

**Arnau Muñoz Barrera, NIU:** 1665982

**José Ortín López, NIU:** 1667573

**Grup:** Divendres 10:30

**Projecte:** Battleship

### **BattleShipController:**

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsMockito( )
- **Tipus de test:** Mock Objects
- **Tècniques utilitzades:**
  - Simulem el comportament de BoardModel, de BoardView, de InputView i de MessageView, verificant que els mètodes associats (getSize(), isEmpty(), getShipChar()) són cridats correctament i mostren els missatges adequats segons l'estat al que hem portat els mock objects.

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsStatement( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement coverage:** Assegurem que cada línia de codi dins del mètode sigui executada almenys una vegada
    - **Caselles vàlides:** Es prova que un vaixell es pot col·locar correctament a una posició vàlida del tauler
    - **Coordenades no vàlides:** Es comprova que, quan s'intenten col·locar vaixells fora dels límits del tauler, es mostra el missatge corresponent i es torna a intentar
    - **Casella ja feta servir:** Es verifica que, si s'intenta col·locar una nau en una casella ja ocupada, es mostri el missatge d'error i no es sobreescriu la casella.

Coverage	vered Instructions	lissed Instructions	Total Instructions
10,9 %	480	3.915	4.395

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips2(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsStatement2( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement coverage:** Assegurem que cada línia de codi dins del mètode sigui executada almenys una vegada
    - **Caselles vàlides:** Es prova que un vaixell es pot col·locar correctament a una posició vàlida del tauler
    - **Coordenades no vàlides:** Es comprova que, quan s'intenten col·locar vaixells fora dels límits del tauler, es mostra el missatge corresponent i es torna a intentar
    - **Casella ja feta servir:** Es verifica que, si s'intenta col·locar una nau en una casella ja ocupada, es mostri el missatge d'error i no es sobreescriu la casella.

Coverage	vered Instructions	lissed Instructions	Total Instructions
11,5 %	507	3.888	4.395

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsLoopSimple( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Loop Simple:**
    - **Col·locació d'un vaixell correctament**
      - Es comprova que un vaixell es col·loca en una casella vàlida del tauler sense cap error. El codi entra al bucle, realitza validacions i col·loca correctament el vaixell a (1,1). Això cobreix el camí simple del bucle sense errors
    - **Col·locació fora dels límits del tauler:**
      - Es verifica que, en introduir coordenades fora dels límits (primer intent a (5,5)), el sistema detecta l'error, mostra el

missatge de “Coordinates out of bounds”, i torna a sol·licitar noves coordenades fins que es col·loca correctament a (1,1). Això comprova el comportament quan hi ha una validació fallida

- **Col·locació en una casella ocupada:**
  - Es comprova que, en intentar col·locar un vaixell en una casella ja ocupada (primer intent a (1,1)), el sistema detecta l'error, mostra el missatge “Cell already occupied”, i permet col·locar la nau en una altra casella (2,2). Això cobreix un altre camí del bucle on es detecten caselles ocupades.

```

302 void testPlaceShipsLoopSimple2() {
303
304     // 1. Placement d'un ship correctament
305
306     BoardModel board1 = new BoardModel(4, '-', 'S', 'X', 'O');
307
308     BoardView boardView1 = new BoardView();
309     String name1 = "prova";
310     MockInputView inputView1 = new MockInputView(new int[] {1,1}); // Coloca el barco en (1,1)
311     MockMessageView messageView1 = new MockMessageView();
312     BattleshipController game1 = new BattleshipController(board1, boardView1, inputView1, messageView1, name1);
313
314     game1.placeShips(1);
315     assertEquals('S', board1.getCell(0, 0));
316
317     // 2. Placement d'un ship fora dels límits
318
319     BoardModel board2 = new BoardModel(4, '-', 'S', 'X', 'O');
320     MockInputView inputView2 = new MockInputView(new int[] {5,5,1,1}); // Primer intent a (5,5), placement final a (1,1)
321     MockMessageView messageView2 = new MockMessageView();
322     BattleshipController game2 = new BattleshipController(board2, boardView1, inputView2, messageView2, name1);
323
324     game2.placeShips(1);
325     assertTrue(messageView2.containsMessage("Coordinates out of bounds. Try again.")); // Missatge d'error per (5,5)
326     assertEquals('S', board2.getCell(0, 0));
327
328     // 3. Placement d'un ship a una casella ja feta servir
329
330     BoardModel board3 = new BoardModel(4, '-', 'S', 'X', 'O');
331     board3.setCell(0, 0, board3.getSize(), 'S');
332     MockInputView inputView3 = new MockInputView(new int[] {1,1,2,2}); // Primer intent a (1,1), placement final a (2,2)
333     MockMessageView messageView3 = new MockMessageView();
334     BattleshipController game3 = new BattleshipController(board3, boardView1, inputView3, messageView3, name1);
335
336     game3.placeShips(1);
337     assertTrue(messageView3.containsMessage("Cell already occupied. Try again.")); // Missatge d'error per casella ocupada
338     assertEquals('S', board3.getCell(0, 0));
339     assertEquals('S', board3.getCell(1, 1));
340 }
341
342
343
344

```

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

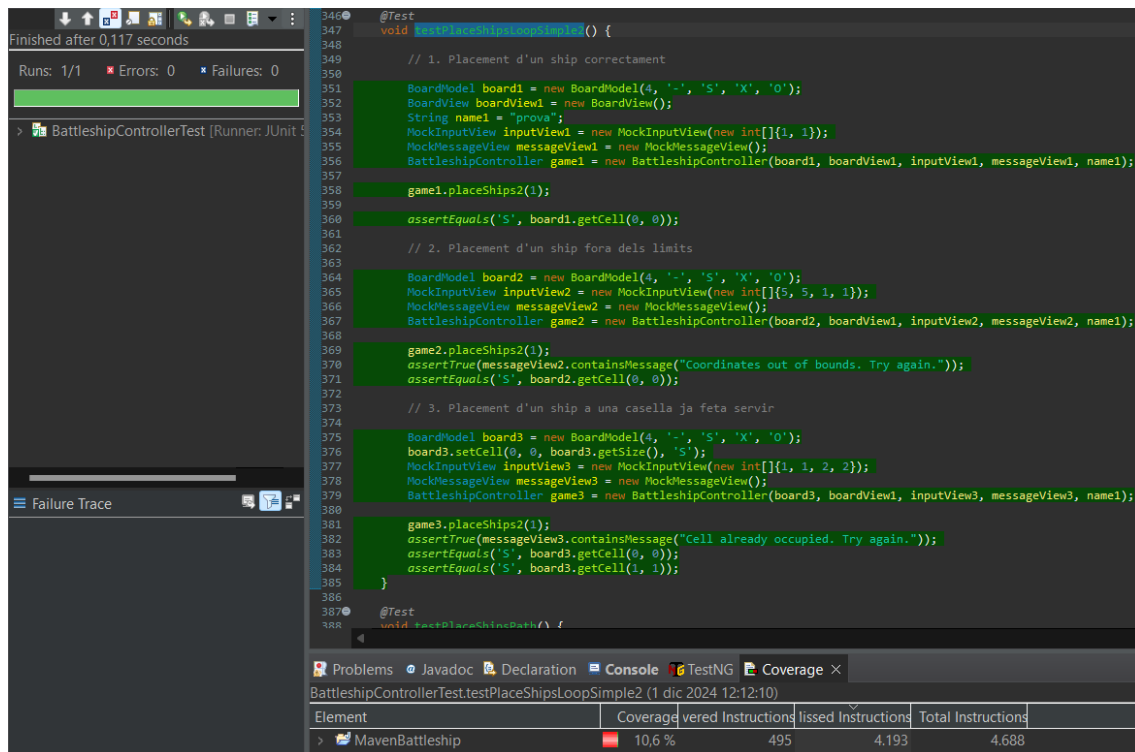
- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips2(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsLoopSimple2( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - Loop Simple:
    - **Col·locació d'un vaixell correctament**
      - Es comprova que un vaixell es col·loca en una casella vàlida del tauler sense cap error. El codi entra al bucle, realitza validacions i col·loca correctament el vaixell a (1,1). Això cobreix el camí simple del bucle sense errors
    - **Col·locació fora dels límits del tauler:**
      - Es verifica que, en introduir coordenades fora dels límits (primer intent a (5,5)), el sistema detecta l'error, mostra el missatge de “Coordinates out of bounds”, i torna a sol·licitar noves coordenades fins que es col·loca correctament a (1,1).

Això comprova el comportament quan hi ha una validació fallida

- **Col·locació en una casella ocupada:**
  - Es comprova que, en intentar col·locar un vaixell en una casella ja ocupada (primer intent a (1,1)), el sistema detecta l'error, mostra el missatge "Cell already occupied", i permet col·locar la nau en una altre casella (2,2). Això cobreix un altre camí del bucle on es detecten caselles ocupades.



```
346 @Test
347 void testPlaceShipLoopSimple() {
348
349     // 1. Placement d'un ship correctament
350
351     BoardModel board1 = new BoardModel(4, '-', 'S', 'X', 'O');
352     BoardView boardView1 = new BoardView();
353     String name1 = "prova";
354     MockInputView inputView1 = new MockInputView(new int[]{1, 1});
355     MockMessageView messageView1 = new MockMessageView();
356     BattleshipController game1 = new BattleshipController(board1, boardView1, inputView1, messageView1, name1);
357
358     game1.placeShips2(1);
359     assertEquals("S", board1.getCell(0, 0));
360
361     // 2. Placement d'un ship fora dels limits
362
363     BoardModel board2 = new BoardModel(4, '-', 'S', 'X', 'O');
364     MockInputView inputView2 = new MockInputView(new int[]{5, 5, 1, 1});
365     MockMessageView messageView2 = new MockMessageView();
366     BattleshipController game2 = new BattleshipController(board2, boardView1, inputView2, messageView2, name1);
367
368     game2.placeShips2(1);
369     assertTrue(messageView2.containsMessage("Coordinates out of bounds. Try again."));
370     assertEquals("S", board2.getCell(0, 0));
371
372     // 3. Placement d'un ship a una casella ja feta servir
373
374     BoardModel board3 = new BoardModel(4, '-', 'S', 'X', 'O');
375     board3.setCell(0, 0, board3.getSize(), 'S');
376     MockInputView inputView3 = new MockInputView(new int[]{1, 1, 2, 2});
377     MockMessageView messageView3 = new MockMessageView();
378     BattleshipController game3 = new BattleshipController(board3, boardView1, inputView3, messageView3, name1);
379
380     game3.placeShips2(1);
381     assertTrue(messageView3.containsMessage("Cell already occupied. Try again."));
382     assertEquals("S", board3.getCell(0, 0));
383     assertEquals("S", board3.getCell(1, 1));
384 }
385
386
387 @Test
388 void testPlaceShipLoop() {
389
390 }
```

Element	Coverage	Verified Instructions	Missed Instructions	Total Instructions
MavenBattleship	10,6 %	495	4.193	4.688

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** placeShips(int shipCount)

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testPlaceShipsPath( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Path coverage:** Assegurem que es cobreixen diferents camins (paths) dins del codi, utilitzant un conjunt específic d'entrades per passar les diverses condicions. Es centra en validar el flux complet a través dels possibles escenaris de col·locació dels vaixells:
    - **Coordenades fora dels límits**
      - El primer intent d'introduir coordenades (5,5) és invalidat perquè està fora dels límits del tauler. Això activa el missatge "Coordinates out of bounds. Try again", demostrant que el mètode detecta i gestiona aquest error correctament
    - **Casella Ocupada**
      - El segon intent (1,1) falla perquè la casella ja està ocupada. Això activa el missatge "Cell already occupied. Try again" i el flux continua fins que es troben coordenades vàlides
    - **Coordenades vàlides**
      - El tercer intent (2,2) és acceptat com a vàlid, i la nau es col·loca correctament en aquesta posició. Això valida que el bucle es comporta de manera adequada quan es donen coordenades vàlides després d'errors anteriors.

The screenshot shows an IDE with a Java test method and its coverage report. The test method is `testPlaceShipsPath` in `BattleshipControllerTest`. It uses `MockInputView` and `MockMessageView` to simulate user input and messages. The test method calls `game.placeShips(1)` and asserts that the message view contains the expected messages and that the ship is placed correctly.

```
387 @Test
388 void testPlaceShipsPath() {
389
390     MockInputView inputView = new MockInputView(new int[] {
391         5, 5, // Path 1: Coords fora dels límits
392         1, 1, // Path 2: Casella ocupada
393         2, 2 // Path 3: Coords vàlides
394     });
395
396     MockMessageView messageView = new MockMessageView();
397     BoardModel board = new BoardModel(4, '-', 'S', 'X', 'O');
398
399     board.setCell(0, 0, board.getSize(), 'S'); // Fem set de la casella (1,1) per pasar pel path de casella ocupada
400
401     BattleshipController game = new BattleshipController(board, new BoardView(), inputView, messageView, "prova");
402     game.placeShips(1);
403
404     assertTrue(messageView.containsMessage("Coordinates out of bounds. Try again."));
405     assertTrue(messageView.containsMessage("Cell already occupied. Try again."));
406     assertEquals('S', board.getCell(1, 1));
407 }
408
409
```

The coverage report at the bottom shows the following data:

Element	Coverage	Verified Instructions	Missed Instructions	Total Instructions
MavenBattleship	8.1 %	380	4.308	4.688

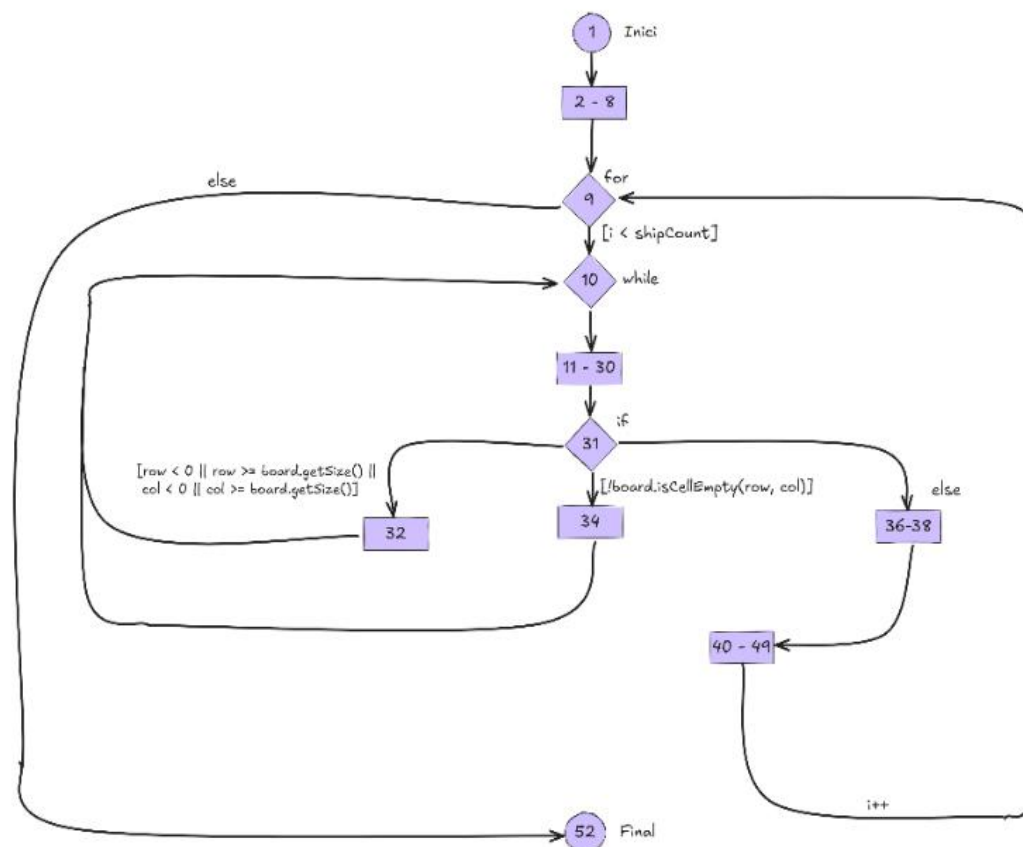
```

1 public void placeShips(int shipCount) {
2
3     //assert invariant(); // invariant check
4
5     messageView.showMessage(playerName + ", place your ships on the board.");
6     messageView.showMessage("Ships available: " + shipCount);
7     //assert (shipCount > 0); "ShipCount value must be greater than 0"; // precondition
8
9     for (int i = 0; i < shipCount; i++) {
10         while (true) {
11
12             int rowPre = inputView.getIntInput(playerName + ", enter row (1-" + board.getSize() + "): ");
13             int colPre = inputView.getIntInput(playerName + ", enter column (1-" + board.getSize() + "): ");
14
15             //assert (rowPre >= 0); "Row value must be greater or equal than 0"; // precondition
16             //assert (rowPre < board.getSize()); "Row value must be lower than board size"; // precondition
17             //assert (colPre >= 0); "Row value must be greater or equal than 0"; // precondition
18             //assert (colPre < board.getSize()); "Row value must be lower than board size"; // precondition
19
20             int row = rowPre - 1;
21             //assert row == rowPre - 1; // postcondition
22
23             int col = colPre - 1;
24             //assert col == colPre - 1; // postcondition
25
26             System.out.println();
27
28             //assert (board.getSize() > 1); "Board size value must be greater than 1"; // precondition
29
30             if (row < 0 || row >= board.getSize() || col < 0 || col >= board.getSize()) {
31                 messageView.showMessage("Coordinates out of bounds. Try again.");
32             } else if (!board.isCellEmpty(row, col)) {
33                 messageView.showMessage("Cell already occupied. Try again.");
34             } else {
35                 board.setCell(row, col, board.getSize(), board.getShipChar());
36                 //assert (board.getCell(row, col) != ' '); "Board cell should not be empty"; // postcondition
37                 break;
38             }
39
40             /*if (!board.isCellEmpty(row, col)) {
41                 messageView.showMessage("Cell already occupied. Try again.");
42             } else if (!board.setCell(row, col, board.getSize(), board.getShipChar())) {
43                 messageView.showMessage("Coordinates out of bounds. Try again.");
44             } else {
45                 break;
46             }
47             */
48         }
49     }
50 }
51
52 }

```



Path Coverage PlaceShips




**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** takeTurn()

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testTakeTurnStatement( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement coverage:** Es prova que totes les línies del codi s'executin, provant diferents casos que poden sorgir durant el joc.
    - **Jugada amb èxit (Hit)**
      - Aquest cas es verifica el flux quan el jugador encerta una casella amb un vaixell. Es col·loca un vaixell a (1,1) ('S') i l'usuari dispara correctament a aquesta casella. La funció retorna 1 per un encert, la casella es marca com a 'X' i el missatge "HIT!" apareix per pantalla
    - **Fallada (Miss)**
      - Aquest cas cobreix el flux per tirs fallits en caselles buides. El jugador dispara a una casella buida (2,2) ('-'). La funció retorna un 0 per casella buida, la casella es marca com a 'O' i apareix un missatge "MISS!" per pantalla
    - **Casella ja feta servir (vaixell descobert)**
      - Aquest cas verifica que el sistema reconeix i gestiona caselles ja encertades. Es dispara, de nou, a la casella (1,1) ja encertada prèviament ('X'). Seguidament. es dispara a la casella (2,2) buida ('-'). La funció treu per pantalla el següent missatge: "Already targeted. Try again.", retorna un 0 perquè el segon tir es a una casella buida (2,2) i apareix un missatge "MISS!" per pantalla.
    - **Casella ja feta servir (fallada prèvia)**
      - Aquest cas verifica que el sistema reconeix i gestiona caselles ja encertades. Es dispara, de nou, a la casella (1,1) ja encertada prèviament ('O'). Seguidament. es dispara a la casella (2,2) buida ('-'). La funció treu per pantalla el següent missatge: "Already targeted. Try again.", retorna un 0 perquè el segon tir es a una casella buida (2,2) i apareix un missatge "MISS!" per pantalla.

Coverage	covered Instructions	misses Instructions	Total Instructions
 15,6 %	687	3.708	4.395

**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.


**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController

- **Mètode:** takeTurn()

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testTakeTurnStatement2 ( )
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement coverage:** Es prova que totes les línies del codi s'executin, provant diferents casos que poden sorgir durant el joc.
    - **Jugada amb èxit (Hit)**
      - Aquest cas es verifica el flux quan el jugador encerta una casella amb un vaixell. Es col·loca un vaixell a (1,1) ('S') i l'usuari dispara correctament a aquesta casella. La funció retorna 1 per un encert, la casella es marca com a 'X' i el missatge "HIT!" apareix per pantalla
    - **Fallada (Miss)**
      - Aquest cas cobreix el flux per tirs fallits en caselles buides. El jugador dispara a una casella buida (2,2) ('-'). La funció retorna un 0 per casella buida, la casella es marca com a 'O' i apareix un missatge "MISS!" per pantalla
    - **Casella ja feta servir (vaixell descobert)**
      - Aquest cas verifica que el sistema reconeix i gestiona caselles ja encertades. Es dispara, de nou, a la casella (1,1) ja encertada prèviament ('X'). Seguidament. es dispara a la casella (2,2) buida ('-'). La funció treu per pantalla el següent missatge: "Already targeted. Try again.", retorna un 0 perquè el segon tir es a una casella buida (2,2) i apareix un missatge "MISS!" per pantalla.
    - **Casella ja feta servir (fallada prèvia)**
      - Aquest cas verifica que el sistema reconeix i gestiona caselles ja encertades. Es dispara, de nou, a la casella (1,1) ja encertada prèviament ('O'). Seguidament. es dispara a la casella (2,2) buida ('-'). La funció treu per pantalla el següent missatge: "Already targeted. Try again.", retorna un 0 perquè el segon tir es a una casella buida (2,2) i apareix un missatge "MISS!" per pantalla.

Coverage	Verified Instructions	Missed Instructions	Total Instructions
 14,4 %	674	4.014	4.688



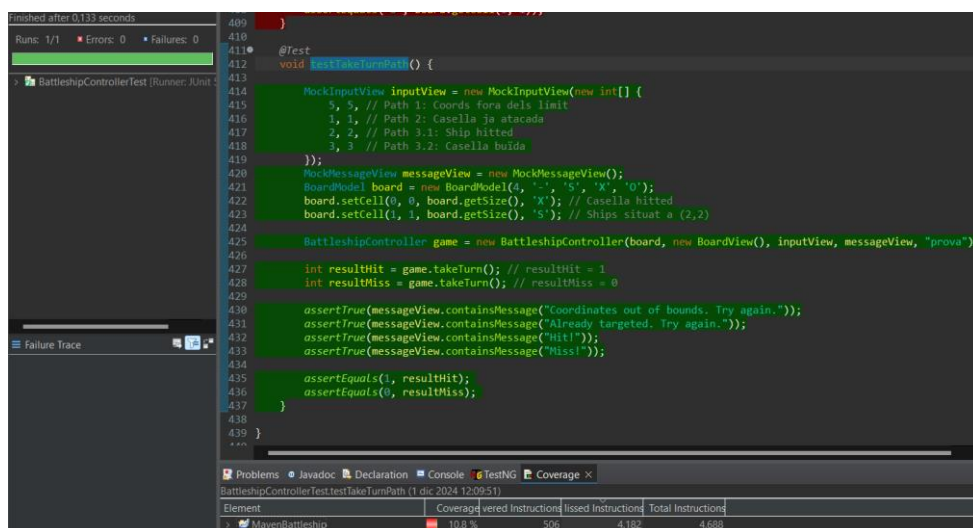
**Funcionalitat:** Permet al jugador col·locar una quantitat definida de vaixells al tauler.

**Localització:**

- **Arxiu:** BattleshipController.java
- **Classe:** BattleshipController
- **Mètode:** takeTurn()

**Test:**

- **Arxiu:** BattleshipControllerTest.java
- **Classe:** BattleshipControllerTest
- **Mètode:** testTakeTurnPath()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Path Coverage:** Assegurant que es recorren totes les possibles seqüències de decisions i camins dins del codi. A través d'una combinació d'entrades, es cobreixen els diferents camins que pot prendre el mètode
    - **Coordenades fora dels límits**
      - Verifica el camí en què les coordenades estan fora dels límits i el sistema detecta l'error. L'entrada (5,5) (fora dels límits del tauler 4x4). No es modificarà cap casella del tauler i es mostrarà per pantalla el missatge "Coordinates out of bounds. Try again"
    - **Casella ja atacada**
      - Cobreix el camí en què el jugador intenta atacar una casella ja utilitzada. L'entrada (1,1) (casella ja ataca amb estat 'X'). No es modifica cap casella del tauler i es mostra per pantalla el missatge "Already targeted. Try again."
    - **Casella amb vaixell (Hit):** Verifica que el sistema gestiona correctament els encerts. L'entrada (2,2) (casella amb un vaixell 'S'). El resultat es 1 (hit), la casella es marca com a 'X' i es mostra per pantalla el missatge "HIT!"
    - **Casella Buida (Miss):** Cobreix el camí en què el jugador ataca una casella buida. L'entrada (3,3) (casella buida amb estat '-'). El resultat és 0 (miss), la casella es marca com a 'O' i es mostra per pantalla el missatge "Miss!"



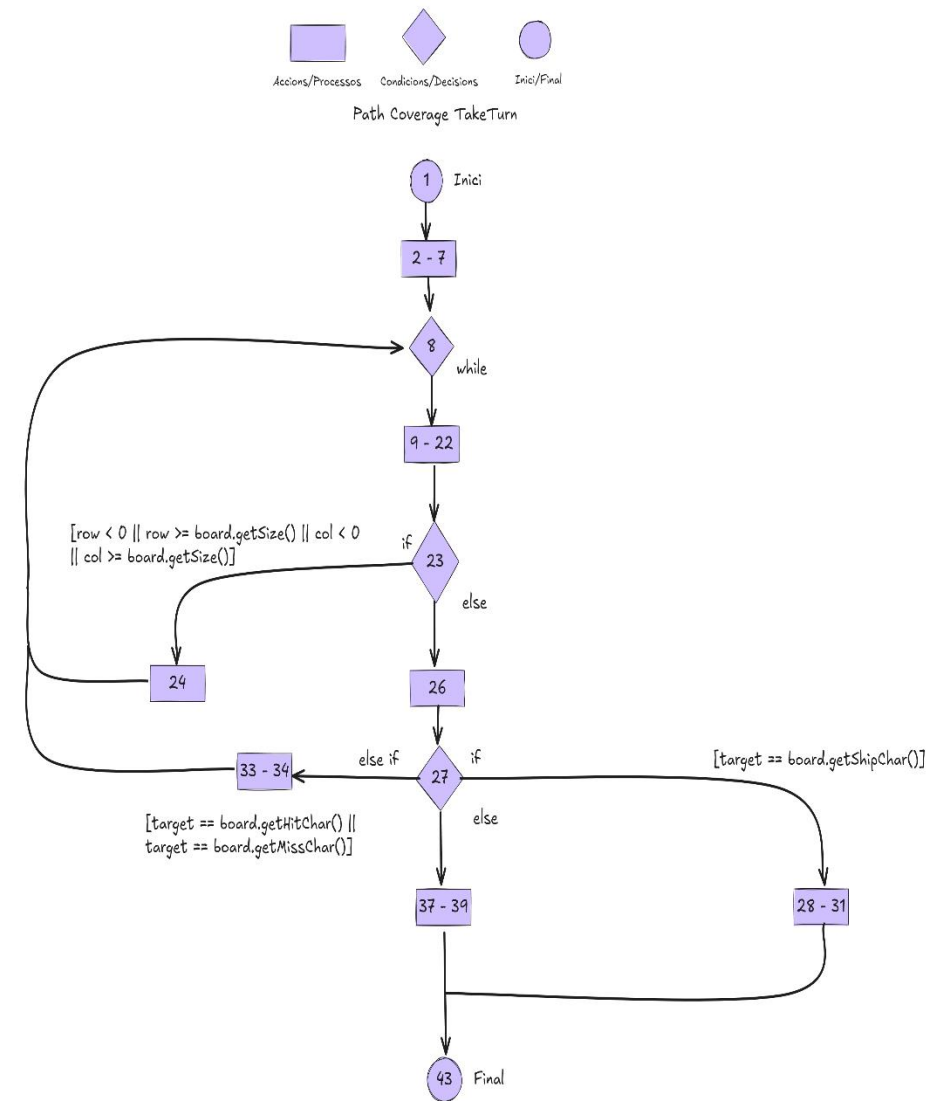
```
409 }
410
411 @Test
412 void testTakeTurnPath() {
413
414     MockInputView inputView = new MockInputView(new int[] {
415         5, 5, // Path 1: Coords fora dels límits
416         1, 1, // Path 2: Casella ja atacada
417         2, 2, // Path 3.1: Ship hit
418         3, 3 // Path 3.2: Casella buida
419     });
420     MockMessageView messageView = new MockMessageView();
421     BoardModel board = new BoardModel(4, '-', 'S', 'X', 'O');
422     board.setCell(0, 0, board.getSize(), 'X'); // Casella hit
423     board.setCell(1, 1, board.getSize(), 'S'); // Ships situat a (2,2)
424
425     BattleshipController game = new BattleshipController(board, new BoardView(), inputView, messageView, "prova");
426
427     int resultHit = game.takeTurn(); // resultHit = 1
428     int resultMiss = game.takeTurn(); // resultMiss = 0
429
430     assertTrue(messageView.containsMessage("Coordinates out of bounds. Try again."));
431     assertTrue(messageView.containsMessage("Already targeted. Try again."));
432     assertTrue(messageView.containsMessage("HIT!"));
433     assertTrue(messageView.containsMessage("Miss!"));
434
435     assertEquals(1, resultHit);
436     assertEquals(0, resultMiss);
437 }
438
439 }
```

Element	Coverage	Verified Instructions	Missed Instructions	Total Instructions
MavenBattleship	100.0 %	506	4.182	4.688

```

1 public int takeTurn() {
2     messageView.showMessage(playerName + ", it's your turn.");
3
4     //assert invariant(); // invariant check
5
6     boardView.printBoard(board, true);
7
8     while (true) {
9         int rowPre = inputView.getIntInput("Enter row: ");
10        int colPre = inputView.getIntInput("Enter column: ");
11
12        //assert (rowPre >= 0): "Row value must be greater or equal than 0"; // precondition
13        //assert (rowPre < board.getSize()): "Row value must be lower than board size"; // precondition
14        //assert (colPre >= 0): "Row value must be greater or equal than 0"; // precondition
15        //assert (colPre < board.getSize()): "Row value must be lower than board size"; // precondition
16
17        int row = rowPre - 1;
18        //assert row == rowPre - 1; // postcondition
19
20        int col = colPre - 1;
21        //assert col == colPre - 1; // postcondition
22
23        if (row < 0 || row >= board.getSize() || col < 0 || col >= board.getSize()) {
24            messageView.showMessage("Coordinates out of bounds. Try again.");
25        } else {
26            char target = board.getCell(row, col);
27            if (target == board.getShipChar()) {
28                board.setCell(row, col, board.getShipChar(), board.getHitChar());
29                messageView.showMessage("Hit!");
30                //assert (board.getCell(row, col) == 'X'): "Board cell should not be hit"; // postcondition
31                return 1; // Hit success
32            } else if (target == board.getHitChar() || target == board.getMissChar()) {
33                messageView.showMessage("Already targeted. Try again.");
34                //assert (board.getCell(row, col) == 'X' || board.getCell(row, col) == '0'): "Board cell must be already used"; // postcondition
35            } else {
36                board.setCell(row, col, board.getShipChar(), board.getMissChar());
37                messageView.showMessage("Miss!");
38                //assert (board.getCell(row, col) == '0'): "Board cell should be missed"; // postcondition
39                return 0; // Miss
40            }
41        }
42    }
43 }

```



## **BoardModel:**

**Funcionalitat:** Configurar el valor d'una celda del tauler sempre que la fila, columna i caràcter compleixin condicions de validació (rang, mida i tipus de caràcter)

**Localització:** <Arxiu, classe i mètode desenvolupat>

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean setCell(int row, int col, int size, char value)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testSetCellPairwise()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Pairwise testing:** les dimensions rellevants (fila, columna, mida i caràcter) poden prendre dos valors principals: **vàlid** i **invàlid**. Aquesta tècnica combina totes les possibles interaccions entre parelles de dimensions (per exemple, fila vàlida amb columna invàlida) per assegurar que es cobreixin les combinacions més representatives amb un nombre mínim de proves.

Casos	row	col	size	value
1	vàlid	vàlid	vàlid	vàlid
2	vàlid	vàlid	vàlid	invàlid
3	vàlid	vàlid	invàlid	vàlid
4	vàlid	invàlid	vàlid	vàlid
5	vàlid	invàlid	vàlid	invàlid
6	invàlid	vàlid	vàlid	vàlid
7	invàlid	vàlid	vàlid	invàlid
8	invàlid	invàlid	invàlid	vàlid
9	invàlid	invàlid	invàlid	invàlid

Quan fem servir una size vàlida, li passem board.getSize(), que retorna la mida del tauler definida a la instància board creada. En canvi, per provar una size invàlida, li passem un valor negatiu (-1), ja que no es pot generar una matriu amb una mida negativa, atès que el nostre tauler és una estructura de dades bidimensional.

**Funcionalitat:** Configurar el valor d'una celda del tauler sempre que la fila, columna i caràcter compleixin condicions de validació (rang, mida i tipus de caràcter)

**Localització:** **Arxiu:** BoardModel.java

- **Classe:** BoardModel
- **Mètode:** boolean setCell(int row, int col, int size, char value)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testSetCellCondition()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**

- **Condition coverage:** Avaluem totes les combinacions possibles de cert i fals a les condicions lògiques del if. En aquest cas, comprovem que les condicions *isBetween(row, size)*, *isBetween(col, size)* i *isValidChar(char)* cobreixen tants casos vàlids (retornen *true*) com invàlids (retornen *false*). Això assegura que el mètode gestiona correctament totes les possibilitats i actualitza la cel·la només quan es compleixen les condicions

The screenshot shows an IDE with a Java test method `testSetCellCondition()` and its coverage report. The test method is as follows:

```

53 @Test
54 void testSetCellCondition() {
55
56     // Condition 1 row, false
57     assertFalse(board.setCell(15, 2, board.getSize(), 'S'));
58
59     // Condition 2 row, true
60     assertTrue(board.setCell(2, 2, board.getSize(), 'S'));
61
62     // Condition 1 col, false
63     assertFalse(board.setCell(2, 15, board.getSize(), 'S'));
64
65     // Condition 2 col, true
66     assertTrue(board.setCell(2, 2, board.getSize(), 'S'));
67
68     // Condition 1 size, false
69     assertFalse(board.setCell(2, 2, -1, 'S'));
70
71     // Condition 2 size, true
72     assertTrue(board.setCell(2, 2, 4, 'S'));
73
74     // Condition 1 value, false
75     assertFalse(board.setCell(2, 2, board.getSize(), '?'));
76
77     // Condition 2 value, true
78     assertTrue(board.setCell(2, 2, board.getSize(), 'S'));
79
80
81 }
82
83
84
85
86
87
88

```

The coverage report at the bottom shows the following data:

Element	Coverage	Verified Instructions	Missed Instructions	Total Instructions
MavenBattleship	3,8 %	176	4.512	4.688

**Funcionalitat:** Configurar el valor d'una cel·la del tauler sempre que la fila, columna i caràcter compleixin condicions de validació (rang, mida i tipus de caràcter)

**Localització:** Arxiu: BoardModel.java

- **Classe:** BoardModel
- **Mètode:** boolean setCellDecision(int row, int col, int size, char value)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testSetCellDecision()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Decision coverage:** Assegurem que totes les possibles decisions del if (el bloc s'executa o no) siguin avaluades, és a dir, que el resultat global de la condició sigui tant *true* com *false*.

The screenshot shows an IDE with a Java test method `testSetCellDecision` and its coverage report. The test method is as follows:

```

89
90 @Test
91 void testSetCellDecision() {
92     // Decision 1, false
93
94     assertFalse(board.setCell(2, 2, board.getSize(), '?'));
95
96     // Decision 1, true
97
98     assertTrue(board.setCell(2, 2, board.getSize(), 'S'));
99 }
100
101

```

The coverage report is shown in a table below the code:

Element	Coverage	Vered Instructions	Missed Instructions	Total Instructions
MavenBattleship	2.5 %	118	4.570	4.688

**Funcionalitat:** Valida si un caràcter donat és vàlid per representar un estat d'una cel·la del tauler (aigua, vaixell, impacte, fallada)

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean isValidChar(char value)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testIsValidCharStatementAndCondition()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:** Realitzem condition, decision i State coverage.
  - **Condition coverage:** Cada condició individual (comparació amb wàter, ship, hit i miss) s'avalua tant com a vertadera com a falsa, per garantir que cada caràcter possible es validat correctament
  - **Statement coverage:** assegurem que el mètode respon correctament segons els valors inicialitzats al constructor paramètric de BoardModel.

Value == water	Value == ship	Value == hit	Value == miss	resultat
true	false	false	false	true
false	true	false	false	true
false	false	true	false	true
false	false	false	true	true
false	false	false	false	false

Coverage	Vered Instructions	Missed Instructions	Total Instructions
2,2 %	104	4.584	4.688

```

102 @Test
103 void testIsValidCharStatementAndCondition() {
104
105     // Statement & Condition coverage
106
107     assertTrue(board.isValidChar('-'));
108     assertTrue(board.isValidChar('S'));
109     assertTrue(board.isValidChar('X'));
110     assertTrue(board.isValidChar('O'));
111     assertFalse(board.isValidChar('?'));
112     assertFalse(board.isValidChar('a'));
113 }

```

Element	Coverage	Vered Instructions	Llissed Instructions	Total Instructions
MavenBattleship	2.2 %	104	4,584	4,688

**Funcionalitat:** Valida si un caràcter donat és vàlid per representar un estat d'una cel·la del tauler (aigua, vaixell, impacte, fallada)

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean isValidChar(char value)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** isValidCharDecision()
- **Tipus de test:** Caixa Blanca

**Tècniques utilitzades:**

- **Decision coverage:** Assegurem que el resultat global del mètode (True o False) és correcte en tots els casos comprovant que totes les combinacions de condicions que determinen si un caràcter és vàlid o no són cobertes.

```

114
115 @Test
116 void testIsValidCharDecision() {
117
118     // Decision 1, false
119
120     assertFalse(board.isValidChar('?'));
121
122     // Decision 1, true
123
124     assertTrue(board.isValidChar('S'));
125 }
126

```

Element	Coverage	Vered Instructions	Llissed Instructions	Total Instructions
MavenBattleship	1.8 %	84	4,604	4,688

**Funcionalitat:** Determina si una coordenada (fila o columna) està dins dels límits del tauler

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean isBetween(int coord, int size)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testIsBetweenParticionsEquivalentesAndValorsLimitAndFrontera()
- **Tipus de test:** Caixa Negra

- **Tècniques utilitzades:** Particions equivalents (en aquest 4x4)
  - **Valors frontera:** 0 i 3
  - **Valors límits:** -1, 1, 2,4
  - **Valors de particions equivalents:** -4, 8

**Funcionalitat:** Determina si una coordenada (fila o columna) està dins dels límits del tauler

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean isBetween(int coord, int size)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testIsBetweenCondition()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Condition Coverage:** Comprovem totes les possibles combinacions de cert i fals per a les dues condicions del if: si coord >= 0 i si coord < size. Verifiquem casos en què les condicions falses (coord < 0 o coord >= size) i casos en què són certes, garantint així que el codi cobreix totes les combinacions possibles i es comporta correctament segons els valors d'entrada

Coord >= 0	Coord < size	Resultat
false	true	false
true	true	true
true	false	false
true	true	true

```

143 @Test
144 void testIsBetweenCondition() {
145
146     // Condition 1 coord >= 0, false
147
148     assertFalse(board.isBetween(-1, board.getSize()));
149
150     // Condition 2 coord >= 0, true
151
152     assertTrue(board.isBetween(2, board.getSize()));
153
154     // Condition 3 coord < size, false
155
156     assertFalse(board.isBetween(8, board.getSize()));
157
158     // Condition 4 coord < size, true
159
160     assertTrue(board.isBetween(2, board.getSize()));
161 }
162

```

Element	Coverage	Verified Instructions	Missed Instructions	Total Instructions
MavenBattleship	2.1 %	99	4,589	4,688

**Funcionalitat:** Determina si una coordenada (fila o columna) està dins dels límits del tauler

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel

- **Mètode:** boolean isBetween(int coord, int size)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testIsBetweenDecision()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Condition Coverage:** Avaluem si el resultat global del retorn és true o false. Testegem un cas en què les dues condicions del if no es compleixen (coord < 0 o coord >= size) i un cas en què totes dues són certes, assegurant que la funció pren la decisió correcta en ambdós escenaris iguals

The screenshot shows an IDE with a Java test method `testIsBetweenDecision()` and its coverage report. The test method contains two assertions: `assertFalse(board.isBetween(-1, board.getSize()));` and `assertTrue(board.isBetween(2, board.getSize()));`. The coverage report at the bottom shows that the method is covered 1.8% of the time, with 83 covered instructions out of a total of 4,688 instructions.

**Funcionalitat:** Verifica si una cel·la està buida, és a dir, si el valor de la cel·la és igual al caràcter d'aigua predefinit.

**Localització:**

- **Arxiu:** BoardModel.java
- **Classe:** BoardModel
- **Mètode:** boolean isEmpty(int row, int col)

**Test:**

- **Arxiu:** BoardModelTest.java
- **Classe:** BoardModelTest
- **Mètode:** testIsEmpty()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Decision coverage:** La decisió que afecta al resultat és mirar si el valor de la cel·la és igual al caràcter water. Aquesta cobertura ens garanteix que s'exploren tants els casos on la decisió és vertader (la cel·la esta buida) com falsa (la cel·la no està buida).

Grid[row][col] == water	Resultat
true	true
false	false
false	false
false	false



## **ShipModel:**

**Funcionalitat:** Retorna la fila (row) on està posicionat el vaixell.

**Localització:**

- **Arxiu:** ShipModel.java
- **Classe:** ShipModel
- **Mètode:** getRow()

**Test:**

- **Arxiu:** ShipModelTest.java
- **Classe:** ShipModelTest
- **Mètode:** testGetRowParticionsEquivalentsAndValorsLimitAndFrontera( )
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Particions equivalents:**
    - Valors vàlids (2) i no vàlids (-4 i 8)
    - Valors límit (0 i 3)
    - Valors frontera per 0 (-1 i 1) i per 3 (2 i 3)

Encara que es permeten valors negatius, es valida que aquest valor compleixi l'invariant establert (fila no negativa).

**Funcionalitat:** Retorna la columna (col) on està posicionat el vaixell.

**Localització:**

- **Arxiu:** ShipModel.java
- **Classe:** ShipModel
- **Mètode:** getCol()

**Test:**

- **Arxiu:** ShipModelTest.java
- **Classe:** ShipModelTest
- **Mètode:** testGetColParticionsEquivalentsAndValorsLimitAndFrontera( )
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Particions equivalents:**
    - Valors vàlids (2) i no vàlids (-4 i 8)
    - Valors límit (0 i 3)
    - Valors frontera per 0 (-1 i 1) i per 3 (2 i 3)

Encara que es permeten valors negatius, es valida que aquest valor compleixi l'invariant establert (columna no negativa) mitjançant un assert.

## **BoardView:**

**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.


**Localització:**

- **Arxiu:** BoardView.java
- **Classe:** BoardView

- **Mètode:** printBoard(BoardModel board, boolean hideShips)

**Test:**

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:**
  - testPrintBoardWithHiddenShipsStatement1()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es valida que les línies de codi associades amb el mode ocult (hideShips = true) són executades completament, incloent-hi la substitució dels vaixells ('S') per aigua ('-')

Coverage	covered Instructions	misses Instructions	Total Instructions
 6,6 %	309	4.379	4.688

En aquest cas, prova que el tauler es mostri correctament amb els vaixells ocults (hideShips = true). Fem ús de la simulació d'Output Stream per capturar el format generat pel mètode per comparar-lo amb un tauler buit esperat, assegurant que només es mostrin caselles d'aigua ('-').


**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.

**Localització:**

- **Arxiu:** BoardView.java
- **Classe:** BoardView
- **Mètode:** printBoard(BoardModel board, boolean hideShips)

**Test:**

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testPrintBoardWithVisibleShipsStatement2()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es valida que les línies de codi associades amb el mode ocult (hideShips = false) són executades completament, incloent-hi la substitució dels vaixells ('S') per aigua ('-')

Coverage	covered Instructions	misses Instructions	Total Instructions
 5,8 %	273	4.415	4.688

Hem dividit en 2 funcions el statement coverage ja que eren funcions molt grans i En aquest cas, prova que el tauler es mostri correctament amb els vaixells visibles (hideShips = false). Fem ús de la simulació d'Output Stream per capturar el format generat pel mètode per comparar-lo amb un tauler buit esperat, assegurant que només es mostrin caselles d'aigua ('-').

**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.


**Localització:**

- **Arxiu:** BoardView.java

- **Classe:** BoardView
- **Mètode:** printBoard(BoardModel board, boolean hideShips)

**Test:**

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testPrintEmptyBoardStatement3()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es valida que les condicions per tauler buits són recorregudes en els dos modes (hideShips = true/false)

	Coverage	vered Instructions	lissed Instructions	Total Instructions
	6,2 %	292	4.396	4.688

Es cobreix el comportament del mètode amb l'escenari inicial sense cap acció sobre el tauler. Fem ús de la simulació d'Output Stream per capturar el format generat pel mètode per comparar-lo amb un tauler buit esperat, assegurant que només es mostrin caselles d'aigua ('-').


**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.

**Localització:**

- **Arxiu:** BoardView.java
- **Classe:** BoardView
- **Mètode:** printBoard(BoardModel board, boolean hideShips)

**Test:**

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testPrintEmptyBoardStatement4()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es recorren totes les línies del mètode per taulers plens amb diferents configuracions.
    - Quan el tauler esta ple de vaixells amb els vaixells sense amagar
    - Quan el tauler esta ple de vaixells amb els vaixells amagats
    - Quan el tauler esta ple de hits sense amagar
    - Quan el tauler esta ple de misses sense amagar

	Coverage	vered Instructions	lissed Instructions	Total Instructions
	12,0 %	562	4.126	4.688

**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.

**Localització:**

- **Arxiu:** BoardView.java
- **Classe:** BoardView
- **Mètode:** printBoard(BoardModel board, boolean hideShips)

#### Test:

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testPrintBoardLoopSimple()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es recorren bucles de diferents dimensions segons la mida del tauler cobrint el comportament en taulers de mida petita i gran, validant la correcció de l'output.

```

152
153
154 @Test
155 void testPrintFullBoardStatement4() {
156     // Amb Ships a tot el board i ships visibles
157
158     BoardModel board = new BoardModel(4, '-', 'S', 'X', 'O');
159     for (int row = 0; row < board.getSize(); row++) {
160         for (int col = 0; col < board.getSize(); col++) {
161             board.setCell(row, col, board.getSize(), 'S');
162         }
163     }
164
165     BoardView boardView = new BoardView();
166
167     ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
168     System.setOut(new PrintStream(outputStream));
169
170     boardView.printBoard(board, false);
171
172     System.setOut(System.out);
173
174     String lineSeparator = System.lineSeparator();
175     String expectedOutput =
176         " 1 2 3 4 " + lineSeparator +
177         "1 S S S S " + lineSeparator +
178         "2 S S S S " + lineSeparator +
179         "3 S S S S " + lineSeparator +
180         "4 S S S S " + lineSeparator;
181
182     assertEquals(
183         expectedOutput.replace("\r\n", "\n"),
184         outputStream.toString().replace("\r\n", "\n")
185     );
186
187     // Amb Ships a tot el board i ships no visibles
188
189     BoardModel board2 = new BoardModel(4, '-', 'S', 'X', 'O');
190     for (int row = 0; row < board2.getSize(); row++) {
191         for (int col = 0; col < board2.getSize(); col++) {
192             board2.setCell(row, col, board2.getSize(), 'S');
193         }
194     }
195 }

```

Element	Coverage	vered Instructions	lissed Instructions	Total Instructions
> MavenBattleship	12,0 %	562	4.126	4.688

**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.

#### Localització:

- **Arxiu:** BoardView.java
- **Classe:** BoardView
- **Mètode:** printBoard(BoardModel board, boolean hideShips)

#### Test:

- **Arxiu:** BoardViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testPrintBoardLoopsNotSimplesWithMockito()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**

- **Mockito:** Simulem el comportament de BoardModel, comprovant que els mètodes associats (getGrid(), getSize()) són cridats correctament.

```

437
438
439 @Test
440 void testPrintBoardLoopsNotSimplesWithMockito() {
441     // Creem una matriu 8x8 per simular el grid
442     // given
443
444     char[][] mockGrid = new char[8][8];
445     for (int i = 0; i < 8; i++) {
446         for (int j = 0; j < 8; j++) {
447             mockGrid[i][j] = '-';
448         }
449     }
450
451     // Creació del mock de BoardModel fent servir Mockito
452     BoardModel mockBoard = mock(BoardModel.class);
453
454     // when
455
456     when(mockBoard.getGrid()).thenReturn(mockGrid);
457     when(mockBoard.getSize()).thenReturn(8);
458     when(mockBoard.getShipChar()).thenReturn('S');
459     when(mockBoard.getCell(0, 0)).thenReturn('-');
460     when(mockBoard.getWater(0, 0)).thenReturn('-');
461
462     // then
463
464     BoardView boardView = new BoardView();
465     boardView.printBoard(mockBoard, false);
466
467     // Verifiquem que el mètode getGrid() hagi estat utilitzat només un cop
468     verify(mockBoard, times(1)).getGrid();
469 }
470
471
472
473
474
475
476

```

Element	Coverage	vered Instructions	lissed Instructions	Total Instructions
> MavenBattleship	4,0 %	188	4.500	4.688

## InputView:

**Funcionalitat:** Llegeix un número enter introduït per l'usuari des de la consola, gestionant entrades invàlides mitjançant un bucle fins que es proporciona un valor vàlid.

**Localització:**

- **Arxiu:** InputView.java
- **Classe:** InputView
- **Mètode:** getInInput(String prompt)

**Test:**

- **Arxiu:** InputViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testgetInInputStatement()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Statement coverage:** Es verifica que totes les línies de codi, incloent-hi el bucle de gestió d'errors, són executades.

Coverage	vered Instructions	lissed Instructions	Total Instructions
1,5 %	72	4.616	4.688

**Funcionalitat:** Llegeix un número enter introduït per l'usuari des de la consola, gestionant entrades invàlides mitjançant un bucle fins que es proporciona un valor vàlid.

**Localització:**

- **Arxiu:** InputView.java
- **Classe:** InputView
- **Mètode:** getInInput(String prompt)

**Test:**

- **Arxiu:** InputViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testValidInputParticionsEquivalentsAndValorsLimitAndFrontera()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Particions equivalents:** Inputs vàlids segons una matriu 4x4
    - Valors límits (0 i 3)
    - Valors frontera (-1,1,2 i 4)
    - Particions equivalent: valors invàlid (-4 i 5), valors vàlids (2)

**Funcionalitat:** Llegeix un número enter introduït per l'usuari des de la consola, gestionant entrades invàlides mitjançant un bucle fins que es proporciona un valor vàlid.

**Localització:**

- **Arxiu:** InputView.java
- **Classe:** InputView
- **Mètode:** getInInput(String prompt)

**Test:**

- **Arxiu:** InputViewTest.java
- **Classe:** InputViewTest
- **Mètode:** testOutOfRangeInputsParticionsEquivalentsAndValorsLimitAndFrontera()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - Particions equivalents, valors límit i valors frontera.  
Valors a provar: -1 i 4.

**Funcionalitat:** Mostra el tauler a la consola amb l'opció de ocultar o mostrar els vaixells segons el valor de hideShips.

**Localització:**

- **Arxiu:** InputView.java
- **Classe:** InputView
- **Mètode:** getInInput(String prompt)

**Test:**

- **Arxiu:** InputViewTest.java
- **Classe:** BoardViewTest
- **Mètode:** testInvalidInputParticionsEquivalentsAndValorsLimitAndFrontera()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - Particions equivalents, valors límit i valors frontera.  
Valors a provar: a, abcd, 3.1415 i 2.

## **MessageView:**


**Funcionalitat:** Mostra per pantalla un missatge dependent de la situació i context de l'execució del joc i del valor que ha estat prèviament introduït per l'usuari des de la consola, gestionant entrades invàlides.

### **Localització:**

- **Arxiu:** MessageView.java
- **Classe:** MessageView
- **Mètode:** showMessage(String message)

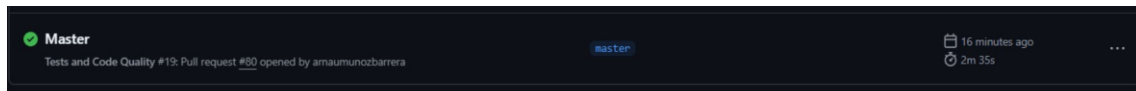
### **Test:**

- **Arxiu:** MessageViewTest.java
- **Classe:** MessageViewTest
- **Mètode:** testShowMessageStatement1() i  
testShowMessage\_EmptyMessageStatement2()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - o **Statement coverage:** Es verifica que totes les línies de codi són executades.

Coverage	covered Instructions	misses Instructions	Total Instructions
 1,1 %	51	4.637	4.688

## CD & CI files:

Amb resultat correcte dels tests a github:



Amb resultat incorrecte dels tests a github:

