Eric Chaves Sánchez - 1633944
Pau Leyva García - 1636526
Dimecres 10:30
Battleship

Informe d'Implementació del Joc "Battleship"

Aplicació de Tècniques de Testing i Arquitectura Model-Vista-Controlador (MVC)

Test i Qualitat del Software
02/12/2024

Eric Chaves Sánchez - 1633944
Pau Leyva García - 1636526
Dimecres 10:30
Battleship

Contingut

Eric Chaves Sánchez - 1633944
Pau Leyva García - 1636526
Dimecres 10:30
Battleship

Descripció del projecte

Aquest informe descriu el desenvolupament i implementació del joc Batalla Naval, utilitzant l'arquitectura Model-Vista-Controlador (MVC). Al llarg del projecte, s'han aplicat tècniques de testing per garantir la qualitat i fiabilitat del sistema. S'expliquen les proves realitzades sobre cada una de les funcionalitats, cobrant especial atenció a les proves de caixa negra i de caixa blanca, així com a la cobertura de sentències i camins, i l'aplicació de proves unitàries per validar la correcta implementació del joc.

Informe de desenvolupament i test per a la classe Barco

Funcionalitat: Creació de la Classe Barco

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Barco
  - **Classe:** Barco
  - **Mètodes:**
    - Barco(int longitud, boolean horizontal)
    - getLongitud()
    - esHorizontal()
    - registrarImpacto(int posicion)
    - estaHundido()
- **Comentaris:**
  - La funcionalitat de creació de vaixells segueix sent robusta i correcta. Es manté la validació per assegurar que la longitud sigui positiva, evitant així errors en la creació de vaixells amb longituds no vàlides (com longitud 0 o negativa).
  - Els mètodes getLongitud() i esHorizontal() continuen proporcionant informació correcta de la longitud i orientació del vaixell.
- **Test:**
  - **Arxiu de test:** uab.tqs.practica.model.TestBarco
  - **Mètodes de test:**
    - **testCrearBarcoHorizontal i testCrearBarcoVertical:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents. Verifiquen que es creen correctament vaixells amb longituds i orientacions vàlides.
    - **testLongitudInvalida:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Valors límit i fora de rang. Verifica que el constructor rebutja longituds no vàlides.
    - **testBarcoLongitudMinima:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents i valors límit. Valida que un vaixell amb la longitud mínima funcioni correctament.

Eric Chaves Sánchez - 1633944
Pau Leyva García - 1636526
Dimecres 10:30
Battleship

Funcionalitat: Registre d'impactes al vaixell

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Barco
  - **Mètode:** registrarImpacto(int posicion)
- **Comentaris:**
  - El mètode registrarImpacto() manté la validació rigorosa per verificar que les posicions són vàlides i dins del rang. Impactes fora de rang llancen una excepció (IllegalArgumentException).
  - S'ha validat el comportament en cas d'impactes duplicats o sobre vaixells ja enfonsats.
- **Test:**
  - **Arxiu de test:** uab.tqs.practica.model.TestBarco
  - **Mètodes de test:**
    - **testImpactoFueraDeRangoNegativo i testImpactoFueraDeRangoPositivo:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Valors límit fora de rang. Verifiquen que no es permetin impactes en posicions no vàlides.
    - **testImpactoEnPosicionesValidas:**
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Cobertura d'instruccions. Valida que es registrin impactes correctament en posicions vàlides.
    - **testImpactosAlternados:**
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Cobertura de camins. Valida el comportament amb impactes alternats.
    - **testRegistrarImpactoDuplicado:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents. Confirma que impactes duplicats no alteren l'estat del vaixell.
    - **testRegistrarImpactoOnSunkShip:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents. Valida que el vaixell es manté enfonsat independentment de nous impactes.

Funcionalitat: Determinació de l'estat del vaixell (Hundit o No)

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Barco
  - **Mètode:** estaHundido()
- **Comentaris:**
  - El mètode estaHundido() segueix funcionant de manera correcta. Retorna true només si totes les posicions han rebut un impacte.
  - Es comprova que el mètode no modifica l'estat intern del vaixell.

Eric Chaves Sánchez - 1633944
Pau Leyva García - 1636526
Dimecres 10:30
Battleship

- **Test:**
  - **Arxiu de test:** uab.tqs.practica.model.TestBarco
  - **Mètodes de test:**
    - **testBarcoNoHundidoInicialmente:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents. Verifica que un vaixell nou no està inicialment enfonsat.
    - **testHundirBarcoCompletamente:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents. Valida que el vaixell es considera enfonsat quan totes les posicions han estat impactades.
    - **testEstadoNoModificadoPorEstaHundido:**
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Verificació d'efectes col·laterals. Assegura que estaHundido() no altera l'estat intern.

