|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | UUT  (Unit Under Test) | Description | Results |
| UT 1 | testvictory | 게임 승패 처리 기능 테스트 | Pass |
| UT 2 - 1 | testcheck\_33 | 검은 돌 쌍3 제한 기능 테스트 | Pass |
| UT 2 - 2 | testcheck\_44 | 검은 돌 쌍4 제한 기능 테스트 | Pass |
| UT 2 - 3 | testcheck\_6 | 검은 돌 6목 제한 기능 테스트 | Pass |

**import** unittest

**import** copy

**from** Main **import** Main

**class** TestMain(unittest.TestCase):

테스트 케이스로 편의를 위해 크게 오목 판에서 검은 돌로

가로, 세로, 대각선으로 돌을 미리 둔 케이스를 만들었다.

**def** setUp(self):

self.m1 = Main()

self.m2 = Main()

self.m2.omok\_pan[0][0] = 1

self.m2.omok\_pan[0][1] = 1

self.m2.omok\_pan[0][2] = 1

self.m3 = Main()

self.m3.omok\_pan[0][0] = 1

self.m3.omok\_pan[1][0] = 1

self.m3.omok\_pan[2][0] = 1

self.m4 = Main()

self.m4.omok\_pan[0][0] = 1

self.m4.omok\_pan[1][1] = 1

self.m4.omok\_pan[2][2] = 1

self.m5 = Main()

self.m5.omok\_pan[7][8] = 1

self.m5.omok\_pan[7][9] = 1

self.m6 = Main()

self.m6.omok\_pan[6][7] = 1

self.m6.omok\_pan[5][7] = 1

흑돌이나 백돌이 이겨서 게임이 종료되는 경우를 검출해내는 함수를 테스트한다.

**def** testvictory(self):

*# victory return : 1 , else : 0*

*# black win*

검은 돌이 가로 5개를 둬서 이기는 경우

self.m2.omok\_pan[0][3] = 1

self.assertEqual(self.m1.victory\_condition(), 0)

self.m2.omok\_pan[0][4] = 2

self.assertEqual(self.m2.victory\_condition(), 0)

self.m2.omok\_pan[0][4] = 1

self.assertEqual(self.m2.victory\_condition(), 1)

self.m2.omok\_pan[0][5] = 1

self.assertEqual(self.m2.victory\_condition(), 1)

흰 돌이 가로 5개를 둬서 이기는 경우

*#white win*

self.m1.omok\_pan[0][0] = 2

self.assertEqual(self.m1.victory\_condition(), 0)

self.m1.omok\_pan[0][1] = 2

self.assertEqual(self.m1.victory\_condition(), 0)

self.m1.omok\_pan[0][2] = 2

self.assertEqual(self.m1.victory\_condition(), 0)

self.m1.omok\_pan[0][3] = 2

self.assertEqual(self.m1.victory\_condition(), 0)

self.m1.omok\_pan[0][4] = 2

self.assertEqual(self.m1.victory\_condition(), 1)

self.m1.omok\_pan[0][5] = 2

self.assertEqual(self.m1.victory\_condition(), 1)

검은 돌이 대각선 5개를 둬서 이기는 경우

*#대각선*

self.m4.omok\_pan[3][3] = 1

self.assertEqual(self.m4.victory\_condition(), 0)

self.m4.omok\_pan[4][4] = 1

self.assertEqual(self.m4.victory\_condition(), 1)

검은 돌이 세로 5개를 둬서 이기는 경우

*#세로*

self.m3.omok\_pan[3][0] = 1

self.assertEqual(self.m3.victory\_condition(), 0)

self.m3.omok\_pan[4][0] = 1

self.assertEqual(self.m3.victory\_condition(), 1)

검은 돌은 게임 규칙상 공정함을 위해 둘 수 없는 위치를 갖도록 했다.

검은 돌을 둘 때 검은 돌 3개 한 줄, 또 다른 한 줄이 만나서 ㄱ자 모양이나 ㅅ모양이 되는 자리에는

돌을 둘 수 없도록 설정했다.

**def** testcheck\_33(self):

*#검은 돌 두어져 있는 자리 초기화하고* *33 검사*

*#ㄱ 모양*

self.m5.omok\_pan[7][9] = 0

self.assertEqual(self.m5.check\_33(7, 9), **False**)

self.m5.omok\_pan[8][9] = 1

self.assertEqual(self.m5.check\_33(7, 9), **False**)

self.m5.omok\_pan[9][9] = 1

self.assertEqual(self.m5.check\_33(7, 9), **True**)

*#대각선 모양*

self.m1.omok\_pan[7][8] = 1

self.assertEqual(self.m1.check\_33(7, 9), **False**)

self.m1.omok\_pan[8][8] = 1

self.assertEqual(self.m1.check\_33(7, 9), **False**)

self.m1.omok\_pan[9][7] = 1

self.assertEqual(self.m1.check\_33(7, 9), **True**)

33 규칙과 유사하게 44규칙 또한 만들고 테스트했다.

**def** testcheck\_44(self):

*#ㄱ 모양*

self.assertEqual(self.m5.check\_44(7, 10), **False**)

self.m5.omok\_pan[8][10] = 1

self.assertEqual(self.m5.check\_44(7, 10), **False**)

self.m5.omok\_pan[9][10] = 1

self.assertEqual(self.m5.check\_44(7, 10), **False**)

self.m5.omok\_pan[10][10] = 1

self.assertEqual(self.m5.check\_44(7, 10), **True**)

*#대각선 모양*

self.m5.omok\_pan[10][9] = 1

self.assertEqual(self.m5.check\_44(11, 10), **False**)

self.m5.omok\_pan[9][8] = 1

self.assertEqual(self.m5.check\_44(11, 10), **False**)

self.m5.omok\_pan[8][7] = 1

self.assertEqual(self.m5.check\_44(11, 10), **True**)

검은 돌 한 줄이 모여서 5개가 되는 경우가 아니라 6개가 되는 경우라면

그 자리에 돌을 둘 수 없도록 설정했다.

**def** testcheck\_6(self):

*# 검은 돌은* *6목을 둘 수 없게 하는지 검사*

self.m2.omok\_pan[0][3] = 1

self.assertEqual(self.m2.check\_6(0, 4), **False**)

self.m2.omok\_pan[0][5] = 1

self.assertEqual(self.m2.check\_6(0, 4), **True**)

self.m2.omok\_pan[0][5] = 2

self.assertEqual(self.m2.check\_6(0, 4), **False**)

self.m2.omok\_pan[0][6] = 1

self.assertEqual(self.m2.check\_6(0, 4), **False**)

self.m2.omok\_pan[0][5] = 1

self.assertEqual(self.m2.check\_6(0, 4), **True**)

*#흰돌은* *6목 둘 수 있는지 검사*

self.m1.omok\_pan[1][5] = 2

self.assertEqual(self.m1.check\_6(1, 8), **False**)

self.m1.omok\_pan[1][6] = 2

self.assertEqual(self.m1.check\_6(1, 8), **False**)

self.m1.omok\_pan[1][7] = 2

self.assertEqual(self.m1.check\_6(1, 8), **False**)

self.m1.omok\_pan[1][9] = 2

self.assertEqual(self.m1.check\_6(1, 8), **False**)

self.m1.omok\_pan[1][10] = 2

self.assertEqual(self.m1.check\_6(1, 8), **False**)

self.m1.omok\_pan[1][8] = 2

*#대각선 검사*

self.m4.omok\_pan[0][0] = 0

self.assertEqual(self.m4.check\_6(3, 3), **False**)

self.m4.omok\_pan[4][4] = 1

self.assertEqual(self.m4.check\_6(3, 3), **False**)

self.m4.omok\_pan[5][5] = 1

self.assertEqual(self.m4.check\_6(3, 3), **False**)

self.m4.omok\_pan[6][6] = 1

self.assertEqual(self.m4.check\_6(3, 3), **True**)

*#세로 검사*

self.m6.omok\_pan[9][7] = 1

self.assertEqual(self.m6.check\_6(8, 7), **False**)

self.m6.omok\_pan[10][7] = 1

self.assertEqual(self.m6.check\_6(8, 7), **True**)

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

unittest.main()