Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

INFORME PRÀCTICA - 2048 GAME

Tipus de tests:

Caixa negra:
- **Particions equivalents, valors límit i frontera**: Es divideix el domini en particions, correctes y no correctes. En aquest test, mirem els valors fronteres, els valors limits i valors diferenciats dins dels limits. En el nostre cas, com tenim una matriu 4x4, els valors correctes son entre 0 i 3 tots dos inclòs. Per això fem aquests tests:
Per a que els tests no donin error, fem assertThrows en els tests que donen errors.
En aquestes captures es mostren els valors -10, -1, 0, 1, 3, 4, 5, 10.

```
assertEquals(b[0][0].getValue(), actual: 2);
assertEquals(b[0][0].getColor(),InfoGame.Color.GREEN);
```

```
assertEquals(b[3][3].getValue(), actual: 2);
assertEquals(b[3][3].getColor(),InfoGame.Color.GREEN);
```

```
board.moveBlock( i: 1, j: 1, c: 'a');
```

```
assertThrows(AssertionError.class, () -> board.moveBlock( i: -1, j: 0, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: 0, j: 0, c: 'x'));
b[0][0] = null;
assertThrows(AssertionError.class, () -> board.moveBlock( i: 0, j: 0, c: 's'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: -1, j: -1, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: 5, j: 5, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: 4, j: 4, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: 10, j: 0, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: -10, j: -10, c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: -10, j: -10, c: 'f'));
assertThrows(AssertionError.class, () -> board.moveBlock(SIZE,  j: 0,  c: 'w'));
assertThrows(AssertionError.class, () -> board.moveBlock( i: 0, SIZE,  c: 'a'));
```

- **Pairwise**: És una manera de fer testing amb combinacions de diferents paràmetres sense tenir la necessitat de fer totes les possibilitats assegurant que no hi hagi cap error. En el nostre cas, hem fet en el canMove utilitzant la pàgina web https://pairwise.teremokgames.com/:
0 0 w
0 1 a

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

0 2 s
0 3 d
1 1 s
1 2 d
1 3 w
1 0 a
2 2 w
2 3 a
2 0 s
2 1 d
3 3 s
3 0 d
3 1 w
3 2 a