Funcionalitat: Cobertura de combinacions amb Pairwise Testing

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Barco
  - **Tota la funcionalitat de la classe Barco.**
- **Comentaris:**
  - S'han provat combinacions representatives de longituds i orientacions utilitzant Pairwise Testing. Això assegura que el sistema gestiona correctament les configuracions més variades (mínima i màxima longitud, orientacions horitzontals i verticals).
- **Test:**
  - **Arxiu de test:** uab.tqs.practica.model.TestBarco
  - **Mètodes de test:**
    - **testPairwiseTestingCompleto:**
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Pairwise Testing. Valida combinacions representatives de longituds i orientacions.

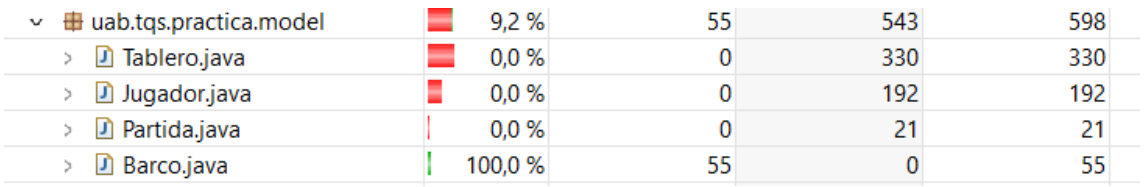
Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
practica	9,4 %	171	1.645	1.816
src/test/java	11,3 %	116	911	1.027
uab.tqs.practica.model	12,1 %	116	840	956
TestTablero.java	0,0 %	0	389	389
TestJugador.java	0,0 %	0	387	387
TestPartida.java	0,0 %	0	64	64
TestBarco.java	100,0 %	116	0	116

Eric Chaves Sánchez - 1633944

Pau Leyva García - 1636526

Dimecres 10:30

Battleship



uab.tqs.practica.model	9,2 %	55	543	598
Tablero.java	0,0 %	0	330	330
Jugador.java	0,0 %	0	192	192
Partida.java	0,0 %	0	21	21
Barco.java	100,0 %	55	0	55

Conclusió:

- **Cobertura de proves:** Tots els mètodes de la classe Barco estan adequadament coberts amb proves unitàries. S'ha validat tant el funcionament bàsic com les situacions més complexes.
- **Qualitat del codi:** El codi és robust i gestiona de manera adequada els errors comuns, com impactes fora de rang, longituds no vàlides o impactes duplicats. Els tests proporcionen confiança en la implementació i cobreixen una àmplia gamma de casos.

Informe de desenvolupament i test per a la classe Jugador

Funcionalitat: Creació de la classe Jugador per gestionar la informació del jugador, el seu tauler i la col·locació de vaixells.

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Jugador
  - **Classe:** Jugador
  - **Mètodes:**
    - Jugador(String nombre)
    - getTablero()
    - getTableroDisparo()
    - getBarcosRestantes()
    - getNombre()
    - mostrarTablero()
    - colocarBarco(int fila, int columna, int longitud, boolean horizontal)
    - recibirDisparo(int fila, int columna)
    - haPerdido()
    - actualizarTableroDisparo(int fila, int columna, String resultado)
- **Test:**
  - **Arxiu:** uab.tqs.practica.model.TestJugador
  - **Mètodes de test associats:**
    - testCreacionJugador:
      Verifica la creació del jugador, el nom, la inicialització del tauler i la quantitat de vaixells restants.
      - **Tipus de test:** Caixa negra.
      - **Tècniques:** Particions equivalents.
    - testColocarBarcoValido:
      Comprova que un vaixell es col·loqui correctament a les posicions específiques.
      - **Tipus de test:** Caixa blanca.

        - **Tècniques:** Cobertura d'instruccions.
    - testColocarBarcoInvalido:
      Verifica que no es pot col·locar un vaixell fora dels límits del tauler o en posicions incorrectes.
        - **Tipus de test:** Caixa negra.
        - **Tècniques:** Particions equivalents.
    - testRecibirDisparoAgua:
      Verifica que el disparo a una casella buida retorna "Agua".
        - **Tipus de test:** Caixa blanca.
        - **Tècniques:** Cobertura d'instruccions.
    - testRecibirDisparoTocado:
      Comprova que el disparo a una casella amb vaixell retorna "Tocado" i actualitza els vaixells restants.
        - **Tipus de test:** Caixa blanca.
        - **Tècniques:** Cobertura d'instruccions.
    - testActualizarTableroDisparo:
      Verifica que el tauler de disparos s'actualitza correctament amb "X" per tocades i "O" per aigua.
        - **Tipus de test:** Caixa blanca.
        - **Tècniques:** Cobertura de camins.
- **Comentaris:**
  Les proves cobreixen la creació de la classe Jugador, incloent la col·locació de vaixells, la resposta als disparos, i la detecció de derrota. Es valida el canvi d'estat del jugador a través de les seves interaccions amb els vaixells.

Funcionalitat: Gestió de la col·locació de vaixells del jugador.

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Jugador
  - **Mètode:** colocarBarco(int fila, int columna, int longitud, boolean horizontal)
- **Test:**
  - **Arxiu:** uab.tqs.practica.model.TestJugador
  - **Mètodes de test associats:**
    - testColocarBarcoValido:
      Comprova que un vaixell es col·loqui correctament al tauler.
        - **Tipus de test:** Caixa blanca.
        - **Tècniques:** Cobertura d'instruccions.
    - testColocarBarcoInvalido:
      Verifica que no es poden col·locar vaixells fora dels límits o en zones ocupades.
        - **Tipus de test:** Caixa negra.
        - **Tècniques:** Particions equivalents.
    - testColocarBarcosSuperpuestos:
      Comprova que no es permet col·locar vaixells superposats.

        - **Tipus de test:** Caixa negra.
        - **Tècniques:** Verificació d'errors.
- **Comentaris:**
  Aquest conjunt de proves cobreix les situacions on els vaixells es col·loquen correctament o fallen per raons de límits de tauler o superposició.

Funcionalitat: Gestió dels disparos del jugador i actualització de l'estat dels vaixells.

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Jugador
  - **Mètodes:**
    - recibirDisparo(int fila, int columna)
    - actualizarTableroDisparo(int fila, int columna, String resultado)
- **Test:**
  - **Arxiu:** uab.tqs.practica.model.TestJugador
  - **Mètodes de test associats:**
    - testRecibirDisparoAgua:
      Verifica que es rep "Agua" quan un disparo no toca cap vaixell.
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Cobertura d'instruccions.
    - testRecibirDisparoTocado:
      Comprova que un vaixell és tocat i actualitza els vaixells restants.
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Cobertura d'instruccions.
    - testActualizarTableroDisparo:
      Verifica que el tauler de disparos s'actualitza amb "X" per tocat i "O" per aigua.
      - **Tipus de test:** Caixa blanca.
      - **Tècniques:** Cobertura de camins.
- **Comentaris:**
  Es cobreixen tant els casos simples (disparar a una casella buida) com els més complexos (disparar a vaixells tocats o enfonsats).

Funcionalitat: Comprovar si el jugador ha perdut (tots els vaixells enfonsats).

- **Localització:**
  - **Arxiu:** uab.tqs.practica.model.Jugador
  - **Mètode:** haPerdido()
- **Test:**
  - **Arxiu:** uab.tqs.practica.model.TestJugador
  - **Mètodes de test associats:**
    - testHaPerdido:
      Comprova que el jugador es considera derrotat quan tots els vaixells estan enfonsats.

  - **Tipus de test:** Caixa negra.
  - **Tècniques:** Cobertura de camins.

- **Comentaris:**
  El test valida que el jugador es considera perdedor només quan tots els vaixells han estat enfonsats.

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
battleship_tqs2024	30,6 %	857	1.945	2.802
src/test/java	30,0 %	580	1.355	1.935
uab.tqs.practica.model	38,5 %	580	927	1.507
TestTablero.java	0,0 %	0	508	508
TestBarco.java	0,0 %	0	355	355
TestPartida.java	0,0 %	0	64	64
TestJugador.java	100,0 %	580	0	580
uab.tqs.practica.controller	0,0 %	0	428	428
src/main/java	31,9 %	277	590	867
uab.tqs.practica.model	42,5 %	277	375	652
Tablero.java	15,8 %	52	278	330
Barco.java	0,0 %	0	67	67
Partida.java	0,0 %	0	27	27
Jugador.java	98,7 %	225	3	228
uab.tqs.practica.controller	0,0 %	0	158	158
uab.tqs.practica.view	0,0 %	0	31	31
uab.tqs.practica	0,0 %	0	26	26

Conclusió:

Les proves cobreixen tots els aspectes de la classe Jugador, incloent la creació del jugador, la col·locació de vaixells, l'efecte dels disparos, i la detecció de derrota. Les tècniques de testing emprades, com la cobertura d'instruccions, cobertures de camins, i particions equivalents, garanteixen que es cobreixen tant els casos normals com els extrems. Així, es pot assegurar que la classe Jugador compleix les funcionalitats esperades.

Informe de desenvolupament i test per a la classe Tablero

Funcionalitat: Creació i gestió del tauler de Joc

- **Ubicació**:
  - Fitxer: uab.tqs.practica.model.Tablero
  - Classe: Tablero
  - Mètodes:
    - Tablero()
    - inicialitzarTablero()
    - colocarBarco(int fila, int columna, int longitud, boolean horizontal)
    - recibirDisparo(int fila, int columna)
    - getMatriz()
    - getTamaño()
    - imprimirTablero()
    - clonarTableroVacio()

- **Proves**:
  - Fitxer: uab.tqs.practica.model.TestTablero
  - Mètodes de prova associats:
    - testInicializacionTablero
      - **Tipus de prova**: Caixa negra.
      - **Tècnica**: Verificació de valors. Verifica que el tauler s'inicialitza amb el valor "~" per cada casella.
    - testColocarBarcoHorizontalExito
      - **Tipus de prova**: Caixa blanca.
      - **Tècnica**: Cobertura d'instruccions. Verifica que un vaixell es col·loca correctament de manera horitzontal.
    - testColocarBarcoVerticalExito
      - **Tipus de prova**: Caixa blanca.
      - **Tècnica**: Cobertura d'instruccions. Verifica que un vaixell es col·loca correctament de manera vertical.
    - testColocarBarcosSuperpuestos
      - **Tipus de prova**: Caixa negra.
      - **Tècnica**: Particions equivalents. Verifica que no es permet col·locar vaixells superposats.
    - testColocarBarcoFueraLimites
      - **Tipus de prova**: Caixa negra.
      - **Tècnica**: Verificació d'errors. Verifica que no es pot col·locar un vaixell fora dels límits del tauler.
    - testDisparoAgua
      - **Tipus de prova**: Caixa blanca.
      - **Tècnica**: Cobertura d'instruccions. Verifica que un disparo en una casella buida retorna "Aigua".
    - testDisparoTocado
      - **Tipus de prova**: Caixa blanca.
      - **Tècnica**: Cobertura d'instruccions. Verifica que un disparo que toca un vaixell retorna "Tocat".
    - testDisparoRepetido
      - **Tipus de prova**: Caixa negra.
      - **Tècnica**: Particions equivalents. Verifica que no es permet disparar dues vegades a la mateixa casella.
    - testDisparoRepetidoEnBarco
      - **Tipus de prova**: Caixa negra.
      - **Tècnica**: Verificació d'errors. Verifica que no es pot disparar a un vaixell que ja ha estat tocat.
    - testHundirBarcoHorizontalCompleto
      - **Tipus de prova**: Caixa blanca.

        - **Tècnica**: Cobertura d'instruccions. Verifica que un vaixell horitzontal es considera enfonsat quan totes les seves parts han estat tocades.
    - testHundirBarcoVerticalCompleto
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura d'instruccions. Verifica que un vaixell vertical es considera enfonsat quan totes les seves parts han estat tocades.
    - testMultiplesBarcos
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura de camins. Verifica que diversos vaixells es poden col·locar i enfonsar correctament.
    - testDisparosFallidos
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura d'instruccions. Verifica que els disparos fallits es marquen com "Aigua".
    - testImprimirTablero
        - **Tipus de prova**: Caixa negra.
        - **Tècnica**: Particions equivalents. Verifica que el tauler es imprimeix correctament per consola.
    - testClonarTableroVacio
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura d'instruccions. Verifica que un tauler es clona correctament i es crea un tauler buit.
    - testInvarianteTablero
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura de camins. Verifica que les llistes internes del tauler són coherents amb el tauler mateix.
- **Comentaris**:
    - Es cobreixen els comportaments clau del tauler de joc, incloent la col·locació de vaixells (tant horitzontals com verticals), els disparos a les caselles (tant buides com ocupades per vaixells) i la detecció de vaixells enfonsats.
    - Es valida que el tauler gestioni adequadament les accions repetides (com disparar a la mateixa casella o intentar col·locar vaixells superposats).
    - S'assegura que els canvis en l'estat del tauler es reflecteixin correctament en la matriu i en els missatges retornats per la funció de disparos.

Funcionalitat: Col·locació de vaixells al tauler

- **Ubicació**:
    - Fitxer: uab.tqs.practica.model.Tablero
    - Mètode: colocarBarco(int fila, int columna, int longitud, boolean horizontal)
- **Proves**:
    - Fitxer: uab.tqs.practica.model.TestTablero

    - Mètodes de prova associats:
        - testColocarBarcoHorizontalExito
            - **Tipus de prova**: Caixa blanca.
            - **Tècnica**: Cobertura d'instruccions. Verifica que el vaixell es col·loca correctament a la matriu.
        - testColocarBarcoVerticalExito
            - **Tipus de prova**: Caixa blanca.
            - **Tècnica**: Cobertura d'instruccions. Verifica que el vaixell es col·loca correctament de forma vertical.
        - testColocarBarcosSuperpuestos
            - **Tipus de prova**: Caixa negra.
            - **Tècnica**: Particions equivalents. Verifica que no es permet la col·locació de vaixells superposats.
        - testColocarBarcoFueraLimites
            - **Tipus de prova**: Caixa negra.
            - **Tècnica**: Verificació d'errors. Verifica que no es pot col·locar un vaixell fora dels límits del tauler.
- **Comentaris**:
    - El mètode de col·locació de vaixells està exhaustivament provat, cobrint tant els casos d'èxit (col·locar vaixells dins del tauler) com els fallits (intentar col·locar-los fora dels límits o superposats).

Funcionalitat: Gestió dels disparos

- **Ubicació**:
    - Fitxer: uab.tqs.practica.model.Tablero
    - Mètodes: recibirDisparo(int fila, int columna) i imprimirTablero()
- **Proves**:
    - Fitxer: uab.tqs.practica.model.TestTablero
    - Mètodes de prova associats:
        - testDisparoAgua
            - **Tipus de prova**: Caixa blanca.
            - **Tècnica**: Cobertura d'instruccions. Verifica que es retorna "Aigua" quan un disparo toca una casella buida.
        - testDisparoTocado
            - **Tipus de prova**: Caixa blanca.
            - **Tècnica**: Cobertura d'instruccions. Verifica que es marca "Tocat" quan un disparo toca una part d'un vaixell.
        - testDisparoRepetido
            - **Tipus de prova**: Caixa negra.
            - **Tècnica**: Particions equivalents. Verifica que no es permet disparar dues vegades a la mateixa casella.

    - testDisparoRepetidoEnBarco
        - **Tipus de prova**: Caixa negra.
        - **Tècnica**: Verificació d'errors. Verifica que no es pot disparar a un vaixell ja tocat.
    - testDisparosFallidos
        - **Tipus de prova**: Caixa blanca.
        - **Tècnica**: Cobertura d'instruccions. Verifica que els disparos fallits es registren com "Aigua".
- **Comentaris**:
    - Les proves de disparos inclouen la verificació de les condicions per tocar un vaixell, fallar el disparo, i evitar repeticions de disparos a la mateixa casella.
    - També s'assegura que els disparos es gestionen adequadament en el tauler.

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
battleship_tqs2024	31,7 %	892	1.923	2.815
src/test/java	26,1 %	508	1.440	1.948
uab.tqs.practica.model	33,7 %	508	999	1.507
TestJugador.java	0,0 %	0	580	580
TestBarco.java	0,0 %	0	355	355
TestPartida.java	0,0 %	0	64	64
TestTablero.java	100,0 %	508	0	508
uab.tqs.practica.controller	0,0 %	0	441	441
src/main/java	44,3 %	384	483	867
uab.tqs.practica.model	58,9 %	384	268	652
Jugador.java	0,0 %	0	228	228
Partida.java	0,0 %	0	27	27
Barco.java	85,1 %	57	10	67
Tablero.java	99,1 %	327	3	330
uab.tqs.practica.controller	0,0 %	0	158	158
uab.tqs.practica.view	0,0 %	0	31	31
uab.tqs.practica	0,0 %	0	26	26

Conclusió:

- **Cobertura de funcionalitat**: Alta. Tots els comportaments clau del tauler de joc estan provats adequadament.
- **Cobertura de codi**: Alta. La major part de les instruccions de la classe Tablero són cobertes per les proves.

Informe de desenvolupament i test per a la classe Partida

Funcionalitat: Gestió de la partida entre dos jugadors

- **Localització**:
    - **Arxiu**: uab.tqs.practica.model.Partida
    - **Classe**: Partida
    - **Mètodes**:
        - Partida()
        - getJugador1()

        - getJugador2()
- **Test**:
    - **Arxiu**: uab.tqs.practica.model.TestPartida
    - **Mètodes de test associats**:
        - testInicializarJugadores:
            - **Tipus de test**: Caixa negra.
            - **Tècniques**: Verificació de valors. Comprova que els jugadors es inicialitzen correctament amb els noms "Jugador 1" i "Jugador 2".
        - testJugadoresDistintos:
            - **Tipus de test**: Caixa blanca.
            - **Tècniques**: Cobertura d'instruccions. Valida que els jugadors són instàncies diferents, per garantir que es creen dos objectes separats per a cada jugador.
        - testTablerosInicializados:
            - **Tipus de test**: Caixa blanca.
            - **Tècniques**: Cobertura d'instruccions. Comprova que els taulers dels jugadors no són nuls, assegurant que cada jugador té un tauler associat al moment de la creació.
- **Comentaris**:
    - Els tests asseguren que els jugadors es creen correctament amb els noms esperats i que els seus taulers es inicialitzen adequadament en el moment de la creació de la partida.
    - El test testJugadoresDistintos és essencial per garantir que cada jugador és una instància separada, evitant conflictes d'objectes compartits.
    - El test testTablerosInicializados assegura que els taulers no són nuls, la qual cosa és important per al correcte desenvolupament de la partida.
    -

Funcionalitat: Creació de la partida i associació dels jugadors

- **Localització**:
    - **Arxiu**: uab.tqs.practica.model.Partida
    - **Mètode**: Partida()
- **Test**:
    - **Arxiu**: uab.tqs.practica.model.TestPartida
    - **Mètodes de test associats**:
        - testInicializarJugadores:
            - **Tipus de test**: Caixa negra.
            - **Tècniques**: Verificació de valors. Es comprova que la partida inicialitza correctament els dos jugadors amb els noms establerts.
        - testTablerosInicializados:
            - **Tipus de test**: Caixa blanca.

        - **Tècniques**: Cobertura d'instruccions. Assegura que els taulers de cada jugador són creats al mateix temps que els jugadors.
- **Comentaris**:
    - Quan es crea la partida, els dos jugadors (Jugador 1 i Jugador 2) es creen automàticament amb els noms predefinits. Aquests noms es poden validar mitjançant els tests de comparació amb els valors esperats.
    - La inicialització dels taulers és una part fonamental perquè cada jugador pugui disposar del seu tauler per al desenvolupament del joc.

