

**Funcionalitat:** Comprova si el vaixell ha estat colpejat a les coordenades indicades.

**Localització:** Ship.java, public class Ship, public boolean isHit(final int x, final int y )

**Test:** ShipTest.java, ShipTest

- Test de caixa negra (partició equivalent, limit i frontera):
  - o void TestisHitLimitInterior()
  - o void TestisHitLimitExterior()
  - o void TestisHitOcupada()
  - o void TestisHitNoOcupada()
- Test de caixa blanca (Statement Coverage):
  - o void TestIsHitStatementCoverage()

```
9 usages  NIU01558589
46 public boolean isHit(final int x, final int y) {
47     assert (x >= 0 && x <= 10 && y >= 0 && y <= 10) :
48         "Les coordenades han de ser dins dels límits";
49     boolean result = false;
50     for (Cell cell : posicionesShip) {
51         assert (cell.getX() >= 0 && cell.getX() <= 10
52             && cell.getY() >= 0 && cell.getY() <= 10) :
53             "Les coordenades de la cel·la han de ser dins dels límits";
54         if (cell.getX() == x && cell.getY() == y) {
55             result = true;
56             break;
57         }
58     }
59     assert (result == true && x >= 0 && x <= 10 && y >= 0 && y <= 10) ||
60         (result == false && !(x >= 0 && x <= 10 && y >= 0 && y <= 10)) :
61         "El resultat ha de ser coherent amb si la cel·la està ocupada o no";
62     return result;
63 }
```

Molts dels testos cobreixen casos similars, però de totes maneres vam decidir fer testos per cada part demanada als criteris de correcció per clarificar-ho tot.

---

**Funcionalitat:** Marca el vaixell com a colpejat a les coordenades indicades.

**Localització:** Ship.java, class Ship, public void markHit(final int x, final int y)

**Test:**

- Test de caixa negra (partició equivalent, limit i frontera):
  - o void markHitLimitInterior()
  - o void markHitLimitExterior()
  - o void markHitGolpejat()

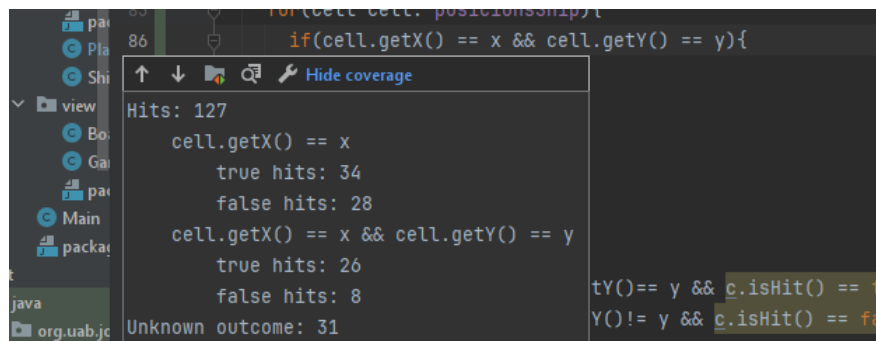
- void markHitLimitNoGolpejat()
- Test de caixa blanca:
  - Statement Coverage:
    - void markHitStatementCoverage()

```

71 public void markHit(final int x, final int y) {
72     // Implement logic to mark the ship as hit at the specified coordinates
73
74     //Precondicions
75     assert posicionsShip != null:
76         "La llista del vaixell no pot ser null";
77     assert (x >= 0 && x < 10):
78         "La x ha d'estar dins del tauler";
79
80     assert (y >= 0 && y < 10):
81         "La y ha d'estar dins del tauler";
82
83     Cell c = new Cell(x, y, 0);
84     //Marcar cel·la com tocada
85     for(Cell cell: posicionsShip){
86         if(cell.getX() == x && cell.getY() == y){
87             cell.hit();
88             c = cell;
89             break;
90         }
91     }
92
93     //Postcondicions
94     assert (c.getX() == x && c.getY() == y && c.isHit() == true
95         || c.getX() != x && c.getY() != y && c.isHit() == false );

```

- Condition Coverage:
  - void markHitConditionCoverage()



```

86 if(cell.getX() == x && cell.getY() == y){
    Hits: 127
    cell.getX() == x
    true hits: 34
    false hits: 28
    cell.getX() == x && cell.getY() == y
    true hits: 26
    false hits: 8
    Unknown outcome: 31