En el nostre cas, com hem inicialitzat un taulell amb 2 valors, hi ha casos on s'intenta fer un canMove d'un bloc buit, és a dir, valor = 0, per tant les precondicions fa que surti error i evitem l'error del test amb assertThrows.

```
//pairwise
assertFalse(board.canMove( i: 0, j: 0, c: 'w'));
assertThrows(AssertionError.class, () -> board.canMove( i: 0,  j: 1,  c: 'a'));
assertThrows(AssertionError.class, () -> board.canMove( i: 0,  j: 2,  c: 's'));
assertThrows(AssertionError.class, () -> board.canMove( i: 0,  j: 3,  c: 'd'));
assertThrows(AssertionError.class, () -> board.canMove( i: 1,  j: 1,  c: 's'));
assertTrue(board.canMove( i: 1, j: 2, c: 'd'));
assertThrows(AssertionError.class, () -> board.canMove( i: 1, j: 3, c: 'w'));
assertThrows(AssertionError.class, () -> board.canMove( i: 1, j: 0, c: 'a'));
assertThrows(AssertionError.class, () -> board.canMove( i: 2, j: 2, c: 'w'));
assertThrows(AssertionError.class, () -> board.canMove( i: 2, j: 3, c: 'a'));
assertThrows(AssertionError.class, () -> board.canMove( i: 2, j: 0, c: 's'));
assertThrows(AssertionError.class, () -> board.canMove( i: 2, j: 1, c: 'd'));
assertFalse(board.canMove( i: 3, j: 3, c: 's'));
assertThrows(AssertionError.class, () -> board.canMove( i: 3, j: 0, c: 'd'));
assertThrows(AssertionError.class, () -> board.canMove( i: 3, j: 1, c: 'w'));
assertThrows(AssertionError.class, () -> board.canMove( i: 3, j: 2, c: 'a'));
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

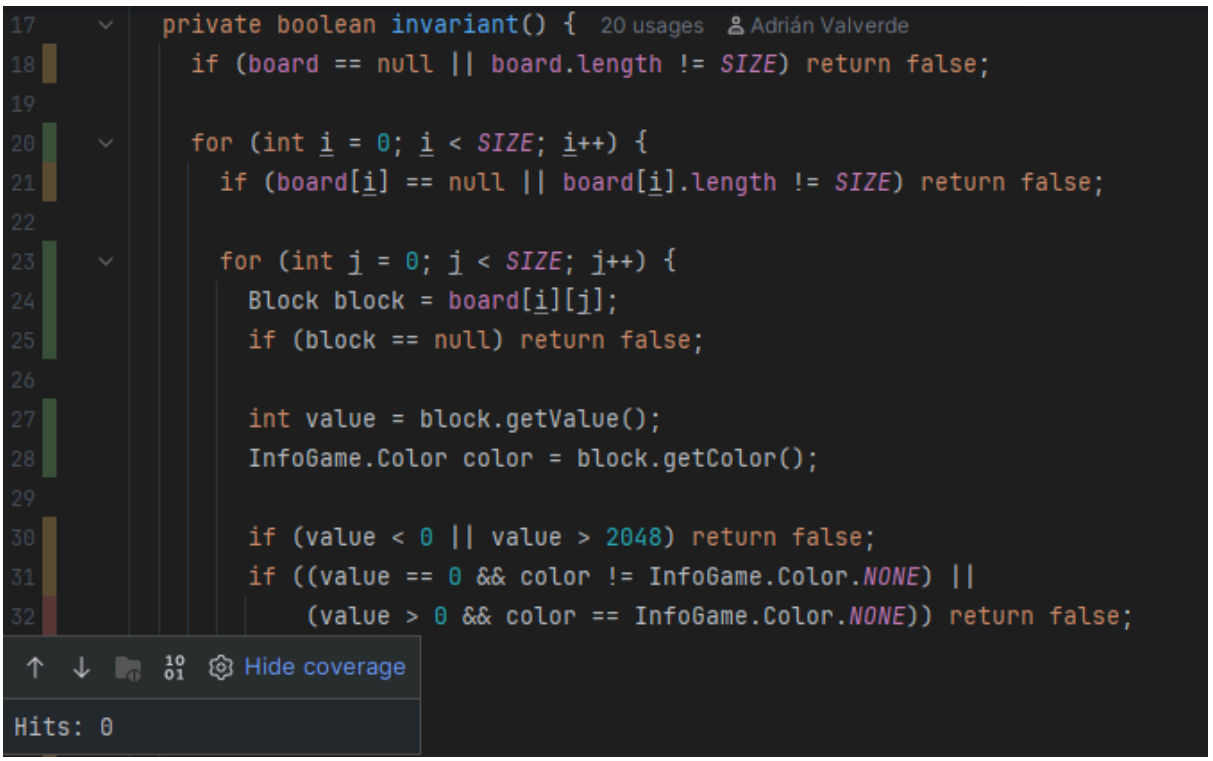
Caixa blanca:

- **Statement coverage**: Serveix per veure si els tests passen per totes les línies de codi en forma de percentatge.
En el nostre codi, tenim invariants que eviten que per exemple el bloc mai tingui un valor diferent als que tenim assignats, és a dir tampoc poden ser negatius. Això fa que precondicions de altres classes mai es puguin testejar i que no passi per aquestes línies de codi i que el coverage no surti 100%. Proves:
Class InfoGame Color:

```
Color(int value) {  24 usages  Adrián Valverde
  //precondition
  assert(value >= 0 && value <= 11);

  this.value = value;

  //postcondition
  assert (value == this.value && this.value >= 0 && this.value <= 11);
```

```
public class Block  86 usages  Adrián Valverde +1
{
  private int value;  15 usages
  private InfoGame.Color color;  11 usages

  private boolean invariant() {  6 usages  Adrián Valverde
    return ((value >= 0 && value <= 2048) && (color.getValue() == log2(value)));
  }
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952



En la linea 32, mai podrem crear un bloc amb value > 0 i color NONE.

