Funcionalitat : Creació i propietats de Coordenades

Localització :

- Arxiu: Coordinate.java
- Classe: Coordinate
- Mètode: Coordinate(int x, int y)

Test :

- Arxiu: CoordinateTest.java
- Classe: CoordinateTest
- Mètode: testGetX() i testGetY()
- **Tipus de test:** Caixa Blanca / Unitari
- **Tècniques utilitzades:**
  - **Statement Coverage:** Es verifica la correcta inicialització de l'objecte inmutable Coordinate i l'accés als seus atributs. Es garanteix que el valor injectat al constructor és exactament el retornat pel *getter*.

```
36.     /**
37.      * Gets the X-coordinate.
38.      *
39.      * @return The x value.
40.      */
41.     public int getX() {
42.         return x;
43.     }
44.
45.     /**
46.      * Gets the Y-coordinate.
47.      *
48.      * @return The y value.
49.      */
50.     public int getY() {
51.         return y;
52.     }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getY()		100 %		n/a	0	1	0	1	0	1
getX()		100 %		n/a	0	1	0	1	0	1

Funcionalitat : Validació de límits del tauler (Valors Extrems)

Localització :

- Arxiu: Coordinate.java
- Classe: Coordinate
- Mètode: isValid(int boardSize)

Test :

- Arxiu: CoordinateTest.java
- Classe: CoordinateTest
- Mètode: testBoundaryInvalidMaxCoordinate() i testInvalidNegativeX()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Boundary Value Analysis (Valors Límit):** Es comprova específicament el comportament en els límits exactes.
  - Es verifica que una coordenada igual a la mida del tauler (ex: 10 en un tauler de 10) retorna false.
  - Es verifica que valors negatius (ex: -1) són rebutjats correctament.

```
25.     /**
26.      * Validates if the coordinate lies within the boundaries of a square board.
27.      *
28.      * @param boardSize The size of the board (e.g., 10 for a 10x10 grid).
29.      * @return true if the coordinate is >= 0 and < boardSize for both axes; false
30.      *         otherwise.
31.      */
32.     public boolean isValid(int boardSize) {
33.         return x >= 0 && x < boardSize && y >= 0 && y < boardSize;
34.     }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
isValid(int)		100 %		100 %	0	5	0	1	0	1

Funcionalitat : Validació robusta de coordenades (Parametritzada)

Localització :

- Arxiu: Coordinate.java
- Classe: Coordinate
- Mètode: isValid(int boardSize)

Test :

- Arxiu: CoordinateTest.java
- Classe: CoordinateTest
- Mètode: testIsValidWithMultipleData(...)
- **Tipus de test:** Caixa Negra / Data Driven
- **Tècniques utilitzades:**
  - **Data Driven Testing:** Ús de @CsvSource per executar la mateixa lògica de validació amb 16 jocs de dades diferents.
  - **Equivalence Partitioning:** Es cobreixen particions vàlides (0,0), (9,9) i invàlides (10,10), (-1,5) en una sola execució massiva.

```
164     /**
165      * Test Case: Data-Driven validation of coordinates (Multiple Scenarios).
166      * * Type: Black Box Testing / Data Driven Testing
167      * * Technique: Parameterized Testing (CsvSource).
168      * * Description: Automatically executes the same test logic against a wide range of
169      * inputs defined in a CSV format. This covers valid cases, boundary values, negatives,
170      * and varying board sizes in a single method, maximizing test coverage efficiency.
171      * * Inputs: X, Y, BoardSize, ExpectedResult.
172      */
173     @ParameterizedTest()
174     @CsvSource({
175         "0, 0, 10, true",
176         "9, 9, 10, true",
177         "5, 5, 10, true",
178         "0, 9, 10, true",
179         "9, 0, 10, true",
180         "10, 10, 10, false",
181         "-1, 5, 10, false",
182         "5, -1, 10, false",
183         "-1, -1, 10, false",
184         "0, 10, 10, false",
185         "10, 0, 10, false",
186         "11, 5, 10, false",
187         "5, 11, 10, false",
188         "1, 1, 5, true",
189         "4, 4, 5, true",
190         "5, 5, 5, false"
191     })
192     public void testIsValidWithMultipleData(int x, int y, int boardSize, boolean expected) {
193         Coordinate coord = new Coordinate(x, y);
194         assertEquals(expected, coord.isValid(boardSize));
195     }
```

Funcionalitat : Gestió d'estats de la Cel·la

Localització :

- Arxiu: Cell.java
- Classe: Cell
- Mètode: setShip(Ship ship)

Test :

- Arxiu: CellTest.java
- Classe: CellTest
- Mètode: testSetShip()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **State Transition Testing:** Es verifica el canvi d'estat de la cel·la des de EMPTY a SHIP.
  - Es comprova que, en assignar un vaixell, la cel·la guarda la referència correcta de l'objecte Ship i actualitza el seu CellState.

```
65.     /**
66.      * Places a ship in this cell.
67.      * If a ship is placed, the cell's state is updated to SHIP.
68.      *
69.      * @param ship The ship to place in this cell.
70.      */
71.     public void setShip(Ship ship) {
72.         this.ship = ship;
73.         if (ship != null) {
74.             this.state = CellState.SHIP;
75.         }
76.     }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setShip(Ship)		100 %		50 %	1	2	0	4	0	1

Funcionalitat : Mecànica d'atac a la Cel·la

Localització :

- Arxiu: Cell.java
- Classe: Cell
- Mètode: attack()

Test :

- Arxiu: CellTest.java
- Classe: CellTest
- Mètode: testAttackCellWithShip() i testAttackEmptyCell()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Equivalence Partitioning:** Es divideixen els casos d'atac en dues classes:
    1. **Hit:** Atacar una cel·la amb vaixell -> Retorna true, estat HIT, incrementa dany al vaixell.
    2. **Miss:** Atacar una cel·la buida -> Retorna false, estat MISS.

```
87.     /**
88.      * Processes an attack on this cell.
89.      * Updates the cell's state to HIT or MISS based on whether a ship is present.
90.      * If a ship is hit, the hit is registered on the ship object.
91.      *
92.      * @return true if the attack was a hit; false if it was a miss.
93.      */
94.     public boolean attack() {
95.         if (hasShip()) {
96.             state = CellState.HIT;
97.             ship.registerHit();
98.             return true;
99.         } else {
100.            // No ship present, so it's a miss (water)
101.            state = CellState.MISS;
102.            return false;
103.        }
104.    }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
attack()		100 %		100 %	0	2	0	6	0	1

Funcionalitat : Definició de Tipus de Vaixells

Localització :

- Arxiu: ShipType.java
- Classe: ShipType
- Mètode: getLength()

Test :

- Arxiu: ShipTest.java
- Classe: ShipTest
- Mètode: testShipTypeLength(ShipType type)
- **Tipus de test:** Caixa Negra / Parametritzat
- **Tècniques utilitzades:**
  - **Equivalence Partitioning (Enum Source):** S'itera automàticament sobre tots els valors de l'enumeració ShipType.
  - Es verifica que cada tipus de vaixell s'inicialitza amb la longitud definida per les regles del joc (Ex: CARRIER = 5, DESTROYER = 2).

```
40.     /**
41.      * Gets the length of the ship based on its type.
42.      *
43.      * @return The length of the ship (number of cells).
44.      */
45.     public int getLength() {
46.         return type.getLength();
47.     }
```

getLength()		100 %		n/a	0	1	0	1	0	1

Funcionalitat : Càlcul de posicions del vaixell

Localització :

- Arxiu: Ship.java
- Classe: Ship
- Mètode: setPosition(Coordinate start, Orientation orientation)

Test :

- Arxiu: ShipTest.java
- Classe: ShipTest
- Mètode: testSetPositionPairwise(...)
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Pairwise Testing:** Tècnica combinatòria per reduir el nombre de casos de prova.
  - S'han seleccionat parells ortogonals de (Coordenada X, Coordenada Y, Orientació, Tipus Vaixell) per maximitzar la cobertura de defectes amb el mínim nombre de tests, cobrint casos com vaixells grans en posició vertical a les vores.

```java
/**
 * Test Case: Verify ship positioning with combinatorial inputs.
 * * Type: Black Box Testing / Data Driven Testing
 * * Technique: Pairwise Testing / Parameterized Test.
 * * Description: Tests multiple combinations of coordinates (X, Y), orientations,
 * and ship types to verify that 'setPosition' correctly calculates the ship's footprint.
 * Using CsvSource allows covering diverse scenarios (corner cases, mixed orientations) efficiently.
 * * Inputs: X, Y, Orientation, ShipType.
 */
@ParameterizedTest()
@CsvSource({
    "0, 0, HORIZONTAL, CARRIER",
    "0, 0, VERTICAL, DESTROYER",
    "5, 5, HORIZONTAL, DESTROYER",
    "5, 5, VERTICAL, CARRIER",
    "9, 0, HORIZONTAL, CRUISER",
    "0, 9, VERTICAL, SUBMARINE",
    "9, 9, HORIZONTAL, BATTLESHIP",
    "1, 1, VERTICAL, BATTLESHIP",
    "2, 3, HORIZONTAL, SUBMARINE",
    "3, 2, VERTICAL, CRUISER"
})
public void testSetPositionPairwise(int x, int y, Orientation orientation, ShipType type) {
    Ship ship = new Ship(type);
    Coordinate start = new Coordinate(x, y);

    ship.setPosition(start, orientation);

    assertEquals(orientation, ship.getOrientation());
    assertEquals(type.getLength(), ship.getCoordinates().size());
    assertEquals(start, ship.getCoordinates().get(index: 0));
}
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setPosition(Coordinate, Orientation)		100 %		100 %	0	3	0	10	0	1

Funcionalitat : Lògica d'enfonsament (Dany Crític)

Localització :

- Arxiu: Ship.java
- Classe: Ship
- Mètode: isSunk()

Test :

- Arxiu: ShipTest.java
- Classe: ShipTest
- Mètode: testIsSunkDataDriven(ShipType type, int hits, boolean expectedSunk)
- **Tipus de test:** Caixa Negra / Taula de Decisió
- **Tècniques utilitzades:**
  - **Boundary Value Analysis:** Es verifiquen les condicions de frontera per a l'enfonsament.
  - Cas 1: Hits < Longitud -> False.
  - Cas 2: Hits == Longitud -> True.
  - Es proven diferents tipus de vaixells per assegurar que la lògica s'adapta a la longitud variable.

```
111.    /**
112.     * Checks if the ship is sunk.
113.     * A ship is sunk if the number of hits equals or exceeds its length.
114.     *
115.     * @return true if the ship is sunk; false otherwise.
116.     */
117.    public boolean isSunk() {
118.        return hitCount >= getLength();
119.    }
```

isSunk()		100 %		100 %	0	2	0	1	0	1

Funcionalitat : Validació de col·locació (Solapament)

Localització :

- Arxiu: Board.java
- Classe: Board
- Mètode: isValidPlacement(Ship ship, Coordinate start, Orientation orientation)

Test :

- Arxiu: BoardTest.java
- Classe: BoardTest
- Mètode: testInvalidPlacementOverlappingShips()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Path Coverage (Camí d'error):** Es força l'execució del camí on cells[x][y].hasShip() retorna true.
  - Es verifica que l'algoritme detecta la col·lisió i retorna false immediatament sense modificar l'estat del tauler.

```
157     /**
158      * Test Case: Attempt to place a ship on top of another.
159      * * Type: Black Box Testing
160      * * Technique: Equivalence Partitioning - Invalid Class (Occupied Space).
161      * * Description: Verifies that the system detects a collision when trying to place a
162      * ship in coordinates that are already occupied by a previously placed ship.
163      * The second placement must be rejected.
164      */
165     @Test
166     public void testInvalidPlacementOverlappingShips() {
167         Ship ship1 = new Ship(ShipType.CRUISER);
168         Ship ship2 = new Ship(ShipType.SUBMARINE);
169 
170         board.placeShip(ship1, new Coordinate(x: 3, y: 3), Orientation.HORIZONTAL);
171         assertFalse(board.isValidPlacement(ship2, new Coordinate(x: 3, y: 3), Orientation.VERTICAL));
172     }
```

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

Funcionalitat : Validació de col·locació (Fora de rang complex)

Localització :

- Arxiu: Board.java
- Classe: Board
- Mètode: isValidPlacement(...)

Test :

- Arxiu: BoardTest.java
- Classe: BoardTest
- Mètode: testInvalidPlacementHorizontalOutInLoop()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Loop Testing & Condition Coverage:** Es prova el cas específic on la coordenada inicial és vàlida (8,0), però el bucle for que itera sobre la longitud del vaixell (CARRIER, len=5) calcula una coordenada final (12,0) que surt del tauler.
  - Això verifica la condició de sortida prematura del bucle de validació.

```
129     /**
130      * Test Case: Attempt placement where ship extends beyond board limits.
131      * * Type: Black Box Testing
132      * * Technique: Boundary Value Analysis (Compound Boundary).
133      * * Description: Verifies that even if the starting coordinate (8,0) is valid, the system
134      * correctly calculates that a ship of length 5 (CARRIER) would extend to index 12, which
135      * is out of bounds. The placement must be rejected.
136      */
137     @Test
138     public void testInvalidPlacementHorizontalOutInLoop() {
139         Ship ship = new Ship(ShipType.CARRIER);
140         assertFalse(board.isValidPlacement(ship, new Coordinate(x: 8, y: 0), Orientation.HORIZONTAL));
141     }
```

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

Funcionalitat : Execució d'atacs al Tauler (Hit/Miss)

Localització :

- Arxiu: Board.java
- Classe: Board
- Mètode: processAttack(Coordinate coordinate)

Test :

- Arxiu: BoardTest.java
- Classe: BoardTest
- Mètode: testProcessAttackHit()
- **Tipus de test:** Caixa Negra
- **Tècniques utilitzades:**
  - **Equivalence Partitioning:** Es verifica que un atac a una coordenada ocupada retorna l'enum AttackResult.HIT i actualitza l'estat intern del tauler.

```
145.    /**
146.     * Processes an attack on a specific coordinate.
147.     * Handles logic for hits, misses, sinking ships, and repeated attacks.
148.     *
149.     * @param coordinate The target coordinate of the attack.
150.     * @return The result of the attack (HIT, MISS, SUNK, ALREADY_ATTACKED).
151.     * @throws IllegalArgumentException if the coordinate is invalid.
152.     */
153.    public AttackResult processAttack(Coordinate coordinate) {
154.        if (!coordinate.isValid(size)) {
155.            throw new IllegalArgumentException("Invalid coordinate: " + coordinate);
156.        }
157.
158.        Cell cell = getCell(coordinate);
159.
160.        if (cell.isAlreadyAttacked()) {
161.            return AttackResult.ALREADY_ATTACKED;
162.        }
163.
164.        boolean hit = cell.attack();
165.
166.        if (hit) {
167.            Ship ship = cell.getShip();
168.            if (ship.isSunk()) {
169.                return AttackResult.SUNK;
170.            }
171.            return AttackResult.HIT;
172.        } else {
173.            return AttackResult.MISS;
174.        }
175.    }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processAttack(Coordinate)		100 %		100 %	0	5	0	12	0	1

Funcionalitat : Gestió d'atacs duplicats

Localització :

- Arxiu: Board.java
- Classe: Board
- Mètode: processAttack(Coordinate coordinate)

Test :

- Arxiu: BoardTest.java
- Classe: BoardTest
- Mètode: testProcessAttackAlreadyAttacked()
- **Tipus de test:** Caixa Blanca
- **Tècniques utilitzades:**
  - **Path Coverage:** Es verifica el camí condicional on cell.isAlreadyAttacked() és cert.
  - Es valida que el sistema retorna AttackResult.ALREADY_ATTACKED i no modifica l'estat de joc (idempotència).

```
363     /**
364      * Test Case: Verify attack logic for duplicate attacks.
365      * * Type: Black Box Testing
366      * * Technique: Equivalence Partitioning - Invalid History.
367      * * Description: Verifies that attacking the same coordinate twice results in a specific
368      * 'ALREADY_ATTACKED' status, preventing game logic errors.
369      */
370     @Test
371     public void testProcessAttackAlreadyAttacked() {
372         board.processAttack(new Coordinate(x: 5, y: 5));
373         AttackResult result = board.processAttack(new Coordinate(x: 5, y: 5));
374 
375         assertEquals(AttackResult.ALREADY_ATTACKED, result);
376     }
```

Funcionalitat : Setup del joc (Ordinador)

Localització :

- Arxiu: Game.java
- Classe: Game
- Mètode: placeComputerShipsRandomly()

Test :

- Arxiu: GameTest.java
- Classe: GameTest
- Mètode: testPlaceComputerShipsRandomly()
- **Tipus de test:** Caixa Negra / Comportament
- **Tècniques utilitzades:**
  - **State Verification:** Es verifica que, després de cridar al mètode, el tauler de l'ordinador conté exactament 5 vaixells.

- Es valida que els vaixells col·locats corresponen als tipus requerits (Carrier, Battleship, etc.).

```
182.    /**
183.     * Automates the placement of ships for the computer player.
184.     * Iterates through all available ship types and attempts to place them at
185.     * random coordinates
186.     * and orientations until a valid placement is found for each.
187.     */
188.    public void placeComputerShipsRandomly() {
189.        ShipType[] types = ShipType.values();
190.
191.        for (ShipType type : types) {
192.            Ship ship = new Ship(type);
193.            boolean placed = false;
194.
195.            // Keep trying random positions until the ship fits
196.            while (!placed) {
197.                int x = random.nextInt(10);
198.                int y = random.nextInt(10);
199.                Orientation orientation = random.nextBoolean()
200.                        ? Orientation.HORIZONTAL
201.                        : Orientation.VERTICAL;
202.
203.                Coordinate start = new Coordinate(x, y);
204.                // The board handles collision detection and boundaries
205.                placed = computerBoard.placeShip(ship, start, orientation);
206.            }
207.        }
208.    }
209. }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
placeComputerShipsRandomly()		100 %		100 %	0	4	0	14	0	1

Funcionalitat : Setup del joc (Jugador)

Localització :

- Arxiu: BoardController.java
- Classe: BoardController
- Mètode: setupPlayerShips(Board board)

Test :

- Arxiu: BoardControllerTest.java
- Classe: BoardControllerTest
- Mètode: testSetupPlayerShips()
- **Tipus de test:** Integració amb Mocks
- **Tècniques utilitzades:**
  - **Mocking (Interaction Verification):** Es verifica la interacció entre el Controlador i la Vista.
  - Comprovem que es crida al mètode placeShip exactament 5 vegades (times(5)), simulant que l'usuari introdueix coordenades vàlides a través de la Vista.

```
31.     /**
32.      * Orchestrates the setup phase where the player places all their ships on the
33.      * board.
34.      * Iterates through all available ship types and prompts the user for
35.      * coordinates and orientation.
36.      * Validates the placement and provides feedback (success or failure) to the
37.      * user.
38.      *
39.      * @param board The board where the ships will be placed.
40.      */
41.     public void setupPlayerShips(Board board) {
42.         ShipType[] shipTypes = ShipType.values();
43.
44.         for (ShipType type : shipTypes) {
45.             boolean placed = false;
46.
47.             // Loop until the current ship is successfully placed
48.             while (!placed) {
49.                 view.displayMessage("\nCurrent board:");
50.                 view.displayBoard(board, false);
51.
52.                 view.displayMessage("\nPlace your " + type.getDisplayName()
53.                         + " (length: " + type.getLength() + ")");
54.
55.                 // Get input from the user via the view
56.                 Coordinate start = view.getCoordinateInput("Initial coordinate");
57.                 Orientation orientation = view.getOrientationInput();
58.
59.                 Ship ship = new Ship(type);
60.                 placed = attemptPlaceShip(board, ship, start, orientation);
61.
62.                 if (!placed) {
63.                     view.displayMessage("\nThe ship cannot be placed there. Try another position.");
64.                 }
65.             }
66.
67.             view.displayMessage("\n¡" + type.getDisplayName() + " successfully placed!\n");
68.         }
69.
70.         view.displayMessage("\n¡All your ships are in place!\n");
71.     }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setupPlayerShips(Board)		94 %		83 %	1	4	1	18	0	1

Funcionalitat : Torn del Jugador (Flux Correcte)

Localització :

- Arxiu: GameController.java
- Classe: GameController
- Mètode: processPlayerTurn()

Test :

- Arxiu: GameControllerTest.java
- Classe: GameControllerTest
- Mètode: testProcessPlayerTurn_HitShip()
- **Tipus de test:** Integració amb Mocks
- **Tècniques utilitzades:**
  - **Behavior Verification:** Es verifica que quan el Model retorna AttackResult.HIT, el controlador invoca view.displayAttackResult(...) i mostra el missatge d'èxit corresponent.

```
132.    /**
133.     * Handles the logic for a single turn of the computer opponent.
134.     * 1. Adds a delay for better user experience (simulating "thinking").
135.     * 2. Executes the computer's attack via the model.
136.     * 3. Checks if the attack hit/sunk a player's ship and notifies the user.
137.     */
138.    public void processComputerTurn() {
139.        view.displayMessage("\n=== COMPUTER'S TURN ===\n");
140.        view.displayMessage("The computer is attacking...\n");
141.
142.        try {
143.            Thread.sleep(1000); // Dramatic pause for UX
144.        } catch (InterruptedException e) {
145.            Thread.currentThread().interrupt();
146.        }
147.
148.        // The model determines where the computer attacks
149.        Coordinate attacked = game.processComputerAttack();
150.        view.displayMessage("Computer attacked: " + attacked);
151.
152.        // Retrieve the result of the attack to inform the player
153.        Cell cell = game.getPlayerBoard().getCell(attacked);
154.        if (cell.getState() == CellState.HIT) {
155.            view.displayMessage(">>> The computer hit one of your ships! <<<");
156.
157.            if (cell.getShip().isSunk()) {
158.                view.displayMessage(">>> Your " + cell.getShip().getType().getDisplayName()
159.                        + " has been sunk! <<<\n");
160.            }
161.        } else {
162.            view.displayMessage(">>> The computer missed! <<<\n");
163.        }
164.    }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processPlayerTurn()		100 %		100 %	0	3	0	16	0	1

Funcionalitat : Torn de l'Ordinador (IA)

Localització :

- Arxiu: Game.java
- Classe: Game
- Mètode: processComputerAttack()

Test :

- Arxiu: GameTest.java
- Classe: GameTest
- Mètode: testProcessComputerAttack()
- **Tipus de test:** Integració
- **Tècniques utilitzades:**
  - **State Transition Testing:** Es verifica que l'atac de l'ordinador canvia l'estat del joc de COMPUTER_TURN a PLAYER_TURN.
  - Es valida que la coordenada generada per la IA és vàlida (dins del tauler).

```
132.    /**
133.     * Handles the logic for a single turn of the computer opponent.
134.     * 1. Adds a delay for better user experience (simulating "thinking").
135.     * 2. Executes the computer's attack via the model.
136.     * 3. Checks if the attack hit/sunk a player's ship and notifies the user.
137.     */
138.    public void processComputerTurn() {
139.        view.displayMessage("\n=== COMPUTER'S TURN ===\n");
140.        view.displayMessage("The computer is attacking...\n");
141.
142.        try {
143.            Thread.sleep(1000); // Dramatic pause for UX
144.        } catch (InterruptedException e) {
145.            Thread.currentThread().interrupt();
146.        }
147.
148.        // The model determines where the computer attacks
149.        Coordinate attacked = game.processComputerAttack();
150.        view.displayMessage("Computer attacked: " + attacked);
151.
152.        // Retrieve the result of the attack to inform the player
153.        Cell cell = game.getPlayerBoard().getCell(attacked);
154.        if (cell.getState() == CellState.HIT) {
155.            view.displayMessage(">>> The computer hit one of your ships! <<<");
156.
157.            if (cell.getShip().isSunk()) {
158.                view.displayMessage(">>> Your " + cell.getShip().getType().getDisplayName()
159.                        + " has been sunk! <<<\n");
160.            }
161.        } else {
162.            view.displayMessage(">>> The computer missed! <<<\n");
163.        }
164.    }
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
processComputerTurn()		94 %		75 %	1	3	2	15	0	1

Funcionalitat : Condició de Victòria (Game Over)

Localització :

- Arxiu: Game.java
- Classe: Game
- Mètode: isGameOver()

Test :

- Arxiu: GameTest.java
- Classe: GameTest
- Mètode: testIsGameOverPlayerWins()
- **Tipus de test:** Integració / Escenari
- **Tècniques utilitzades:**
  - **Scenario Testing:** Es simula un final de partida on s'enfonsa l'últim vaixell enemic.
  - Es verifica que isGameOver() retorna true i que l'estat del joc s'actualitza correctament a PLAYER_WON.

```
149.    /**
150.     * Checks if the game has reached a terminal state (win/loss).
151.     * A game is over if all ships on either board are sunk.
152.     *
153.     * @return true if the game is over; false otherwise.
154.     */
155.    public boolean isGameOver() {
156.        if (computerBoard.allShipsSunk()) {
157.            return true;
158.        }
159.
160.        if (playerBoard.allShipsSunk()) {
161.            return true;
162.        }
163.
164.        return false;
165.    }
```

isGameOver()		100 %		100 %	0	3	0	5	0	1