```

- Decision Coverage:
  - void markHitDecisionCoverage()

```

1 public void markHit(final int x, final int y) {
2     // Implement logic to mark the ship as hit at the specified coordin
3
4     //Precondicions
5     assert posicionsShip != null:
6         "La llista del vaixell no pot ser null";
7     assert (x >= 0 && x < 10):
8         "La x ha d'estar dins del tauler";
9
10    assert (y >= 0 && y < 10):
11        "La y ha d'estar dins del tauler";
12
13    Cell c = new Cell(x, y);
14    //Marcar cel·la com tocada
15    for(Cell cell: posicionsShip){
16        if(cell.getX() == x && cell.getY() == y){
17            cell.hit();
18            c = cell;
19            break;
20        }
21    }

```

Comentari: a IntelliJ, les condicions queden marcades de color verd si es compleixen en tots els seus casos, i groc si es cobreixen parcialment.

---

**Funcionalitat:** Comprova si el vaixell ha estat enfonsat

**Localització:** Ship.java, class Ship, public boolean isSunk()

**Test:** ShipTest.java, ShipTest

- Caixa negra (partició equivalent, limit i frontera)
  - void isSunkLimitInterior()
  - void isSunkLimitExterior()
- Caixa blanca:
- Statement Coverage:
  - void isSunkStatementCoverage()

```

104 public boolean isSunk() {
105     //Hem de determinar si totes les Cells del vaixell han estat tocadetes
106
107     Boolean isHit = true;
108     //Precondicions
109     assert posicionsShip.isEmpty()==false:
110         "La llista de cel·les no pot estar buida";
111     assert posicionsShip.size() > 10:
112         "Un vaixell no pot ser més gran que el tauler";
113     if(posicionsShip.isEmpty()){
114         return false;
115     }
116     for(Cell cell: posicionsShip){
117
118         if(cell.getX() < 0 || cell.getX() > 10
119             || cell.getY() < 0 || cell.getY() > 10){
120             return false;
121         }
122
123         //Si una cell del vaixell no s'ha tocat no està enfonsat
124         if (cell.isHit() == false){
125             isHit = false;
126             return false;
127             //No està enfonsat
128         }
129     }
130     //Postcondicions
131     assert isHit == true:
132         "Vaixell enfonsat, totes les cel·les golpejades";
133     assert isHit == false:
134         "Vaixell no enfonsat, no totes les cel·les golpejades";
135     return true;
136     //Està enfonsat
137 }

```

- Decision Coverage:
  - void isSunkDecisionCoverage() (no se com justificar-ho amb IntelliJ, pero fent ús del debug entra als dos casos per l'if, tant treu com false)
- Condition Coverage:
  - void isSunkConditionCoverage()

```

    if(cell.getX() < 0 || cell.getX() > 10
        || cell.getY() < 0 || cell.getY() > 10){
        return false;
    }

```

**Funcionalitat:** situar un vaixell al tauler

**Localització:** Board.java, class Board, public boolean placeShip(Ship ship, int x, int y, boolean isHorizontal)

**Test:** BoardTest, class BoardTest

- Caixa negra (partició equivalent, limit i frontera)
  - void placeShipValidSituation()
  - void placeShipInvalidOutOfBoard()

- void placeShipCellOccupiedBoard()
- void placeShipVerticalBoard()
- void placeShipLimitBoard()
- void placeEmptyShipBoard()
- Caixa blanca
  - Statement Coverage
    - void placeShipBoundHorizontalBoard()
    - void placeShipBoundVerticalBoard()
    - void placeShipOccupiedCellsBoard()
    - void placeShipSuccessBoard()
    - void placeShipDecisionFirstCaseBoard()
    - void placeShipDecisionSecondCaseBoard()
  - Decision Coverage
    - void placeShipDecisionFirstCaseBoard()
    - void placeShipDecisionSecondCaseBoard()
  - Condition Coverage
    - void placeShipConditionFirstCaseBoard()
    - void placeShipConditionSecondCaseBoard()
    - void placeShipConditionThirdCaseBoard()
    - void placeShipConditionFourthCaseBoard()
  - Loop testing
    - void placeShipFirstCell()
    - void placeShipSecondCell()
    - void placeShipPenultimateCell()
    - void placeShipLastCell()

```

46 @ public boolean placeShip(Ship ship, int x,
47     int y, boolean isHorizontal) {
48     //Precondicions
49     assert ship != null : "El vaixell no pot ser null";
50     assert ship.getTamany() > 0 : "Tamany del vaixell positiu";
51     assert x >= 0 && y >= 0 : "Coordenades dins de rang";
52
53     int shipSize = ship.getTamany();
54     // Validar que el vaixell no surti dels limits
55     if((isHorizontal && (y + shipSize > board[0].length))
56         || (!isHorizontal && (x+shipSize > board.length))) {
57         return false;
58     }
59
60     // Validar que les cel·les no estiguin ocupades
61     List<Cell> llistaPosShip = ship.getPosicionsShip();
62     for (int i = 0; i < shipSize; i++) {
63         Cell cell = llistaPosShip.get(i);
64         if(cell.getX() >= board.length
65             || cell.getY() >= board[0].length
66             || board[cell.getX()][cell.getY()] == 1) {
67             return false;
68         }
69     }
70     // Afegir el vaixell
71     for (int i = 0; i < shipSize; i++) {
72         Cell cell = llistaPosShip.get(i);
73         board[cell.getX()][cell.getY()] = 1;
74     }
75
76     // Afegir vaixell a la llista de vaixells
77     ships.add(ship);
78
79     assert ships.contains(ship) : "El vaixell no ha estat afegir correctament";
80
81     // Cas existós
82     return true;
83 }

```

**Funcionalitat:** Registra un cop a una situació específica del tauler

**Localització:** Board.java, class Board, public boolean takeShot(int x, int y)

**Test:** BoardTest, class BoardTest

- Caixa negra (partició equivalent, limit i frontera)
  - void takeShotLimitInterior()
  - void takeShotLimitExterior()
  - void takeShotJaColpejada()
  - void takeShotCellAigua()
  - void takeShotCellVaixell()
- Caixa Blanca
  - Statement Coverage
    - void takeShotStatementCoverage()
  - Path Coverage
    - void takeShotPathCoverage()
  - Loop Simple
    - void takeShotSimpleLoop()

```

94 public boolean takeShot(int x, int y) {
95
96     //Precondicions
97     assert board != null:
98         "El tauler no pot ser null";
99     assert board.length == 10 && board[0].length == 10:
100         "El tauler ha de mesurar 10";
101     assert x >= 0 && x <= 9:
102         "Les coordenades han d'estar dins del rang";
103     assert y >= 0 && y <= 9:
104         "Les coordenades han d'estar dins del rang";
105     assert ships != null:
106         "La llista de vaixes no pot estar buida";
107
108     //Validar les coordenades
109     if (x < 0 || x > 9 || y < 0 || y > 9){
110         return false;
111     }
112
113     boolean res = false;
114     //Cell ja golpejada?
115
116     if (board[x][y] == -1){
117         res = false;
118     }
119
120     //Si no ha estat golpejada
121     //Marcar com golpejada
122     //Es golpeja un vaixell?
123     if(board[x][y] == 1){
124         //Marcar el ship com a golpejat
125         board[x][y] = -1;
126
127         for(Ship ship: ships){
128             if(ship.isHit(x,y)){
129                 //Golpejem el vaixell
130                 ship.markHit(x,y);
131                 System.out.println("Vaixell tocat");
132                 res = true;
133             }
134         }
135     } else {
136
137         //Si no es golpeja un vaixell --> aigua
138         //Aigua = -1
139         board[x][y] = -1;
140         System.out.println("Aigua");
141         res = false;
142     }
143
144     assert board != null:
145         "El tauler no pot ser null";
146     return res;
147
148 }
149

```

**Funcionalitat:** Comprova si tots els vaixells s'han enfonsat

**Localització:** Board.java, class Board, public boolean isAllShipsSunk()

**Test:** BoardTest.java, class BoardTest

- Caixa negra (partició equivalent, limit i frontera)
  - void isAllShipsSunkLimitInterior()
  - void isAllShipsSunkLimitExterior()
  - void isAllShipsSunkShipsSenseMesura()
- Caixa blanca

```
/**
 * Checks if all ships are sunk.
 *
 * @return true if all ships are sunk, false otherwise
 */
5 usages clauu113 +1
public boolean isAllShipsSunk() {

    //Precondicions
    assert ships != null:
        "No pot no haver-hi vaixells";
    if(ships.isEmpty()){
        return false;
    }
    for(Ship ship: ships){
        //Si algun no està enfonsat
        if(!ship.isSunk()){
            return false;
        }
    }
    return true;
}
```