Com aquest cas en tenim tots els que fan que el percentatge no sigui 100% de les classes. També hem pensat en no fer testing de les classes que només fan funcions de JavaUtil o altres. Un exemple seria la classe ScannerGame, que només conté un objecte tipus Scanner i les seves funcions només són JavaUtil per a poder fer el mock mitjançant IOC. Per tant, els asserts dels invariant, sempre son true, per tant sempre baixarà el percentatge.

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```java
package eng.uab.tqs.game2048.model;

import java.util.Scanner;

public class ScannerGame {  5 usages  Adrián Valverde Ambrosio
  Scanner scanner;  4 usages

  public ScannerGame() { this.scanner = new Scanner(System.in); }

  public String scanName() {  no usages  1 override  Adrián Valverde Ambrosio
    String input = scanner.nextLine().trim().toLowerCase();
    return input;
  }

  public Character scanOptions() {  1 usage  1 override  Adrián Valverde Ambrosio
    String input = scanner.nextLine().trim().toLowerCase();
    return input.charAt(0);
  }

  public String nextLine() { return scanner.nextLine(); }
}
```

Block	75,1 %	187	62	249
Block()	64,7 %	22	12	34
Block(int, Color)	84,9 %	45	8	53
getColor()	100,0 %	3	0	3
getValue()	100,0 %	3	0	3
invariant()	89,5 %	17	2	19
log2(int)	100,0 %	12	0	12
mix()	71,4 %	60	24	84
resetBlock()	60,0 %	24	16	40

Board.java	88,3 %	1.450	192	1.642
Board	88,3 %	1.450	192	1.642
Board()	83,7 %	123	24	147
Board(Board)	84,6 %	154	28	182
canMove(int, int, char)	94,4 %	134	8	142
drawBoard()	100,0 %	27	0	27
getBoard()	100,0 %	3	0	3
getGenerator()	100,0 %	3	0	3
getRestart()	100,0 %	3	0	3
getScore()	100,0 %	3	0	3
invariant()	83,9 %	104	20	124
isGameOver()	86,0 %	74	12	86
isGameWinned()	78,2 %	43	12	55
join(int, int, char)	92,7 %	255	20	275
moveBlock(int, int, char)	85,5 %	142	24	166
moveBoard(char)	96,0 %	291	12	303
random()	73,9 %	68	24	92
randomInicialize()	63,6 %	14	8	22
setGenerator(Generator)	100,0 %	4	0	4
setRestart(boolean)	100,0 %	4	0	4

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

Game.java	71,4 %	432	173	605
Game	71,4 %	432	173	605
Game()	100,0 %	24	0	24
getExit()	100,0 %	3	0	3
getGameMatch()	63,6 %	7	4	11
getScores()	58,6 %	17	12	29
invariant()	68,0 %	17	8	25
menu()	82,6 %	57	12	69
playGame()	60,0 %	24	16	40
playGameFX()	56,8 %	21	16	37
saveScore(String)	81,9 %	77	17	94
setManager(FileManager)	60,0 %	12	8	20
setScanner(ScannerGame)	60,0 %	12	8	20
showScores()	80,3 %	114	28	142
showScoresFX()	74,2 %	46	16	62

GameMatch.java	70,2 %	179	76	255
GameMatch	70,2 %	179	76	255
getBoard()	63,6 %	7	4	11
invariant()	66,7 %	8	4	12
setBoard(Board)	60,0 %	12	8	20
setScanner(ScannerMovement)	57,1 %	16	12	28
startGame()	74,6 %	88	30	118
startGameFX()	65,4 %	34	18	52

Generator.java	100,0 %	50	0	50
Generator	100,0 %	50	0	50
genRandom()	100,0 %	8	0	8
randomInicialize(Block[][])	100,0 %	39	0	39

InfoGame.java	93,8 %	165	11	176
InfoGame	0,0 %	0	3	3
Color	95,4 %	165	8	173
getValue()	100,0 %	3	0	3

Resum final:

Element	Class, %	Method, %	Line, %	Branch, %
eng.uab.tqs.game2048.model	68% (13/19)	82% (87/106)	92% (756/8...	77% (549/712)
mock	71% (5/7)	77% (31/40)	88% (287/3...	89% (162/181)
Block	100% (1/1)	100% (9/9)	100% (36/36)	62% (34/54)
Board	100% (1/1)	100% (19/19)	99% (263/2...	80% (256/318)
FileManager	0% (0/1)	0% (0/4)	0% (0/4)	100% (0/0)
Game	100% (1/1)	93% (14/15)	95% (97/102)	60% (60/99)
GameMatch	100% (1/1)	100% (6/6)	95% (42/44)	63% (29/46)
Generator	100% (1/1)	100% (2/2)	100% (9/9)	75% (3/4)
InfoGame	50% (1/2)	100% (4/4)	100% (18/18)	50% (5/10)
main	0% (0/1)	0% (0/1)	0% (0/2)	100% (0/0)
Movement	0% (0/1)	100% (0/0)	100% (0/0)	100% (0/0)
ScannerGame	100% (1/1)	25% (1/4)	28% (2/7)	100% (0/0)
ScannerMovement	100% (1/1)	50% (1/2)	50% (2/4)	100% (0/0)

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

- Decision coverage, condition coverage i path coverage:

Decision i condition coverage: es basa en el fet que el codi pren totes les decisions possibles, és a dir, totes les condicions passen per false i per true. En el nostre cas veiem codi on s'executa cada cas del switch, i a més a més, en la funció de canMove, trobem un if després del switch, això crea molts camins.

Path coverage: el path coverage en ajuda a que tots els camins que es poden executar en el codi, s'executin. Amb ajuda d'un diagrama de flux pots veure tots els paths que pot tenir el teu mètode .

De la classe board:

- public int[] moveBlock(int i, int j, char c)

