Informe de Pràctiques - Test i Qualitat del Software

Assignatura	**Test i Qualitat del Software (Curs 2025-2026)**
Alumnes - NIU	Levon Kesoyan Galstyan - 1668018 Luis Martinez Zamora - 1668180
Projecte	Battleship
Repositori	https://github.com/luismz14/BattleShipTDD

1. Introducció

Aquest document detalla l'estratègia de proves i qualitat implementada per al desenvolupament del projecte *Battleship*. El projecte s'ha desenvolupat des de zero seguint la metodologia TDD, assegurant una arquitectura MVC i aplicant tècniques avançades de test (Caixa Negra, Caixa Blanca, Mocks i Automatització).

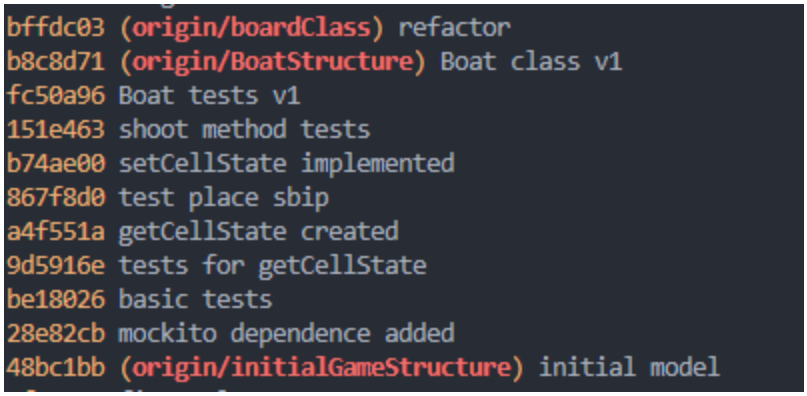
2. Arquitectura MVC

El codi s'ha estructurat seguint el patró Model-Vista-Controlador per facilitar la testejabilitat aïllada de cada component, especialment del Model i el Controlador, sense dependències de la interfície d'usuari.

- **Model (es.uab.tqs.battleship.model):** Conté la lògica del joc (Game, Board, Ship, Coordinate, Cell). Aquest paquet no té cap dependència de la Vista ni de llibreries externes d'entrada/sortida.
- **Vista (es.uab.tqs.battleship.view):** Defineix la interfície GameView i la implementació ConsoleView. És totalment passiva i es limita a mostrar l'estat del joc.
- **Controlador (es.uab.tqs.battleship.controller):** Orquestra el flux del joc (GameController, BoardController). Connecta Model i Vista mitjançant injecció de dependències.

3. TDD (Test Driven Development)

S'ha aplicat la metodologia TDD per a totes les classes i mètodes, seguint l'ordre Test→Codi→Refactor (si fos necessari). Això es pot veure al git log.



4. Proves de Caixa Negra

S'han dissenyat proves funcionals per validar les especificacions sense inspeccionar el codi intern.

4.1. Particions Equivalents i Valors Límit

S'han analitzat exhaustivament les entrades de mètodes crítics. Un exemple clar és la classe Coordinate.

Exemple: CoordinateTest
Es verifiquen coordenades basant-se en la mida del tauler (N=10).

Tipus de Valor	Input (X, Y)	Resultat Esperat	Test Implementat
Vàlid (Central)	5, 5	true	testValidCenterCoordinate
Límit Inferior	0, 0	true	testBoundaryMinCoordinate
Límit Superior	9, 9	true	testBoundaryMaxCoordinate
Invàlid (Negatiu)	-1, 5	false	testInvalidNegativeX
Invàlid (Frontera Sup)	10, 10	false	testBoundaryInvalidMaxCoordinate

4.2. Pairwise Testing

S'ha realitzat pairwise testing a les funcions on ha sigut convenient. Un exemple és

Ship.setPosition(Coordinate start, Orientation orientation):

- **Mètode testejat:** Ship.setPosition(Coordinate start, Orientation orientation).
- **Variables combinades:** Coordenada X, Coordenada Y, Orientació, Tipus de Vaixell.
- **Implementació:** Veure ShipTest.testSetPositionPairwise. S'utilitza l'anotació @CsvSource per injectar només els parells necessaris en lloc de provar totes les combinacions possibles, reduint dràsticament el nombre de tests sense perdre cobertura funcional.

5. Automatització (Data Driven Testing)

S'ha implementat *Data Driven Testing* en dos mètodes clau utilitzant JUnit 5 (@ParameterizedTest).

1. **Validació de Coordenades (CoordinateTest):**
   - Mètode de test: testIsValidWithMultipleData.
   - Descripció: S'injecten 16 casos de prova diferents (vàlids, límits i invàlids) en un sol mètode de test per verificar la robustesa de Coordinate.isValid().
2. **Enfonsament de Vaixells (ShipTest):**
   - Mètode de test: testIsSunkDataDriven.
   - Descripció: Es proven automàticament múltiples combinacions de tipus de vaixell i nombre d'impactes rebuts per verificar si l'estat isSunk() es calcula correctament segons la longitud del vaixell.

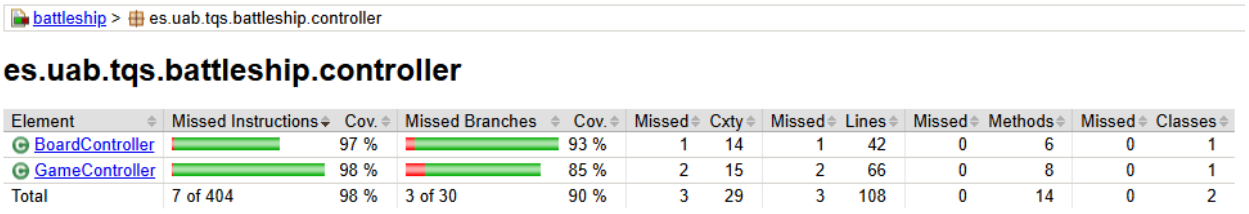
6. Proves de Caixa Blanca

S'ha analitzat l'estructura interna del codi per garantir la màxima cobertura estructural i lògica.

6.1. Cobertura de Sentències (Statement Coverage)

A continuació adjuntem les captures de pantalla on s'esmenten els percentatges de coverage de cada mètode:

CONTROLLER:



battleship > es.uab.tqs.battleship.controller > GameController

GameController

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processComputerTurn()		94 %		75 %	1	3	2	15	0	1
processPlayerTurn()		100 %		100 %	0	3	0	16	0	1
startGame()		100 %		n/a	0	1	0	10	0	1
displayFinalResult()		100 %		n/a	0	1	0	7	0	1
playGameLoop()		100 %		83 %	1	4	0	7	0	1
GameController(GameView)		100 %		n/a	0	1	0	5	0	1
GameController(GameView, Game, BoardController)		100 %		n/a	0	1	0	5	0	1
getGame()		100 %		n/a	0	1	0	1	0	1
Total	3 of 237	98 %	2 of 14	85 %	2	15	2	66	0	8

battleship > es.uab.tqs.battleship.controller > BoardController

BoardController

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setupPlayerShips(Board)		94 %		83 %	1	4	1	18	0	1
displayBoardStats(Board)		100 %		n/a	0	1	0	8	0	1
getRemainingShips(Board)		100 %		100 %	0	3	0	6	0	1
isValidAttack(Board, Coordinate)		100 %		100 %	0	3	0	4	0	1
attemptPlaceShip(Board, Ship, Coordinate, Orientation)		100 %		100 %	0	2	0	3	0	1
BoardController(GameView)		100 %		n/a	0	1	0	3	0	1
Total	4 of 167	97 %	1 of 16	93 %	1	14	1	42	0	6

MODEL:

battleship > es.uab.tqs.battleship.model

es.uab.tqs.battleship.model

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Ship		94 %		100 %	1	16	1	30	1	11	0	1
Game		97 %		84 %	4	24	2	57	0	11	0	1
Coordinate		97 %		93 %	1	15	0	13	0	7	0	1
Board		100 %		95 %	2	34	0	58	0	10	0	1
Cell		100 %		90 %	1	14	0	22	0	9	0	1
ShipType		100 %		n/a	0	4	0	12	0	4	0	1
AttackResult		100 %		n/a	0	3	0	9	0	3	0	1
GameStatus		100 %		n/a	0	1	0	7	0	1	0	1
CellState		100 %		n/a	0	1	0	5	0	1	0	1
Orientation		100 %		n/a	0	1	0	3	0	1	0	1
Total	15 of 926	98 %	8 of 110	92 %	9	113	3	216	1	58	0	10

battleship > es.uab.tqs.battleship.model > Ship

Ship

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
toString()		0 %		n/a	1	1	1	1	1	1
setPosition(Coordinate, Orientation)		100 %		100 %	0	3	0	10	0	1
occupiesCoordinate(Coordinate)		100 %		100 %	0	3	0	5	0	1
Ship(ShipType)		100 %		n/a	0	1	0	6	0	1
isSunk()		100 %		100 %	0	2	0	1	0	1
registerHit()		100 %		n/a	0	1	0	2	0	1
getCoordinates()		100 %		n/a	0	1	0	1	0	1
getLength()		100 %		n/a	0	1	0	1	0	1
getType()		100 %		n/a	0	1	0	1	0	1
getOrientation()		100 %		n/a	0	1	0	1	0	1
getHitCount()		100 %		n/a	0	1	0	1	0	1
Total	7 of 121	94 %	0 of 10	100 %	1	16	1	30	1	11

battleship > es.uab.tqs.battleship.model > Game

Game

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processComputerAttack()		86 %		75 %	1	3	1	8	0	1
getWinner()		85 %		75 %	1	3	1	5	0	1
placeComputerShipsRandomly()		100 %		100 %	0	4	0	14	0	1
processPlayerAttack(Coordinate)		100 %		100 %	0	3	0	7	0	1
generateRandomAttack()		100 %		50 %	1	2	0	6	0	1
Game()		100 %		n/a	0	1	0	6	0	1
isGameOver()		100 %		100 %	0	3	0	5	0	1
startGame()		100 %		50 %	1	2	0	3	0	1
getPlayerBoard()		100 %		n/a	0	1	0	1	0	1
getComputerBoard()		100 %		n/a	0	1	0	1	0	1
getStatus()		100 %		n/a	0	1	0	1	0	1
Total	6 of 210	97 %	4 of 26	84 %	4	24	2	57	0	11

battleship > es.uab.tqs.battleship.model > Coordinate

Coordinate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
equals(Object)		92 %		87 %	1	5	0	4	0	1
isValid(int)		100 %		100 %	0	5	0	1	0	1
hashCode()		100 %		n/a	0	1	0	1	0	1
Coordinate(int, int)		100 %		n/a	0	1	0	4	0	1
toString()		100 %		n/a	0	1	0	1	0	1
getX()		100 %		n/a	0	1	0	1	0	1
getY()		100 %		n/a	0	1	0	1	0	1
Total	2 of 82	97 %	1 of 16	93 %	1	15	0	13	0	7

battleship > es.uab.tqs.battleship.model > Board

Board

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
isValidPlacement(Ship, Coordinate, Orientation)		100 %		100 %	0	10	0	16	0	1
Board(int)		100 %		100 %	0	3	0	8	0	1
placeShip(Ship, Coordinate, Orientation)		100 %		100 %	0	3	0	8	0	1
processAttack(Coordinate)		100 %		100 %	0	5	0	12	0	1
getCell(int, int)		100 %		75 %	2	5	0	3	0	1
allShipsSunk()		100 %		100 %	0	4	0	7	0	1
getCell(Coordinate)		100 %		n/a	0	1	0	1	0	1
getShips()		100 %		n/a	0	1	0	1	0	1
getShipCount()		100 %		n/a	0	1	0	1	0	1
getSize()		100 %		n/a	0	1	0	1	0	1
Total	0 of 261	100 %	2 of 48	95 %	2	34	0	58	0	10

battleship > es.uab.tqs.battleship.model > Cell

Cell

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
attack()		100 %		100 %	0	2	0	6	0	1
Cell(Coordinate)		100 %		n/a	0	1	0	5	0	1
isAlreadyAttacked()		100 %		100 %	0	3	0	1	0	1
setShip(Ship)		100 %		50 %	1	2	0	4	0	1
hasShip()		100 %		100 %	0	2	0	1	0	1
setState(CellState)		100 %		n/a	0	1	0	2	0	1
getState()		100 %		n/a	0	1	0	1	0	1
getCoordinate()		100 %		n/a	0	1	0	1	0	1
getShip()		100 %		n/a	0	1	0	1	0	1
Total	0 of 69	100 %	1 of 10	90 %	1	14	0	22	0	9

battleship > es.uab.tqs.battleship.model > ShipType

ShipType

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
static {...}		100 %		n/a	0	1	0	6	0	1
ShipType(String, int, int, String)		100 %		n/a	0	1	0	4	0	1
getLength()		100 %		n/a	0	1	0	1	0	1
getDisplayName()		100 %		n/a	0	1	0	1	0	1
Total	0 of 60	100 %	0 of 0	n/a	0	4	0	12	0	4

battleship > es.uab.tqs.battleship.model > AttackResult

AttackResult

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
static {...}		100 %		n/a	0	1	0	5	0	1
AttackResult(String, int, String)		100 %		n/a	0	1	0	3	0	1
getMessage()		100 %		n/a	0	1	0	1	0	1
Total	0 of 42	100 %	0 of 0	n/a	0	3	0	9	0	3

battleship > es.uab.tqs.battleship.model > GameStatus

GameStatus

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
static {...}		100 %		n/a	0	1	0	7	0	1
Total	0 of 39	100 %	0 of 0	n/a	0	1	0	7	0	1

battleship > es.uab.tqs.battleship.model > CellState

CellState

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
static {...}		100 %		n/a	0	1	0	5	0	1
Total	0 of 27	100 %	0 of 0	n/a	0	1	0	5	0	1

battleship > es.uab.tqs.battleship.model > Orientation

Orientation

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
static {...}		100 %		n/a	0	1	0	3	0	1
Total	0 of 15	100 %	0 of 0	n/a	0	1	0	3	0	1

Com podem veure, s'ha assolit el màxim percentatge en la gran majoria de mètodes, passant del 90% en pràcticament totes, juntament amb un cobertura superior al 90% en cada classe.

6.2. Decision & Condition Coverage

S'ha verificat que totes les branques (true/false) de les condicions complexes s'executen en almenys 3 mètodes crítics.

Mètodes Analitzats:

1. **Board.isValidPlacement:** Es verifiquen les condicions de límits del tauler i solapament de vaixells (múltiples if i punts de retorn false).

```java
public boolean isValidPlacement(Ship ship, Coordinate start, Orientation orientation) {
    int length = ship.getLength();
    int x = start.getX();
    int y = start.getY();

    if (x < 0 || x >= size || y < 0 || y >= size) {
        return false;
    }

    for (int i = 0; i < length; i++) {
        int currentX = x;
        int currentY = y;

        if (orientation == Orientation.HORIZONTAL) {
            currentX += i;
        } else {
            currentY += i;
        }

        if (currentX >= size || currentY >= size) {
            return false;
        }

        if (cells[currentX][currentY].hasShip()) {
            return false;
        }
    }

    return true;
}
```

2. **Board.processAttack:** Es cobreixen els camins de coordenades invàlides, cel·les ja atacades, resultats de *HIT*, *MISS* i *SUNK*.

```java
public AttackResult processAttack(Coordinate coordinate) {
    if (!coordinate.isValid(size)) {
        throw new IllegalArgumentException("Invalid coordinate: " + coordinate);
    }

    Cell cell = getCell(coordinate);

    if (cell.isAlreadyAttacked()) {
        return AttackResult.ALREADY_ATTACKED;
    }

    boolean hit = cell.attack();

    if (hit) {
        Ship ship = cell.getShip();
        if (ship.isSunk()) {
            return AttackResult.SUNK;
        }
        return AttackResult.HIT;
    } else {
        return AttackResult.MISS;
    }
}
```

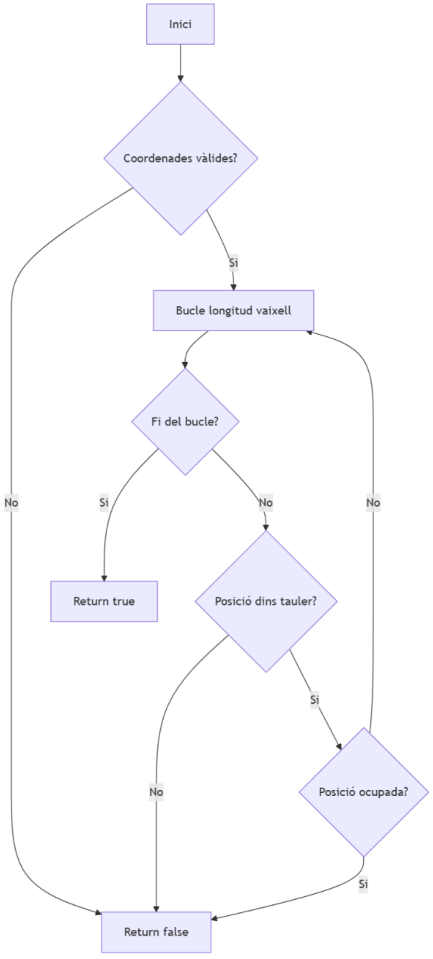
3. **GameController.processPlayerTurn:** Es verifiquen els fluxos normals i la captura d'excepcions (IllegalArgumentException).

```java
public void processPlayerTurn() {
    view.displayMessage(message: "\n=== YOUR TURN ===\n");
    view.displayMessage(message: "Enemy board:");
    view.displayBoard(game.getComputerBoard(), hideShips: true);

    view.displayMessage(message: "\nYour board:");
    view.displayBoard(game.getPlayerBoard(), hideShips: false);

    Coordinate target = view.getCoordinateInput(prompt: "Enter coordinates to attack");

    try {
        AttackResult result = game.processPlayerAttack(target);
        view.displayAttackResult(result);

        if (result == AttackResult.SUNK) {
            view.displayMessage(message: "You have sunk an enemy ship!\n");
        }

        if (result == AttackResult.HIT) {
            view.displayMessage(message: "You hitted an enemy ship!\n");
        }

    } catch (IllegalArgumentException e) {
        view.displayMessage("Error: " + e.getMessage());
    }
}
```

6.3. Path Coverage

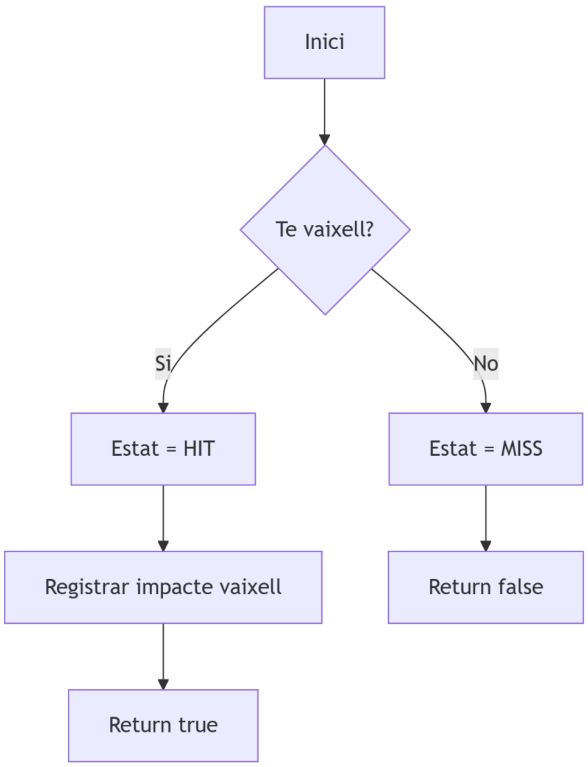
S'ha realitzat l'anàlisi de camins linealment independents per a dos mètodes, documentant-los amb diagrames de flux.

Mètode 1: Board.isValidPlacement
Aquest mètode conté lògica crítica per evitar col·locar vaixells fora del tauler o sobreposats.
Diagrama de Flux (Board.isValidPlacement):



```java
public boolean isValidPlacement(Ship ship, Coordinate start, Orientation orientation) {
    int length = ship.getLength();
    int x = start.getX();
    int y = start.getY();

    if (x < 0 || x >= size || y < 0 || y >= size) {
        return false;
    }

    for (int i = 0; i < length; i++) {
        int currentX = x;
        int currentY = y;

        if (orientation == Orientation.HORIZONTAL) {
            currentX += i;
        } else {
            currentY += i;
        }

        if (currentX >= size || currentY >= size) {
            return false;
        }

        if (cells[currentX][currentY].hasShip()) {
            return false;
        }
    }

    return true;
}
```

Mètode 2: Cell.attack
Gestiona la transició d'estats de la cel·la quan rep un atac.
Diagrama de Flux (Cell.attack):




```
41      public boolean attack() {
42          if (hasShip()) {
43              state = CellState.HIT;
44              ship.registerHit();
45              return true;
46          } else {
47              // water
48              state = CellState.MISS;
49              return false;
50          }
51      }
```

6.4. Loop Testing

S'han verificat el bucle de la funció playGameLoop()

```java
@Test
public void testPlayGameLoop_AlternatesTurnsAndEnds() {
    // First iteration: -> Player Turn
    // Second iteration: -> Computer Turn
    // Third iteration: -> Game Over
    when(mockGame.isGameOver())
        .thenReturn(value: false)
        .thenReturn(value: false)
        .thenReturn(value: true);

    when(mockGame.getStatus())
        .thenReturn(GameStatus.PLAYER_TURN)
        .thenReturn(GameStatus.COMPUTER_TURN);

    when(mockView.getCoordinateInput(anyString())).thenReturn(mock(classToMock: Coordinate.class));
    when(mockGame.processComputerAttack()).thenReturn(mock(classToMock: Coordinate.class));

    when(mockPlayerBoard.getCell(any())).thenReturn(mock(classToMock: Cell.class));

    controller.playGameLoop();

    verify(mockGame, atLeast(minNumberOfInvocations: 3)).isGameOver();
    verify(mockView, atLeastOnce()).getCoordinateInput(anyString());
    verify(mockGame, atLeastOnce()).processComputerAttack();
}
```

7. Mock Objects

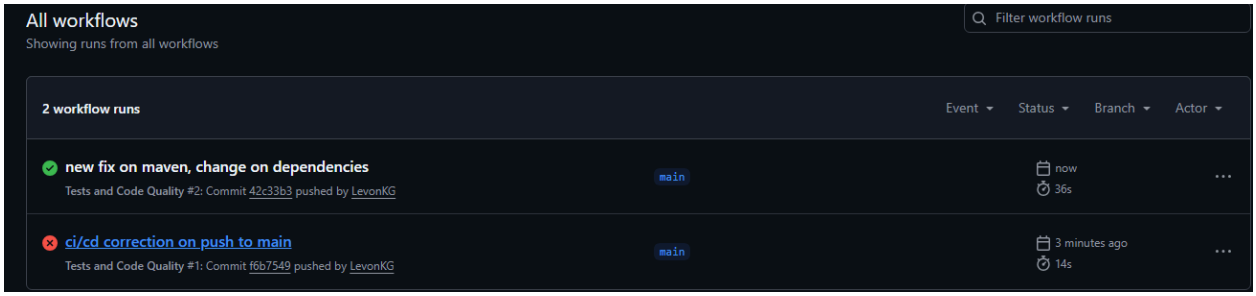
S'ha utilitzat la llibreria Mockito per aïllar completament les proves unitàries.

1. **Mock de la Vista (GameControllerTest):** S'utilitza GameView mockView per verificar que el controlador envia els missatges correctes (verify(mockView).displayMessage(...)) sense dependre de la consola real.
2. **Mocks del Model (BoardControllerTest):**
    - **Board mockBoard:** Es simula el comportament del tauler (when(mockBoard.isValidPlacement...).thenReturn(...)) per provar la lògica del controlador sense instanciar un tauler real complex.
    - **Ship mockShip:** S'utilitzen mocks de vaixells per forçar estats (when(mockShip.isSunk()).thenReturn(true)) i verificar el recompte de vaixells restants (getRemainingShips).

8. Integració Contínua (CI/CD) i Qualitat

S'ha configurat un pipeline a **GitHub Actions** per garantir la qualitat del codi automàticament.

- **Configuració:** S'ha creat l'arxiu .github/workflows/test.yml que defineix el flux de treball.
- **Automatització:** Compilació i execució automàtica de mvn test en cada *Push* i *Pull Request* a la branca main.
- **Qualitat del codi:** Amb cada merge o pull request a main, es garantitza la qualitat del codi seguint unes normes preestablertes a l'arxiu "checkstyle.xml".


All workflows
Showing runs from all workflows
Filter workflow runs
2 workflow runs
Event
Status
Branch
Actor
new fix on maven, change on dependencies
Tests and Code Quality #2: Commit 42c33b3 pushed by LevonKG
main
now
36s
ci/cd correction on push to main
Tests and Code Quality #1: Commit f6b7549 pushed by LevonKG
main
3 minutes ago
14s