Should().Be(point);

}

Тест 6 // размещение за пределами (горизонталь и вертикаль, меньшая и большая границы)

Входные данные: два корабля с позициями

Ожидаемый результат: ничего

[Test]

public void PutShip\_ShouldThrow\_WhenShipNotAdded()

{

var otherShip = new Ship(3);

var point = new Point(2, 5);

Action action = () => field.PutShip(otherShip, point);

action.Should().Throw<InvalidOperationException>();

otherShip.Position.Should().BeNull();

}

Тест 7 // размещение на границе пределов (горизонталь и вертикаль, меньшая и большая границы)

Входные данные: два корабля с позициями

Ожидаемый результат: позицию корабля

[TestCase(0, 5, Direction.Vertical)]

[TestCase(9, 5, Direction.Vertical)]

[TestCase(0, 5, Direction.Horizontal)]

[TestCase(7, 5, Direction.Horizontal)]

[TestCase(2, 0, Direction.Horizontal)]

[TestCase(2, 9, Direction.Horizontal)]

[TestCase(2, 0, Direction.Vertical)]

[TestCase(2, 7, Direction.Vertical)]

public void PutShip\_ShouldSetShipPosition\_WhenJustBeforeOutOfTheBounds(int x, int y, Direction direction)

{

var point = new Point(x, y);

ship.Direction = direction;

field.PutShip(ship, point).Should().BeTrue();

ship.Position.Should().Be(point);

}

Тест 8 // размещение размещенного корабля в допустимую позицию

[TestCase(-1, 5, Direction.Vertical)]

[TestCase(10, 5, Direction.Vertical)]

[TestCase(-1, 5, Direction.Horizontal)]

[TestCase(8, 5, Direction.Horizontal)]

[TestCase(2, -1, Direction.Horizontal)]

[TestCase(2, 10, Direction.Horizontal)]

[TestCase(2, -1, Direction.Vertical)]

[TestCase(2, 8, Direction.Vertical)]

public void PutShip\_ShouldSetShipPosition\_WhenOutOfTheBounds(int x, int y, Direction direction)

{

var point = new Point(x, y);

ship.Direction = direction;

field.PutShip(ship, point).Should().BeFalse();

ship.Position.Should().BeNull();

}

Тест 9 // размещенного корабля в недопустимую позицию (не перемещается, false)

Входные данные: два корабля с позициями

Ожидаемый результат: последняя позиция

[Test]

public void PutShip\_ShouldResetShipPosition\_WhenInside()

{

var point = new Point(2, 5);

field.PutShip(ship, point);

var secondPoint = new Point(3, 6);

field.PutShip(ship, secondPoint).Should().BeTrue();

ship.Position.Should().Be(secondPoint);

}

// > GetShipsAt

Тест 10 // корабли не отслеживаются в точке, если не добавлены

Входные данные: два корабля с позициями

Ожидаемый результат: ничего

[Test]

public void PutShip\_ShouldRemoveShipPosition\_WhenOut()

{

var point = new Point(2, 5);

field.PutShip(ship, point).Should().BeTrue();

var secondPoint = new Point(10, 5);

field.PutShip(ship, secondPoint).Should().BeFalse();

ship.Position.Should().BeNull();

}

Тест 11 // размещенный корабль должен отслеживаться GetShipAt (горизонталь и вертикаль, в любой из точек)

[Test]

public void GetShipsAt\_ShouldReturnNothing\_WhenNoShips()

{

var point = new Point(2, 5);

field.GetShipsAt(point).Should().BeEmpty();

}

Тест 12 // размещенный корабль не должен отслеживаться в остальных точках (горизонталь и вертикаль, горизонтальны и вертикальный корабль)

Входные данные: два корабля с позициями

Ожидаемый результат: корабль со своей позицией

[TestCase(0, 0, Direction.Horizontal)]

[TestCase(1, 0, Direction.Horizontal)]

[TestCase(2, 0, Direction.Horizontal)]

[TestCase(0, 0, Direction.Vertical)]

[TestCase(0, 1, Direction.Vertical)]

[TestCase(0, 2, Direction.Vertical)]

public void GetShipsAt\_ShouldReturnShip\_WhenPointInShip(int dx, int dy, Direction direction)

{

ship.Direction = direction;

var point = new Point(2, 5);

field.PutShip(ship, point);

var deltaPoint = new Point(point.X + dx, point.Y + dy);

field.GetShipsAt(deltaPoint).Should().BeEquivalentTo(ship);

}

Тест 13 // неудачно размещенный корабль не должен отслеживаться

Входные данные: два корабля с позициями

Ожидаемый результат: ничего

[TestCase(-1, 0, Direction.Horizontal)]

[TestCase(3, 0, Direction.Horizontal)]

[TestCase(0, -1, Direction.Horizontal)]

[TestCase(0, 1, Direction.Horizontal)]

[TestCase(-1, 0, Direction.Vertical)]

[TestCase(1, 0, Direction.Vertical)]

[TestCase(0, -1, Direction.Vertical)]

[TestCase(0, 3, Direction.Vertical)]

public void GetShipsAt\_ShouldReturnNothing\_WhenPointOutOfShip(int dx, int dy, Direction direction)

{

ship.Direction = direction;

var point = new Point(2, 5);

field.PutShip(ship, point);

var deltaPoint = new Point(point.X + dx, point.Y + dy);

field.GetShipsAt(deltaPoint).Should().BeEmpty();

}

Тест 14 // несколько кораблей в точке отслеживаются, сортировка по размеру (сначала более короткие)

[Test]

public void GetShipsAt\_ShouldReturnNothing\_AfterPutFailed()

{

var point = new Point(8, 5);

ship.Direction = Direction.Horizontal;

field.PutShip(ship, point);

field.GetShipsAt(point).Should().BeEmpty();

}

Тест 15 // \* убранный корабль должен перестать отслеживаться

Входные данные: корабль с позициями

Ожидаемый результат: получаем корабль со второй позицией

[Test]

public void GetShipsAt\_ShouldReturnShip\_AfterReset()

{

var point = new Point(2, 5);

ship.Direction = Direction.Horizontal;

field.PutShip(ship, point);

var secondPoint = new Point(3, 6);

field.PutShip(ship, secondPoint);

field.GetShipsAt(secondPoint).Should().BeEquivalentTo(ship);

}

Тест 16 // \* перемещенный корабль должен отслеживаться в новой точке

Входные данные: корабль с несколькими позициями

Ожидаемый результат: ничего

[Test]

public void GetShipsAt\_ShouldReturnNothing\_AfterRemove()

{

var point = new Point(2, 5);

ship.Direction = Direction.Horizontal;

field.PutShip(ship, point);

var secondPoint = new Point(10, 5);

field.PutShip(ship, secondPoint);

field.GetShipsAt(secondPoint).Should().BeEmpty();

}

Тест 17 // \* дважды добавленный корабль отслеживается только один раз

Входные данные: корабль с позицией добавляем дважды

Ожидаемый результат: получаем на выход корабль один раз

[Test]

public void GetShipsAt\_ShouldReturnShipsFromSmallToBig\_WhenSeveralShips()

{

var point = new Point(2, 5);

ship.Direction = Direction.Horizontal;

bigShip.Direction = Direction.Vertical;

field.PutShip(ship, point);

field.PutShip(smallShip, point);

field.PutShip(bigShip, point);

field.GetShipsAt(point)

.Should().BeEquivalentTo(

new[] { smallShip, ship, bigShip },

config => config.WithStrictOrdering());

}

[Test]

public void GetShipsAt\_ShouldReturnOneShip\_WhenPutSeveralTimes()

{

var point = new Point(2, 5);

field.PutShip(ship, point);

field.PutShip(ship, point);

field.GetShipsAt(point).Should().BeEquivalentTo(ship);

}

// > ChangeShipDirection

Тест 18 // ничего не делает, если корабль не размещен

Входные данные: корабль

Ожидаемый результат: поле отрисованное

[Test]

public void ChangeShipDirection\_ShouldThrow\_WhenShipNotAdded()

{

var otherShip = new Ship(3);

Action action = () => field.ChangeShipDirection(otherShip);

action.Should().Throw<InvalidOperationException>();

}

Тест 19 // изменение направления просто меняет его

Входные данные: корабль с позиций и направлением

Ожидаемый результат: изменено направление корабля

[Test]

public void ChangeShipDirection\_ShouldDoNoting\_WhenNotPut()

{

ship.Direction = Direction.Horizontal;

field.ChangeShipDirection(ship).Should().BeFalse();

ship.Direction.Should().Be(Direction.Horizontal);

}

Тест 20 // корабль отодвигается для крайне правого вертикального

Входные данные: два корабля с позициями

Ожидаемый результат: получаем позицию корабля

[TestCase(Direction.Horizontal, Direction.Vertical)]

[TestCase(Direction.Vertical, Direction.Horizontal)]

public void ChangeShipDirection\_ShouldJustChangeDirection\_WhenEnoughSpace(Direction beforeDirection, Direction afterDirection)

{

var point = new Point(2, 5);

ship.Direction = beforeDirection;

field.PutShip(ship, point);

field.ChangeShipDirection(ship).Should().BeTrue();

ship.Direction.Should().Be(afterDirection);

ship.Position.Should().Be(point);

}

Тест 21 // корабль отодвигается для крайне низкого горизонтального

Входные данные: два корабля с позициями

Ожидаемый результат: новую позицию корабля

[TestCase(9, 5, Direction.Vertical, 7, 5)]

[TestCase(8, 5, Direction.Vertical, 7, 5)]

[TestCase(2, 9, Direction.Horizontal, 2, 7)]

[TestCase(2, 8, Direction.Horizontal, 2, 7)]

public void ChangeShipDirection\_ShouldMoveShip\_WhenNotEnoughSpace(int x, int y, Direction direction,

int newX, int newY)

{

var point = new Point(x, y);

ship.Direction = direction;

field.PutShip(ship, point);

field.ChangeShipDirection(ship).Should().BeTrue();

ship.Position.Should().Be(new Point(newX, newY));

}

Тест 22 // если отодвинуться не получается, то корабль удаляется с поля и возвращается false

Входные данные: два корабля с позициями

Ожидаемый результат: ничего не получаем

[Test]

public void ChangeShipDirection\_ShouldRemoveShip\_WhenCanNotMoveHorizontalShip()

{

var field = new Field(3, 2);

var ship = new Ship(3);

field.AddShip(ship);

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(0, 0));

field.ChangeShipDirection(ship).Should().BeFalse();

ship.Position.Should().BeNull();

}

[Test]

public void ChangeShipDirection\_ShouldRemoveShip\_WhenCanNotMoveVerticalShip()

{

var field = new Field(2, 3);

var ship = new Ship(3);

field.AddShip(ship);

ship.Direction = Direction.Vertical;

field.PutShip(ship, new Point(0, 0));

field.ChangeShipDirection(ship).Should().BeFalse();

ship.Position.Should().BeNull();

}

// > GetConflictingPoints

Тест 23 // нет, если нет кораблей

Входные данные: два корабля с позициями

Ожидаемый результат: ничего

[Test]

public void GetConflictingPoints\_ShouldBeEmpty\_WhenNoShips()

{

field.GetConflictingPoints().Should().BeEmpty();

}

Тест 24 // нет, если корабль один

Входные данные: корабль с позицией

Ожидаемый результат: ничего не выводит кроме корабля

[Test]

public void GetConflictingPoints\_ShouldBeEmpty\_WhenOneShip()

{

var point = new Point(2, 5);

field.PutShip(ship, point);

field.GetConflictingPoints().Should().BeEmpty();

}

Тест 25 // полное перекрытие в основной точке

Входные данные: два корабля с позициями

Ожидаемый результат: точки перекрытия

[Test]

public void GetConflictingPoints\_ShouldReturn3Points\_WhenMainPointConflict()

{

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(2, 5));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(2, 5));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(2, 5),

new Point(3, 5),

new Point(2, 6));

}

[Test]

public void GetConflictingPoints\_ShouldReturn3Points\_WhenBackPointConflict()

{

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(2, 8));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(4, 5));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(4, 8),

new Point(3, 8),

new Point(4, 7));

}

Тест 26 // полное перекрытие в точке окончаний

Входные данные: два корабля с позициями и размерами

Ожидаемый результат: конфликт в двух точках

[Test]

public void GetConflictingPoints\_ShouldReturn2Points\_WhenMainPointTouch()

{

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(3, 5));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(2, 6));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(3, 5),

new Point(2, 6));

}

Тест 27 // касание рядом с основной точкой

Входные данные: два корабля с позициями

Ожидаемый результат: точки пересечения

[Test]

public void GetConflictingPoints\_ShouldReturn2Points\_WhenBackPointTouch()

{

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(1, 8));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(4, 4));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(3, 8),

new Point(4, 7));

}

Тест 28 // касание рядом с точкой окончаний

Входные данные: два корабля с позициями

Ожидаемый результат: все точки пересечения

[Test]

public void GetConflictingPoints\_ShouldReturnPointsOnce\_WhenSeveralShipsInConflict()

{

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(2, 5));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(2, 5));

smallShip.Direction = Direction.Horizontal;

field.PutShip(smallShip, new Point(2, 5));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(2, 5),

new Point(3, 5),

new Point(2, 6));

}

Тест 29 // несколько кораблей

Входные данные: два корабля с позициями

Ожидаемый результат: точки расстановки и конфликта пересечения

[Test]

public void GetConflictingPoints\_ShouldAllPoints\_WhenSeveralConflicts()

{

var ship1 = new Ship(1);

var ship2 = new Ship(2);

field.AddShip(ship1);

field.AddShip(ship2);

ship.Direction = Direction.Horizontal;

field.PutShip(ship, new Point(2, 5));

bigShip.Direction = Direction.Vertical;

field.PutShip(bigShip, new Point(2, 5));

field.PutShip(ship1, new Point(4, 5));

field.PutShip(ship2, new Point(2, 8));

field.GetConflictingPoints()

.Should().BeEquivalentTo(

new Point(2, 5),

new Point(2, 6),

new Point(2, 7),

new Point(2, 8),

new Point(3, 8),

new Point(3, 5),

new Point(4, 5));

}

}

**Интеграционные тесты**

для класса Игра:

Тест 1.Проверяем, начал ли игру пользователь

Входные параметры: Начинаем игру, задаем параметры игры, создаем корабль с размером 3, вызываем этапы игры и проверяем как будет проходить.

Ожидаемый результат: игра прошла успешно и заканчивается.

С помощью конструктора создаем игру и двух играков

public void Constructor\_ShouldCreateNotStarted()

{

var game = new Game();

game.Stage.Should().Be(GameStage.NotStarted);

game.FirstPlayer.Should().BeNull();

game.SecondPlayer.Should().BeNull();

game.CurrentPlayer.Should().BeNull();

}

[Test]

public void Game\_ShouldPlay()

{

var game = new Game(options =>

{

options.SetShipSizes(3);

options.Width = 10;

options.Height = 10;

});

game.Start("first", "second");

game.Stage.Should().Be(GameStage.ArrangingShips);

var ship1 = game.CurrentPlayer.Field.GetShipToPutOrNull();

game.CurrentPlayer.Field.PutShip(ship1, new Point(2, 5));

game.EndArrangingCurrentPlayerShips();

game.Stage.Should().Be(GameStage.ArrangingShips);

var ship2 = game.CurrentPlayer.Field.GetShipToPutOrNull();

game.CurrentPlayer.Field.PutShip(ship2, new Point(2, 5));

game.EndArrangingCurrentPlayerShips();

game.Stage.Should().Be(GameStage.Battle);

game.ShootTo(new Point(2, 5));

game.ShootTo(new Point(3, 5));

game.ShootTo(new Point(4, 5));

game.Stage.Should().Be(GameStage.Finished);

}

Для игры компьютера:

Проверяем работу алгоритма автоматического выстрела

Для этого создаем корабли и расстанавливаем их на некоторые позиции

public class FieldShootingSpecification

{

private Field field;

private Ship ship;

private Ship bigShip;

private Ship smallShip;

[SetUp]

public void SetUp()

{

ship = new Ship(3);

bigShip = new Ship(4);

smallShip = new Ship(1);

field = new Field(10, 10);

field.AddShip(ship);

field.AddShip(bigShip);

field.AddShip(smallShip);

}

Тест 2. проверяем промах

Входные параметры: корабль с позицией и точка выстрела

Ожидаемый результат: выстрел пропущен, проставлена на этом месте поля точку

[TestCase(1, 5)]

[TestCase(5, 5)]

[TestCase(2, 4)]

[TestCase(2, 6)]

public void ShootTo\_ShouldReturnMiss\_WhenMiss(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint)

.Should().Be(ShotResult.Miss);

}

[TestCase(1, 5)]

[TestCase(5, 5)]

[TestCase(2, 4)]

[TestCase(2, 6)]

public void GetShots\_ShouldReturnOneShot\_WhenMiss(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.GetShots()

.Should().BeEquivalentTo(shotPoint);

}

Тест 3: Проверяем повторный пропуск выстрела

Входные параметры: корабль с позицией и точка выстрела

Ожидаемый результат: получаем точку где пропущен выстрел, ставим просто точку

[TestCase(1, 5)]

[TestCase(5, 5)]

[TestCase(2, 4)]

[TestCase(2, 6)]

public void ShootTo\_ShouldReturnCancel\_WhenMissAgain(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.ShootTo(shotPoint)

.Should().Be(ShotResult.Cancel);

}

[TestCase(1, 5)]

[TestCase(5, 5)]

[TestCase(2, 4)]

[TestCase(2, 6)]

public void GetShots\_ShouldNotDuplicateShot\_WhenMissAgain(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.ShootTo(shotPoint);

field.GetShots()

.Should().BeEquivalentTo(shotPoint);

}

Тест 4: Проверяем попадание выстрела

Входные параметры: корабль с позицией и точка выстрела

Ожидаемый результат: получаем закрашенный квадрат , ставим взрыв, отправляем пропуск хода соседнего игрока

[TestCase(2, 5)]

[TestCase(3, 5)]

[TestCase(4, 5)]

public void ShootTo\_ShouldReturnHit\_WhenHit(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint)

.Should().Be(ShotResult.Hit);

}

[TestCase(2, 5)]

[TestCase(3, 5)]

[TestCase(4, 5)]

public void GetShots\_ShouldReturnOneShot\_WhenHit(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.GetShots()

.Should().BeEquivalentTo(shotPoint);

}

Тест 5: Проверяем попадание выстрела

Входные параметры: корабль с позицией и точка выстрела

Ожидаемый результат: получаем закрашенный квадрат , ставим взрыв

[TestCase(2, 5)]

[TestCase(3, 5)]

[TestCase(4, 5)]

public void ShootTo\_ShouldReturnCancel\_WhenHitAgain(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.ShootTo(shotPoint)

.Should().Be(ShotResult.Cancel);

}

[TestCase(2, 5)]

[TestCase(3, 5)]

[TestCase(4, 5)]

public void GetShots\_ShouldNotDuplicateShot\_WhenHitAgain(int x, int y)

{

field.PutShip(ship, new Point(2, 5));

var shotPoint = new Point(x, y);

field.ShootTo(shotPoint);

field.ShootTo(shotPoint);

field.GetShots()

.Should().BeEquivalentTo(shotPoint);

}

Для кораблей:

Создаем корабль размером 3

Получаем его позицию

Перемещаем корабль по направлению и положению

public void Constructor\_ShouldCreateNotPlacedHorizontalShip()

{

var ship = new Ship(3);

ship.Size.Should().Be(3);

ship.Direction.Should().Be(Direction.Horizontal);

ship.Position.Should().BeNull();

}

Тест 6: Проверяем не расстановку корабля

Входные параметры: корабль 3х палубный

Ожидаемый результат: ничего не получаем так как нет выбранного места для корабля

[Test]

public void GetPositionPoints\_ShouldReturnNothing\_WhenNotPlaced()

{

var ship = new Ship(3);

ship.Position = null;

ship.GetPositionPoints().Should().BeEmpty();

}

Тест 7: Проверяем расстановку корабля в горизонтальной позиции

Входные параметры: корабль 3х палубный с позицией

Ожидаемый результат: получаем Массив с расположенными кораблем точками

[Test]

public void GetPositionPoints\_ShouldBeOk\_WhenPlacedHorizontally()

{

var ship = new Ship(3);

ship.Position = new Point(2, 5);

ship.Direction = Direction.Horizontal;

ship.GetPositionPoints().Should().BeEquivalentTo(new[] {

new Point(2, 5),

new Point(3, 5),

new Point(4, 5),

});

}

Тест 8: Проверяем расстановку корабля в вертикальной позиции

Входные параметры: корабль 3х палубный с позицией

Ожидаемый результат: получаем Массив с расположенными кораблем точками

[Test]

public void GetPositionPoints\_ShouldBeOk\_WhenPlacedVertically()

{

var ship = new Ship(3);

ship.Position = new Point(2, 5);

ship.Direction = Direction.Vertical;

ship.GetPositionPoints().Should().BeEquivalentTo(new[] {

new Point(2, 5),

new Point(2, 6),

new Point(2, 7),

});

}

}