```
148   public int[] moveBlock(int i, int j, char c) {  31 usages  1 override  Adrián Valverde Ambrosio
149     assert invariant();
150     //preconditions
151     assert(i >= 0);
152     assert(i < SIZE);
153     assert(j >= 0 && j < SIZE);
154     assert(board[i][j].getValue() > 0);
155     assert(c == 'w' || c == 'a' || c == 's' || c == 'd');
156
157     int i2 = i;
158     int j2 = j;
159     switch (c) {
160       case 'a':
161         j2 = j-1;
162         break;
163
164       case 'w':
165         i2 = i-1;
166         break;
167
168       case 'd':
169         j2 = j+1;
170         break;
171
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```
        case 's':
          i2 = i+1;
          break;

        default:
          break;
      }
      board[i2][j2] = new Block(board[i][j].getValue(), board[i][j].getColor());
      board[i][j].resetBlock();
      int[] coords = {i2,j2};
      assert invariant();
      assert(board[i][j].getValue() == 0);
      assert(board[i2][j2].getValue() != 0);
      return coords;
    }
```

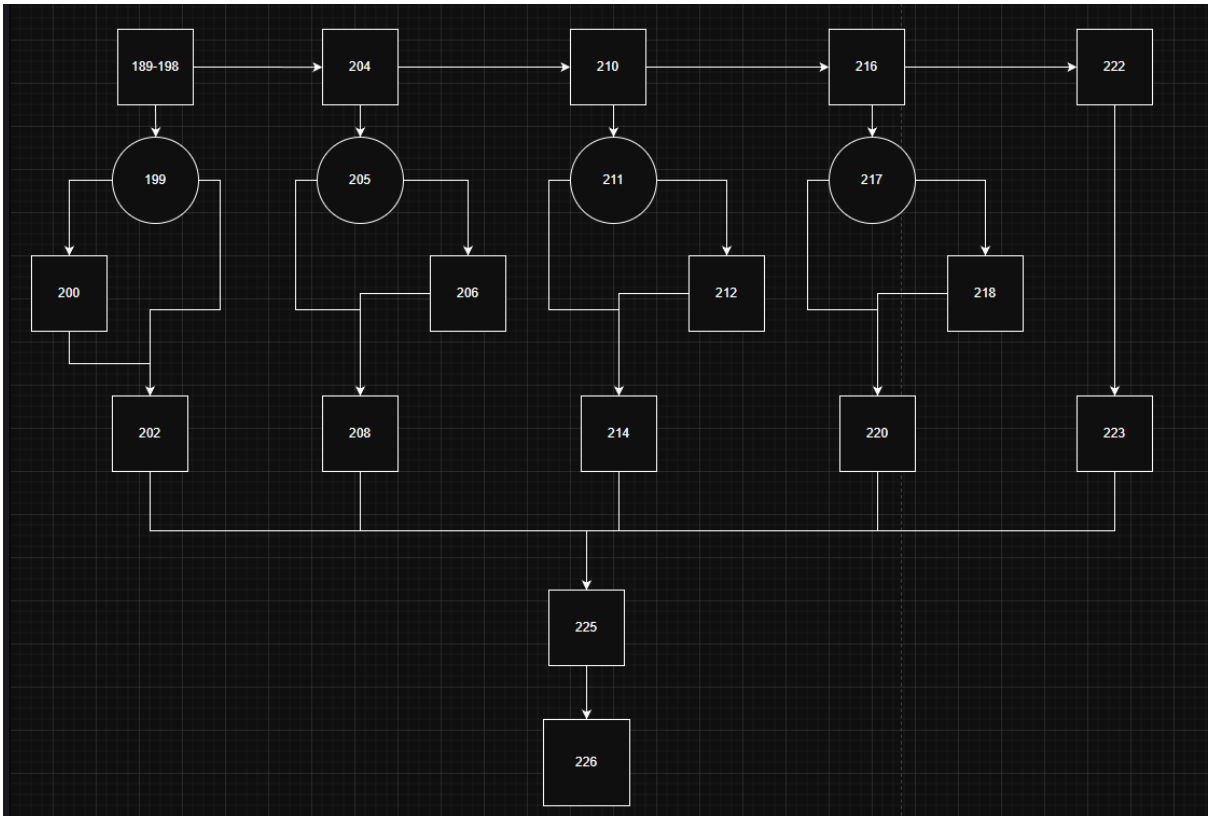
- public boolean canMove(int i, int j, char c)

```
  public boolean canMove(int i, int j, char c) {  41 usages  1 override  Adrián Valverde Ambrosio +1
    assert invariant();
    //preconditions
    assert(i >= 0 && i < SIZE);
    assert(j >= 0 && j < SIZE);
    assert(board[i][j].getValue() > 0);
    assert(c == 'w' || c == 'a' || c == 's' || c == 'd');

    boolean result = false;
    switch (c) {
      case 'a':
        if (j > 0) {
          result = (0 == board[i][j - 1].getValue());
        }
        break;

      case 'w':
        if (i > 0) {
          result = (0 == board[i - 1][j].getValue());
        }
        break;