---

**Funcionalitat:** Valida que hem situat correctament el vaixell al tauler

**Localització:** Board.java, class Board, public boolean isValidPlacement(Ship ship, int x, int y, boolean isHorizontal)

**Test:** BoardTest.java, class BoardTest

- Caixa negra (partició equivalent, limit i frontera)
  - void isValidPlacementLimitInterior()
  - void isValidPlacementLimitExterior()
  - void isValidPlacementNullShip()
  - void isValidPlacementCellJaOcupadaVaixell()
  - void isValidPlacementCoordenadesValides()
- Caixa Blanca



```

82 @      public boolean isValidPlacement(Ship ship, int x,
83                                     int y, boolean isHorizontal) {
84
85         //Precondicions
86         assert ship != null:
87             "El vaixell no pot ser null";
88         assert board.length == 10 && board[0].length == 10:
89             "El tauler ha de mesurar 10";
90         assert x >= 0 && x <= 9:
91             "Les coordenades han d'estar dins del rang";
92         assert y >= 0 && y <= 9:
93             "Les coordenades han d'estar dins del rang";
94         assert ship.getTamany() <= 0:
95             "El tamany del vaixell ha de ser més gran que 0";
96
97         //El vaixell no es null
98         if(ship == null){
99             return false;
100        }
101
102        //verificar coordenades
103        if (x < 0 || x > 9 || y < 0 || y > 9){
104            return false;
105        }
106        //Obtenir tamany vaixell
107        int shipSize = ship.getTamany();
108
109        int rowLimit = 0;
110        int colLimit = 0;
111        if(isHorizontal){
112            rowLimit = shipSize;
113            colLimit = 1;
114        }else{
115
116            rowLimit = 1;
117
118            colLimit = shipSize;
119        }
120
121        //verificar cells lliures
122        for (int row = x; row < x + rowLimit; row++){
123            for(int col = y; col < y + colLimit; col++){
124
125                if (row < 0 || row >= board.length || col < 0 || col >= board[0].length) {
126                    return false;
127                }
128
129                if(board[row][col] != 0){
130                    return false;
131                }
132            }
133        }
134
135        return true;
136    }
137
138 }
139

```

En aquest cas, el statement coverage queda cobert amb el codi de les funcions de tests planejades per la caixa negra.

- Loop Test
  - Loop aniuat
    - void isValidPlacementInnerLoop()
    - void isValidPlacementOuterLoop()
    - void isValidPlacementLimitExterior2()

Nota: Tant la classe Player com la classe Cell disposen de metodes que no requereixen de testeig, ja que el seu funcionament es limita a retornar valors de funcions d'altres classes o la funcio de getters o setters.

Nota2: Dins de totes les classes especificades s'ha aplicat el principi de Design By Contract amb asserts.

**Funcionalitat:** Situació inicial dels vaixells dels jugadors

**Localització:** GameController.java, class GameController, public void placeShips(Player player)

**Test:** GameControllerTest.java, class GameControllerTest

- Caixa Blanca:
  - Statement Coverage
  - LoopAnuat
  - Path Coverage

```

29 @ 3 usages → KNU01358589
30 public void placeShips(Player player) {
31     // Precondició: El jugador no ha de ser nul
32     assert player != null : "Player cannot be null.";
33
34     Scanner scanner = new Scanner(System.in);
35     System.out.println(player.getName() + ", place your ships:");
36
37     // Invariante: El jugador ha de tenir un tauler adequat per col·locar els vaixells
38     assert player.hasBoard() : "Player does not have a valid board to place ships.";
39
40     for (int i = 1; i <= 5; i++) { // En aquest cas, posarem 5 vaixells amb longituds 1,2,3,4,5
41         System.out.println("Placing ship of size " + i);
42
43         boolean placed = false;
44         while (!placed) {
45             System.out.print("Enter x, y coordinates and orientation (h/v): ");
46             int x = scanner.nextInt();
47             int y = scanner.nextInt();
48             char orientation = scanner.next().charAt(0);
49             boolean isHorizontal = (orientation == 'h');
50
51             Ship ship = new Ship(BoardUtils.generateCells(x, y, i, isHorizontal), i);
52             if (player.placeShips(ship, x, y, isHorizontal)) {
53                 System.out.println("Ship placed successfully!");
54                 // Postcondició: El vaixell ha estat col·locat correctament
55                 placed = true;
56             } else {
57                 System.out.println("Invalid placement. Try again.");
58             }
59         }
60     }
61 }
62

```

- Mock Object

- Mockito i Mock Objects dins dels metodes de PathCoverage, StatementCoverage i LoopCoverage.

---

**Funcionalitat:** Desenvolupament del CI

**Localització:** pom.xml, .github/workflows/maven.yml, checkstyle.xml

Per la configuració de les CI, hem fet ús de GitHub Actions.

El primer que vam fer va ser afegir al nostre pom.xml del projecte els plugins de allo que volíem comprovar cada vegada que es fes un merge. En el nostre cas vam afegir Checkstyle, amb un arxiu de configuració extern per només comprovar que no es passés de les 120 línies

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC
    "-//Checkstyle//DTD Checkstyle Configuration 1.3//EN"
    "https://checkstyle.org/dtds/configuration_1_3.dtd">

<module name="Checker">
    <!-- TreeWalker module is the main module that runs all checks -->
    <module name="TreeWalker">
        <module name="LineLength">
            <property name="max" value="120"/>
        </module>
    </module>
</module>
```

I el maven surefire per tal de poder aplicar els testos de JUnit.

Després d'això, vam recarregar el pom.xml, i vam anar a GitHub. Allà, a l'apartat de Actions, vam escollir una plantilla per **Java CI with Maven**, sobre la que vam afegir els nostres canvis per executar el checkstyle i que es fes una build segons l'especificat al pom.xml.

```

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 17
      uses: actions/setup-java@v4
      with:
        java-version: '17'
        distribution: 'temurin'
        cache: maven
    - name: Build with Maven
      run: mvn -B package --file pom.xml

    - name: Install dependencies and run tests
      run: mvn clean install
      env:
        CI: true

    - name: Check code style with Checkstyle
      run: mvn checkstyle:check

```

Després vam anar a l'apartat de Settings -> Branches -> Add Rule

### Protect matching branches

☐ **Require a pull request before merging**  
 When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ **Require status checks to pass before merging**  
 Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

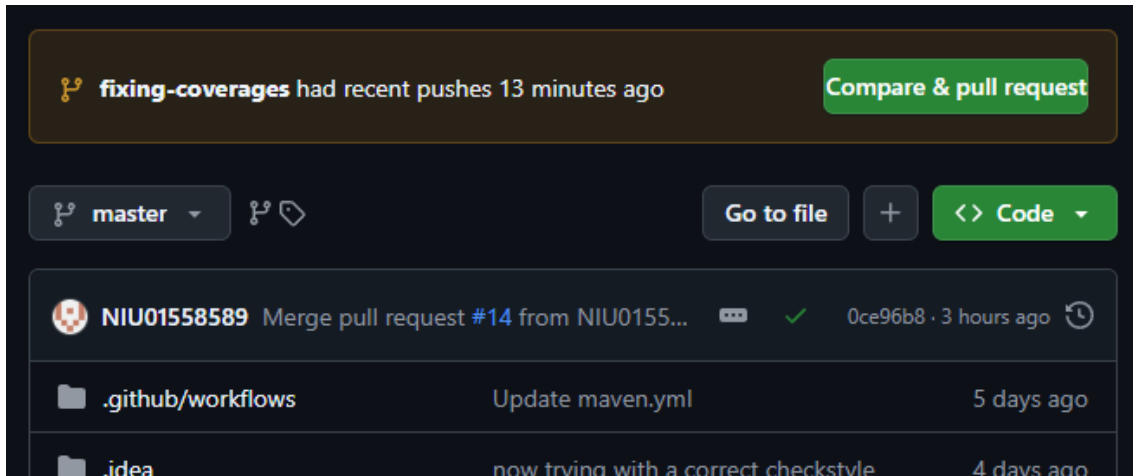
☐ **Require branches to be up to date before merging**  
 This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

#### Status checks that are required

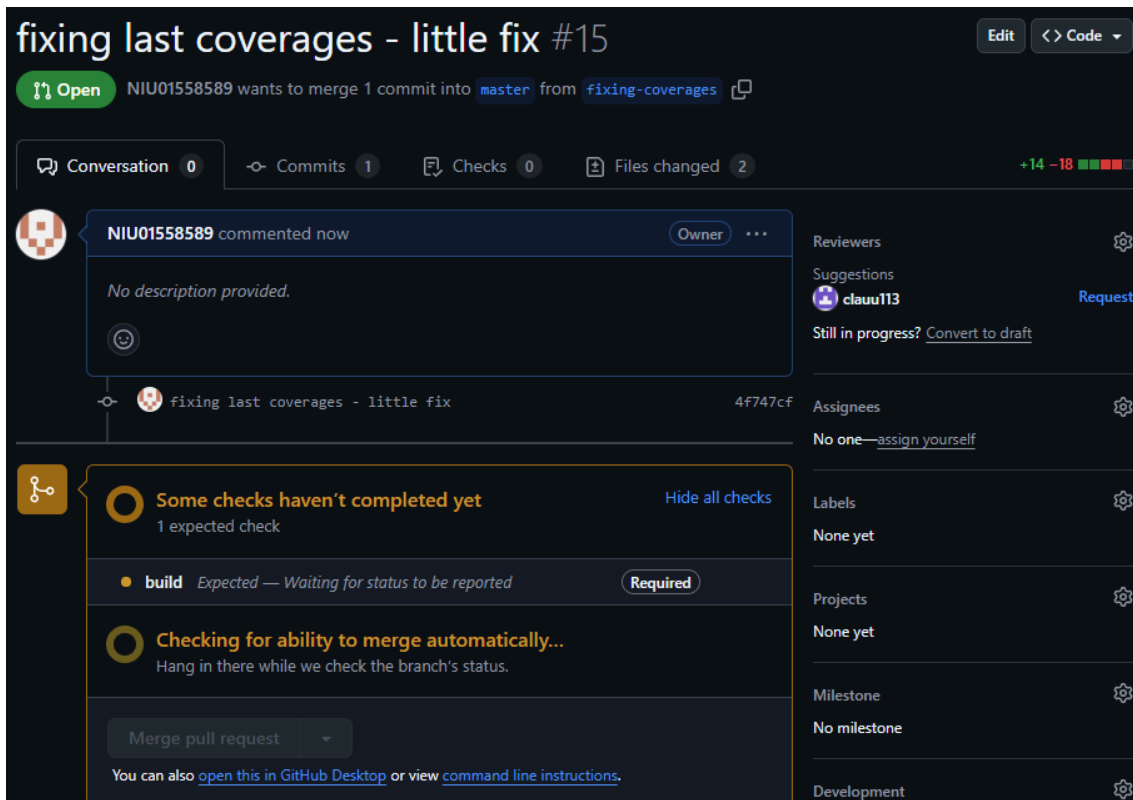
build	GitHub Actions <span>▼</span> <span>✕</span>
-------	--

Una vegada fet això, expliquem com fem els merge requests

Primer de tot, des de IntelliJ pujem la branca amb un **git push origin nom-de-la-branca**.  
Llavors en anem a Github, on posa **merge & pull requests**



Confirmem la pull request i es llavors quan s'executen els testos i el checkstyle automaticament:



En cas negatiu, el botó es bloqueja i no es permet el merge

test 2 about ci #3

[Open](#) NIU01558589 wants to merge 1 commit into `master` from `feature-checking-ci`

Conversation 0 Commits 1 Checks 0 Files changed 1

NIU01558589 commented now

No description provided.

test 2 about ci

**All checks have failed** 1 failing check

Java CI with Maven / build (pull\_request) Failing after 10s

**Required statuses must pass before merging**

All required [statuses](#) and check runs on this pull request must run successfully to enable automatic merging.

☐ Merge without waiting for requirements to be met (bypass branch protections)

[Merge pull request](#)

Reviewers

No reviews

Still in progress? [Convert to draft](#)

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues

En cas positiu, es permet el merge:

fixing last coverages - little fix #15

[Open](#) NIU01558589 wants to merge 1 commit into `master` from `fixing-coverages`

Conversation 0 Commits 1 Checks 0 Files changed 2

NIU01558589 commented 1 minute ago

No description provided.

fixing last coverages - little fix

**All checks have passed** 1 successful check

**This branch has no conflicts with the base branch**



Merging can be performed automatically.


[Merge pull request](#)


You can also [open this in GitHub Desktop](#) or view [command line instructions](#).


Confirmem el merge amb el boto **merge pull request**.


# fixing last coverages - little fix #15

 Merged NIU01558589 merged 1 commit into `master` from `fixing-coverages`  now

 Conversation 0

 Commits 1

 Checks 0

 Files changed 2





NIU01558589 commented 2 minutes ago

Owner ...


No description provided.



  fixing last coverages - little fix

✓ 4f747cf



 NIU01558589 merged commit 99a19cc into `master`


View details

Revert

now

1 check passed



 NIU01558589 deleted the `fixing-coverages` branch now

Restore branch