Funcionalitat: Obtenir els jugadors de la partida

- **Localització**:
    - **Arxiu**: uab.tqs.practica.model.Partida
    - **Mètodes**: getJugador1() i getJugador2()
- **Test**:
    - **Arxiu**: uab.tqs.practica.model.TestPartida
    - **Mètodes de test associats**:
        - testInicializarJugadores:
            - **Tipus de test**: Caixa negra.
            - **Tècniques**: Verificació de valors. Es comprova que es pot accedir als jugadors correctament mitjançant els mètodes getJugador1() i getJugador2().
        - testJugadoresDistintos:
            - **Tipus de test**: Caixa blanca.
            - **Tècniques**: Cobertura d'instruccions. Verifica que els jugadors retornats per aquests mètodes són diferents entre ells, el que és crucial per assegurar que cada jugador té una instància separada.
- **Comentaris**:
    - Els mètodes getJugador1() i getJugador2() proporcionen un accés fàcil als jugadors de la partida. Aquests mètodes han estat provats per garantir que es poden obtenir els jugadors correctament i que es mantenen com a instàncies independents.

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
TestTablero.java	0,0 %	0	389	389
TestJugador.java	0,0 %	0	387	387
TestBarco.java	0,0 %	0	116	116
TestPartida.java	100,0 %	64	0	64
uab.tqs.practica.model	15,9 %	95	503	598
Tablero.java	14,8 %	49	281	330
Jugador.java	13,0 %	25	167	192
Barco.java	0,0 %	0	55	55
Partida.java	100,0 %	21	0	21

Conclusió:

Els tests asseguren que la classe Partida i la seva funcionalitat per gestionar dos jugadors es comporten de manera robusta. Els jugadors són inicialitzats amb noms predefinits, els seus taulers són creats correctament i es pot obtenir cada jugador de manera independent. Els tests cobreixen escenaris clau com la inicialització i la separació d'instàncies entre jugadors, així com la verificació de que cada jugador té el seu tauler associat.

Informe de desenvolupament i test per a la classe PartidaController

Funcionalitat: Controlador de la partida

- **Localització:**
  - **Arxiu:** uab.tqs.practica.controller.PartidaController
  - **Classe:** PartidaController
  - **Mètodes:**
    - iniciarPartida(): Inicia la partida, configurant els jugadors i els seus taulers.
    - configurarBarcos(Jugador jugador): Configura els vaixells dels jugadors mitjançant el mètode colocarBarcos() i mostra els taulers amb vista.mostrarTablero().
    - iniciarJuego(): Comença el cicle de torns fins que un jugador guanya.
    - disparar(Jugador jugadorAtacante, Jugador jugadorDefensor): Gestiona la mecànica de disparar entre jugadors, actualitzant els taulers de disparo i mostrant el resultat.
    - limpiarConsola(): Netaja la consola després de cada acció per mantenir una visualització neta durant el joc.
- **Test:**
- **Arxiu de test:** uab.tqs.practica.controller.TestPartidaController
  - **Mètodes de test associats:**
    - testIniciarPartida(): Verifica que quan comença la partida, els jugadors col·loquen els seus vaixells i que es mostren els seus taulers.
    - testConfigurarBarcos(): Comprova que el mètode configurarBarcos() col·loca els vaixells dels jugadors correctament i que es mostren els taulers.
    - testDisparar_Tocado(): Prova que el mètode disparar funciona bé quan el disparo és un èxit ("Tocado").
    - testDisparar_Agua(): Comprova que el mètode disparar funciona correctament quan el disparo és un fallar ("Agua").
    - testIniciarJuego_Jugador1Gana(): Comprova que el joc finalitza correctament quan el jugador 1 guanya.
    - testIniciarJuego_Jugador2Gana(): Verifica que el joc finalitza correctament quan el jugador 2 guanya.
    - testLimpiarConsola(): Assegura que la consola es neteja correctament després de cada acció.
    - testDisparar_FueraDeLimites(): Comprova que el mètode disparar llança una excepció quan les coordenades de disparo estan fora de límits.

- testCambioTurno_Acierto(): Verifica que els torns es gestionen correctament quan un jugador encerta un disparo.

Comentaris:

- **Configuració de la partida:**
  Els tests simulen les accions de la partida per validar que el controlador interactua correctament amb els jugadors i la vista. En concret, el test testIniciarPartida() verifica que, quan es comença la partida, els jugadors col·loquen els seus vaixells i que els taulers es mostren correctament després de cada acció.
- **Interacció durant el joc:**
  Els tests testDisparar_Tocado() i testDisparar_Agua() validen que el controlador gestiona correctament els disparos, tant si són encerts com si són fallos. Es comprova que els missatges de la vista s'actualitzen correctament i que els taulers de disparo es modifiquen en conseqüència.
- **Fi de la partida:**
  Els tests testIniciarJuego_Jugador1Gana() i testIniciarJuego_Jugador2Gana() asseguren que el sistema detecta quan un jugador guanya la partida i mostra el missatge de guanyador corresponent. Aquestes proves validen que el flux del joc es tanca correctament en detectar que un jugador ha perdut.
- **Neteges de la consola:**
  El mètode testLimpiarConsola() comprova que la consola es neteja després de cada acció, tot i que no es pot verificar directament la neteja de la consola en els tests automatitzats. Aquesta validació assegura que la visualització del joc es mantingui clara i sense errors de format.
- **Gestió d'errors i validació:**
  El test testDisparar_FueraDeLimites() verifica que el mètode disparar llança una excepció quan les coordenades d'un disparo són invàlides, assegurant així que el sistema gestiona correctament els errors d'entrada.
- **Encert i canvi de torn:**
  Finalment, el test testCambioTurno_Acierto() comprova que els torns es gestionen adequadament quan un jugador encerta un disparo. Aquesta prova assegura que el sistema registra correctament l'encert i canvia el torn en conseqüència.

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
battleship_tqs2024	20,1 %	563	2.239	2.802
src/test/java	21,0 %	407	1.528	1.935
uab.tqs.practica.model	0,0 %	0	1.507	1.507
uab.tqs.practica.controller	95,1 %	407	21	428
TestPartidaController.java	95,1 %	407	21	428
src/main/java	18,0 %	156	711	867
uab.tqs.practica.model	0,0 %	0	652	652
uab.tqs.practica.view	0,0 %	0	31	31
uab.tqs.practica	0,0 %	0	26	26
uab.tqs.practica.controller	98,7 %	156	2	158
PartidaController.java	98,7 %	156	2	158

Conclusió:
Els tests cobreixen diversos aspectes essencials de la funcionalitat del PartidaController, incloent

la configuració de la partida, la mecànica dels disparos, la gestió dels torns, i la finalització del joc. Els mètodes estan dissenyats per separar les responsabilitats de configuració, accions de joc i visualització, i els tests asseguren que el flux del joc es desenvolupi de manera coherent i sense errors. Aquests tests proporcionen una validació exhaustiva de la lògica del controlador, la qual cosa contribueix a una millor qualitat de codi i a la detecció ràpida de possibles errors durant el desenvolupament.

Funcionalitat: Vista de la consola

- **Localització:**
  - **Arxiu:** uab.tqs.practica.view.ConsolaVista
  - **Classe:** ConsolaVista
  - **Mètodes:**
    - **mostrarMensaje(String mensaje):** Mostra un missatge a la consola. Aquest mètode imprimeix qualsevol missatge passat com a paràmetre a la consola, permetent la interacció amb l'usuari durant el joc.
    - **mostrarTablero(String[][] tablero):** Mostra el tauler del jugador a la consola. Aquest mètode rep un tauler com a matriu bidimensional i imprimeix cada fila a la consola per tal de visualitzar l'estat del joc de manera clara.
- **Comentaris:**
  - **Interacció amb la vista:** Els mètodes mostrarMensaje() i mostrarTablero() són les úniques maneres en què la classe ConsolaVista interactua amb l'usuari i mostra informació del joc. Aquests mètodes són responsables de proporcionar una experiència visual a l'usuari durant la partida.
  - **Simplicitat de la implementació:** La classe ConsolaVista és senzilla i eficient en la seva tasca. Només té dues funcions bàsiques: mostrar missatges i mostrar taulers. La seva responsabilitat no és més que la de comunicar-se amb l'usuari a través de la consola, sense cap lògica de negoci o de manipulació del joc.
- **Conclusió:** La classe ConsolaVista és essencial per a la interacció amb l'usuari i per garantir que la informació sobre l'estat del joc es presenti de manera clara i visual. Els mètodes estan dissenyats per ser simples i directes, permetent una fàcil adaptació o modificació si es volgués canviar la manera de presentar la vista en el futur (per exemple, per a una interfície gràfica en lloc de la consola).

Funcionalitat: Classe Main

- **Localització:**
  - **Arxiu:** uab.tqs.practica.Main
  - **Classe:** Main
  - **Mètodes:**
    - **main(String[] args):** És el punt d'entrada de l'aplicació. Crea una instància de la partida, la vista i el controlador. Després, inicia la partida mitjançant el controlador que s'encarrega de gestionar el flux del joc.
- **Comentaris:**
  - **Funció del mètode main:** La classe Main és l'element inicial que posa en marxa el joc. Crea les instàncies necessàries de les classes Partida, ConsolaVista i

PartidaController, i les connecta entre elles. El mètode main invoca la funcionalitat per iniciar la partida, donant lloc a la interacció entre el model (Partida), la vista (ConsolaVista) i el controlador (PartidaController).

- **Sincronització entre classes:** La classe Main serveix com a punt d'entrada i coordinació entre les diferents parts del sistema. Permet la inicialització i execució de la lògica del joc sense necessitat de canviar el flux d'execució o la configuració del joc, garantint una estructura neta i modular del codi.

- **Conclusió:** La classe Main compleix la funció de coordinar l'inici de la partida. Això permet a les altres classes (controlador, vista i model) interactuar entre si, seguint l'arquitectura del patró MVC. Aquesta separació facilita la gestió del codi i permet futures ampliacions o canvis en la lògica del joc sense afectar la resta del sistema.

Informe de l'utilització de Design by Contract en el projecte

Aplicació a la classe Tablero

La clase Tablero es una parte fundamental del juego de *Batalla Naval* y gestiona las operaciones relacionadas con la colocación de barcos y la recepción de disparos. A continuación se detallan algunos ejemplos donde se aplica el concepto de *Design by Contract* en esta clase:

Precondiciones

La classe Tablero és una part fonamental del joc i gestiona les operacions relacionades amb la col·locació de vaixells i la recepció de disparos. A continuació es detallen alguns exemples on s'aplica el concepte de *Design by Contract* en aquesta classe:

```
if (fila < 0 || fila >= tamaño || columna < 0 || columna >= tamaño) {
    throw new IllegalArgumentException("Las coordenadas del barco están fuera del tablero.");
}
```

En aquest cas, la precondició és que el vaixell ha de cabre completament dins del tauler sense excedir els seus límits horitzontals.

Postcondicions

Les postcondicions de la funció colocarBarco() verifiquen que després de col·locar un vaixell al tauler, les caselles corresponents estiguin ocupades pel vaixell ("B").

```
for (int i = 0; i < longitud; i++) {
    matriz[fila][columna + i] = "B";
}
```

La postcondició aquí és que les caselles corresponents al vaixell han de ser marcades com a ocupades amb "B". Si alguna de les precondicions no es compleix, el vaixell no es col·loca, i l'estat del tauler roman sense canvis.

Invariant

Una invariant del tauler és que sempre ha de tenir una mida fixa, en aquest cas de 10x10. A més, ha d'estar buit abans que es col·loquin vaixells o rebin trets. Això assegura que no hi hagi interaccions inesperades entre les operacions del joc, com la col·locació de vaixells en posicions ja ocupades o fora del tauler.

El tauler s'inicialitza amb el valor "~" per a totes les caselles, cosa que representa aigua i assegura que les caselles estiguin buides abans de col·locar un vaixell:

```
private void inicializarTablero() {
    for (int i = 0; i < tamaño; i++) {
        for (int j = 0; j < tamaño; j++) {
            matriz[i][j] = "~";
        }
    }
}
```

Això és un exemple d'una invariant, ja que l'estat del tauler ha de ser consistent en tot moment, sense canvis inesperats en les caselles que no estiguin relacionades amb les operacions del joc.

Aplicació a la classe Barco

Les precondicions en la classe Barco defineixen els requisits que han de complir-se abans de realitzar una operació o de crear un objecte.

- Precondició en el constructor: Quan es crea un vaixell amb un determinat tamany (longitud), es verifica que la longitud sigui superior a 0, ja que no té sentit crear un vaixell de longitud zero o negativa. Si la longitud no compleix aquest requisit, es llança una excepció IllegalArgumentException.

```
if (longitud <= 0) {
    throw new IllegalArgumentException("La longitud del barco debe ser mayor que 0.");
}
```

Aquesta precondició garanteix que qualsevol vaixell creat tingui una longitud vàlida.

- Precondició en el registre d'impactes: En el mètode registrarImpacto(), es comprova que la posició especificada estigui dins dels límits de l'array impactos (és a dir, que la posició sigui vàlida dins de la longitud del vaixell). Si la posició no està dins d'aquest rang, es llança una excepció IllegalArgumentException.

```
if (posicion < 0 || posicion >= impactos.length) {
    throw new IllegalArgumentException("Posición fuera de rango.");
}
impactos[posicion] = true;
```

Aquesta precondició evita que es registrin impactes en posicions no vàlides del vaixell.

Postcondicions

Les postcondicions en la classe Barco defineixen les condicions que han de complir-se després d'executar una operació.

- **Postcondició en el mètode registrarImpacto()**: Després de registrar un impacte en una posició, el valor de l'array impactos en la posició especificada es marca com true, indicant que aquesta part del vaixell ha estat tocada.

```
impactos[posicion] = true;
```

Després d'executar el mètode, la casella corresponent del vaixell ha de tenir el valor true, indicant que aquesta part del vaixell ha estat tocada. Si la posició és vàlida, la postcondició s'haurà complert.

- **Postcondició en el mètode estaHundido()**: El mètode estaHundido() retorna true només si tots els elements de l'array impactos són true. Això significa que totes les parts del vaixell han estat tocades i, per tant, el vaixell està hundit.

```
public boolean estaHundido() {
    for (boolean impacto : impactos) {
        if (!impacto) {
            return false;
        }
    }
    return true;
}
```

Després d'executar aquest mètode, el resultat serà correcte si totes les posicions del vaixell han rebut un impacte.

Invariants

Les invariants són condicions que han de mantenir-se certes en tot moment per a un objecte. Per a la classe Barco, les invariants són:

- **Invariant de la longitud del vaixell**: Un cop creat el vaixell, la seva longitud ha de ser constant i no pot canviar al llarg de la vida de l'objecte. Això es reflecteix en la inicialització de la variable longitud al constructor:

```
this.longitud = longitud;
this.impactos = new boolean[longitud];
```

Aquesta invariant garanteix que el vaixell tingui un nombre d'impactes corresponent a la seva longitud i que no s'hi puguin afegir o eliminar elements.