      case 'd':
        if (j < SIZE - 1) {
          result = (0 == board[i][j + 1].getValue());
        }
        break;
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```
216         case 's':
217           if (i < SIZE - 1) {
218             result = (0 == board[i + 1][j].getValue());
219           }
220           break;
221
222         default:
223           break;
224       }
225       assert invariant();
226       return result;
227     }
```

- path coverage:



Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

- public boolean join(int i, int j, char c)

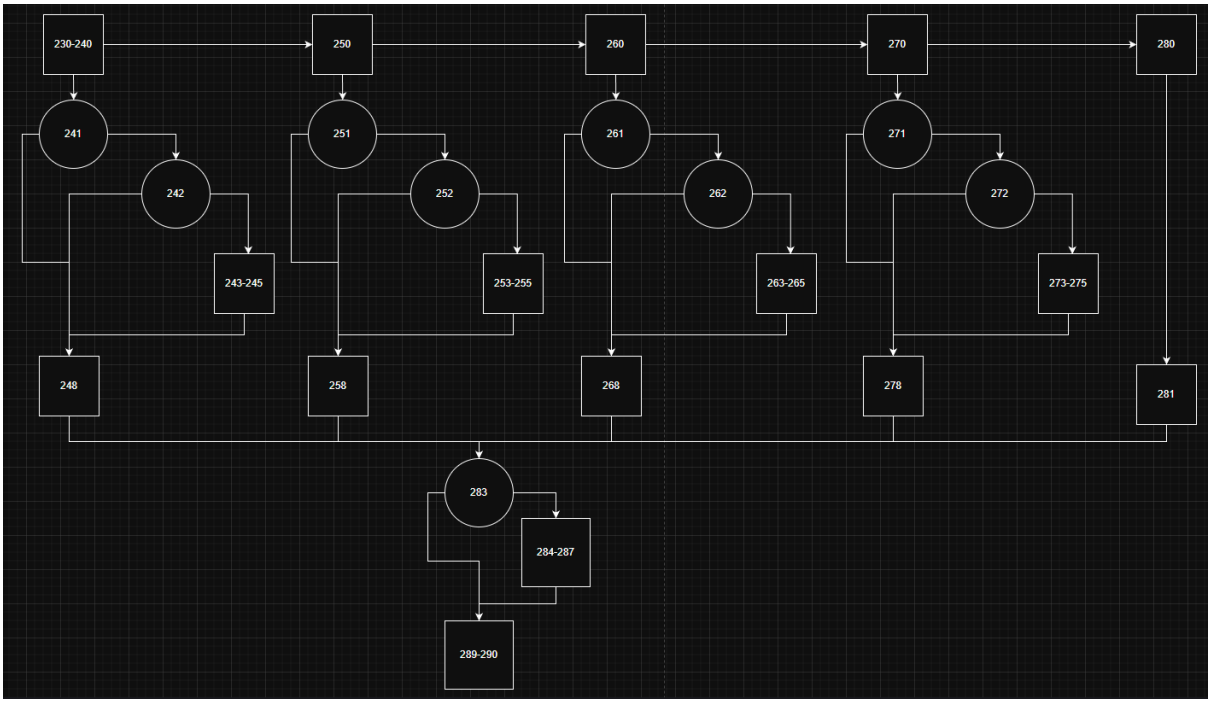
```java
public boolean join(int i, int j, char c) {  20 usages  1 override  Adrián Valverde Ambrosio +2
  assert invariant();
  //preconditions
  assert(i >= 0 && i < SIZE);
  assert(j >= 0 && j < SIZE);
  assert(board[i][j].getValue() > 0);
  assert(c == 'w' || c == 'a' || c == 's' || c == 'd');

  boolean result = false;
  int scoreAdd = 0;
  switch (c) {
    case 'w':
      if (i > 0) {
        if (board[i][j].getValue() == board[i-1][j].getValue()) {
          board[i-1][j].mix();
          result = true;
          scoreAdd += board[i-1][j].getValue();
        }
      }
      break;

    case 'a':
      if (j > 0) {
        if (board[i][j].getValue() == board[i][j - 1].getValue()) {
          board[i][j-1].mix();
          result = true;
          scoreAdd += board[i][j-1].getValue();
        }
      }
      break;
```

```java

    case 's':
      if (i < SIZE - 1) {
        if (board[i][j].getValue() == board[i+1][j].getValue()) {
          board[i+1][j].mix();
          result = true;
          scoreAdd += board[i+1][j].getValue();
        }
      }
      break;

    case 'd':
      if (j < SIZE - 1) {
        if (board[i][j].getValue() == board[i][j + 1].getValue()) {
          board[i][j+1].mix();
          result = true;
          scoreAdd += board[i][j+1].getValue();
        }
      }
      break;

    default:
      break;
  }
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```
283         if (result) {
284           board[i][j].resetBlock();
285           //postcondition
286           assert score < scoreAdd+score;
287           score += scoreAdd;
288         }
289         assert invariant();
290         return result;
291       }
```

- path coverage:



Loop testing:

- simple loop → showScores() de la classe Game

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```java
@Test  Adrián Valverde
void loopSimpleTest()
{
  //avoid loop
  fm.setConfig("EMPTY_FILE");
  assertDoesNotThrow(() -> game.showScores());
  List<String> s = game.getScores();
  assertNotNull(s);
  assertEquals( expected: 0, s.size());

  //one iteration
  fm.setConfig("ONE_SCORE");
  game.showScores();
  s = game.getScores();
  assertEquals( expected: 1, s.size());
  assertEquals( expected: "TestUser:400", s.get(0));

  //two iterations
  fm.setConfig("two");
  game.showScores();
  List<String> s2 = game.getScores();
  assertEquals( expected: 2, s2.size());

    //Some iterations
    fm.setConfig("some");
    game.showScores();
    List<String> s3 = game.getScores();
    assertEquals( expected: 4, s3.size());

    //ten iterations
    fm.setConfig("ten");
    game.showScores();
    List<String> s4 = game.getScores();
    assertEquals( expected: 10, s4.size());

  }
}
```

```java
for (String s : scores) {
  String[] parts = s.split( regex: ":");
  System.out.printf("%d\t%s\t\t%s\n", rank, parts[0].trim(), parts[1].trim());
  rank++;
}
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

- Loops aniuats →isGameOver() de la classe Board

```
414     @Test  Adrián Valverde
415     void loopNestedGameOver()
416     {
417       //interior simple loop
418       Board board = new Board();
419       assertFalse(board.isGameOver());
420       Block[][] b = board.getBoard();
421
422       //some iterations
423       for(int j = 0; j < SIZE; j++)
424       {b[0][j] = new Block( value: 2, InfoGame.Color.GREEN);}
425       assertFalse(board.isGameOver());
426
427       //half iterations
428       for (int i = 0; i < 2; i++) {
429         for (int j = 0; j < 4; j++) {
430           b[i][j] = new Block( value: 2, InfoGame.Color.GREEN);
431         }
432       }
433
```

```
434       assertFalse(board.isGameOver());
435
436
437       //full iterations
438       for (int i = 0; i < SIZE; i++ )
439       {
440         for (int j = 0; j < SIZE; j++ )
441         {b[0][j] = new Block( value: 2, InfoGame.Color.GREEN);}
442       }
443
444       boolean result = board.isGameOver();
445       //we just want to run through all the board
446       assertTrue( condition: result || !result);
447
448     }
449 }
```

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```java
public boolean isGameOver() {  15 usages  1 override  Adrián Valverde
    assert invariant();
    boolean movementDone = false;
    int i = 0;
    for (i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j].getValue() == 0) {
                assert invariant();
                return false;
            }
        }
    }
```

Data driven test:

La tècnica de Data-Driven Testing consisteix a separar les dades de prova de la lògica del test, de manera que un mateix cas de prova s'executa múltiples vegades utilitzant diferents conjunts de dades externes. En el nostre cas, hem creat 2 arxius CSV on hem deixat les dades a llegir i dins dels mètodes de test, només cal llegir l'arxiu

Hem provat aquesta tècnica pels mètodes de moveBoard() i showScored()

```java


@Test  new *
void moveBoard_dataDriven() throws Exception {
    List<String> lines = Files.readAllLines(
        Path.of( first: "src/test/java/eng/uab/tqs/game2048/model/datadriven/board.csv"));

    for (String line : lines) {
        char dir = line.trim().charAt(0);

        Board board = new Board();
        board.setGenerator(gen);
        board.randomInicialize();
        assertDoesNotThrow(() -> board.moveBoard(dir));
    }
}

}
```

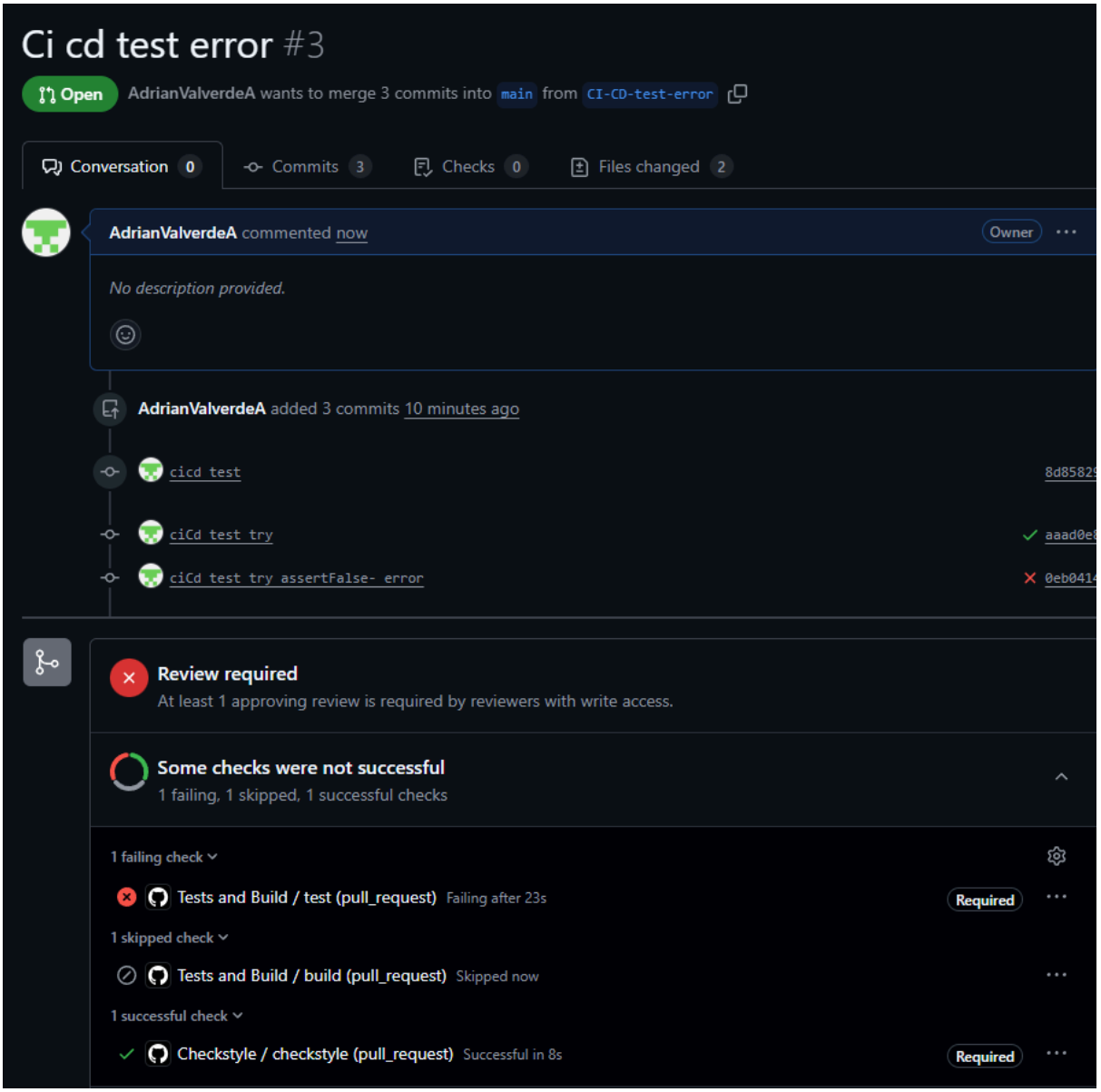
Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

```java
@Test  new *
void saveScore_dataDriven() throws Exception {

  List<String> names = Files.readAllLines(
      Path.of( first: "src/test/java/eng/uab/tqs/game2048/model/datadriven/score.csv"));

  for (String name : names) {

    Game game = new Game();
    FileManagerMock fm = new FileManagerMock();
    game.setManager(fm);

    fm.setConfig("EMPTY_FILE");

    BoardMock board = new BoardMock();
    board.setScore(100);

    game.getGameMatch().setBoard(board);

    game.saveScore(name);

    assertNotNull(game.getScores());
  }
}
}
```

CI test:

S'ha configurat un flux de CI mitjançant GitHub Actions que executa els tests automàticament a cada pull request. El merge es bloqueja si els tests no passen, o si no segueixen l'estructura que té el checkstyle. A la imatge, podem veure un merge que ha fallat, ja que no complia cap de la premisses establerta de no existir errors de tests.

Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952



Izan Caballer Jimenez - 1710282
Adrián Valverde Ambrosio - 1707952

Falta aprovació de persona amb permisos, pero pasa correctament: