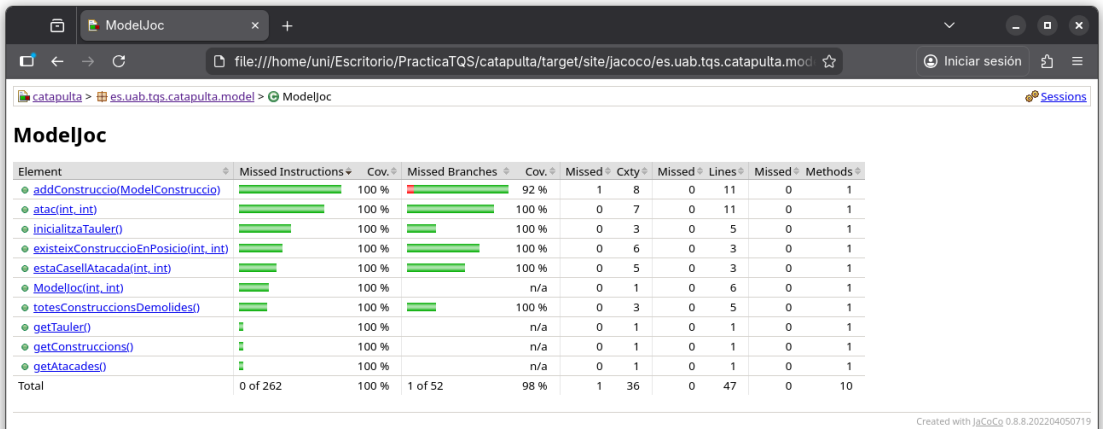
Joan Valentín Sánchez Rodríguez
Jan García I Girona
Dijous – 15:00
Catapulta

Model

ModelJoc.java



Funcionalitat 1: Inicialització del tauler

Funcionalitat: Inicialitza el tauler del joc amb una matriu de dimensions especificades (width x height), marcant totes les caselles com a buides (valor 0) i cap casella com atacada.

Localització: ModelJoc.java - ModelJoc.inicialitzaTauler()

Test: ModelJocTest.java - ModelJocTest.testInicialitzaTauler(). Test de caixa blanca utilitzant les tècniques de loop testing (verificació de bucles nested per recórrer tota la matriu) i statement coverage (cobertura del 100% de les sentències del mètode).

Funcionalitat 2: Afegir construcció al tauler

Funcionalitat: Afegeix una construcció al tauler assignant-li un identificador únic i marcant les caselles corresponents segons la seva mida i posició inicial. Valida que la construcció no sigui null i gestiona construccions que surten parcialment dels límits del tauler.

Localització: ModelJoc.java - ModelJoc.addConstruccio()

Test: ModelJocTest.java - ModelJocTest.testAddConstruccio(). Test de caixa negra i blanca combinant particions equivalents (construcció vàlida, null, fora de límits), valors límit i frontera (coordenades (0,0), (4,4), (-1,-1)), decision coverage (verificació de la branca if construccio == null), condition coverage (condicions de límits del tauler) i loop testing (bucles nested per marcar totes les caselles de la construcció).

Funcionalitat 3: Atacar una posició

Funcionalitat: Processa un atac a una posició específica del tauler. Verifica si les coordenades estan dins dels límits, si la casella ja ha estat atacada prèviament, i si hi ha una construcció en aquesta posició. Marca la casella com atacada i, si hi ha construcció, registra l'impacte.

Localització: ModelJoc.java - ModelJoc.atac()

Test: ModelJocTest.java - ModelJocTest.testAtac(). Test de caixa blanca utilitzant particions equivalents (casella buida vs construcció present), valors límit i frontera (cantonades (0,0), (0,4), (4,0), (4,4) i fora de límits (-1,-1), (5,5)), decision coverage (totes les branques if cobertes), condition coverage (condicions complexes amb operadors lògics), path coverage (4 camins principals: out of bounds, ja atacada, construcció hit, aigua) i pairwise testing (combinacions de coordenades i estat de construcció).

Funcionalitat 4: Verificar si una casella està atacada

Funcionalitat: Consulta si una casella específica ja ha estat atacada anteriorment. Retorna false si les coordenades estan fora dels límits del tauler.

Localització: ModelJoc.java - ModelJoc.estaCasellAtacada()

Test: ModelJocTest.java - ModelJocTest.testEstaCasellAtacada(). Test de caixa negra amb particions equivalents (casella atacada vs no atacada), valors límit i frontera (coordenades fora de límits (-1,0), (5,0), (0,-1), (0,5)) i condition coverage (verificació de totes les condicions del if compost).

Funcionalitat 5: Verificar existència de construcció en posició

Funcionalitat: Determina si existeix una construcció en una posició específica del tauler consultant el valor de la matriu (valor > 0 indica construcció present).

Localització: ModelJoc.java - ModelJoc.existeixConstruccioEnPosicio()

Test: ModelJocTest.java - ModelJocTest.testExisteixConstruccioEnPosicio(). Test de caixa negra utilitzant particions equivalents (casella buida vs construcció present), valors límit i frontera (coordenades fora de límits) i condition coverage (verificació de totes les condicions de límits i del valor de retorn).

Funcionalitat 6: Comprovar si totes les construccions estan demolides

Funcionalitat: Itera sobre la llista de construccions del joc i verifica si totes han estat completament demolides. Retorna true només si totes les construccions tenen l'estat estaDemolida() a true.

Localització: ModelJoc.java - ModelJoc.totesConstruccionsDemolides()

Test: ModelJocTest.java - ModelJocTest.testTotesConstruccionsDemolides(). Test de caixa blanca amb particions equivalents (cap demolida, algunes demolides, totes demolides), decision coverage (verificació de la condició if dins del loop), loop testing (iteració sobre múltiples construccions) i path coverage (camins amb llista buida, una construcció intacta, totes demolides).

Funcionalitat 7: Obtenir el tauler

Funcionalitat: Retorna la matriu que representa el tauler del joc amb els identificadors de les construccions en cada posició.

Localització: ModelJoc.java – ModelJoc.getTauler()

Test: ModelJocTest.java - ModelJocTest.testInicialitzaTauler() i ModelJocTest.testAddConstruccio(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna la referència correcta a la matriu.

Funcionalitat 8: Obtenir les construccions

Funcionalitat: Retorna la llista de totes les construccions afegides al joc.

Localització: ModelJoc.java - ModelJoc.getConstruccions()

Test: ModelJocTest.java - ModelJocTest.testAddConstruccio() i ModelJocTest.testTotesConstruccionsDemolides(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna la referència correcta a la llista.

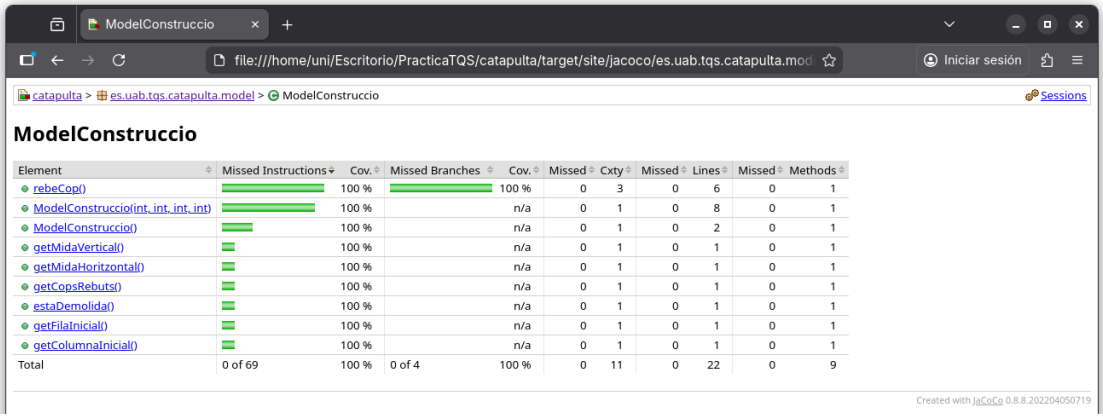
Funcionalitat 9: Obtenir les caselles atacades

Funcionalitat: Retorna la matriu booleana que indica quines caselles del tauler han estat atacades.

Localització: ModelJoc.java - ModelJoc.getAtacades()

Test: ModelJocTest.java - ModelJocTest.testAtac(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna la referència correcta a la matriu booleana.

ModelConstruccio.java



Funcionalitat 1: Inicialització de construcció amb paràmetres

Funcionalitat: Crea una nova construcció amb paràmetres de mida (vertical i horitzontal) i posició inicial (fila i columna). Inicialitza els atributs copsRebuts a 0 i estaDemolida a false.

Localització: ModelConstruccio.java - ModelConstruccio.ModelConstruccio(int, int, int, int)

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra amb particions equivalents (constructor amb paràmetres vàlids), valors límit i frontera (coordenades negatives (-1,-5), coordenades 0, diferents mides 2x3) i statement coverage (100% de les sentències del constructor).

Funcionalitat 2: Inicialització de construcció per defecte

Funcionalitat: Crea una nova construcció amb valors per defecte: mida 1x1 i posició inicial (0,0). Utilitza el constructor amb paràmetres internament.

Localització: ModelConstruccio.java - ModelConstruccio.ModelConstruccio()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra amb particions equivalents (constructor per defecte) i statement coverage (verificació que els valors inicialitzats són 1, 1, 0, 0).

Funcionalitat 3: Rebre cop a la construcció

Funcionalitat: Registra un impacte a la construcció incrementant el comptador de cops rebuts. Calcula si la construcció ha estat completament demolida comparant els cops rebuts amb el nombre total de caselles (midaVertical × midaHoritzontal). Si la construcció ja està demolida, no incrementa el comptador.

Localització: ModelConstruccio.java - ModelConstruccio.rebeCop()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testRebeCop(). Test de caixa blanca utilitzant particions equivalents (construcció no demolida vs demolida), decision coverage (verificació de les branques !estaDemolida i copsRebuts >= copsNecessaris), condition coverage (condicions dins dels if), loop testing (múltiples cops fins a demolició en construcció 3x3 amb 9 iteracions), path coverage (camins abans de demolició, després de demolició, cop a construcció ja demolida) i pairwise testing (combinacions de mides 1x1, 2x2, 3x3, 1x2, 2x1).

Funcionalitat 4: Obtenir mida vertical

Funcionalitat: Retorna l'altura de la construcció (nombre de files que ocupa).

Localització: ModelConstruccio.java - ModelConstruccio.getMidaVertical()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna el valor correcte de midaVertical.

Funcionalitat 5: Obtenir mida horitzontal

Funcionalitat: Retorna l'amplada de la construcció (nombre de columnes que ocupa).

Localització: ModelConstruccio.java - ModelConstruccio.getMidaHoritzontal()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna el valor correcte de midaHoritzontal.

Funcionalitat 6: Obtenir cops rebuts

Funcionalitat: Retorna el nombre total de cops que ha rebut la construcció fins al moment.

Localització: ModelConstruccio.java - ModelConstruccio.getCopsRebuts()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor() i ModelConstruccioTest.testRebeCop(). Test de caixa negra utilitzant statement coverage per verificar que el getter retorna el valor correcte de copsRebuts en diferents estats (0 inicial, incrementat després de rebeCop()).

Funcionalitat 7: Verificar si està demolida

Funcionalitat: Retorna l'estat de la construcció indicant si ha estat completament demolida (true) o encara té caselles sense impactar (false).

Localització: ModelConstruccio.java - ModelConstruccio.estaDemolida()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor() i ModelConstruccioTest.testRebeCop(). Test de caixa negra amb particions equivalents

(construcció intacta vs demolida), decision coverage (verificació de canvi d'estat després de rebeCop()) i statement coverage.

Funcionalitat 8: Obtenir fila inicial

Funcionalitat: Retorna la coordenada de fila on comença la construcció en el tauler.

Localització: ModelConstruccio.java - ModelConstruccio.getFilaInicial()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra amb valors límit i frontera (coordenades negatives -1, coordenades 0, coordenades positives) utilitzant statement coverage.

Funcionalitat 9: Obtenir columna inicial

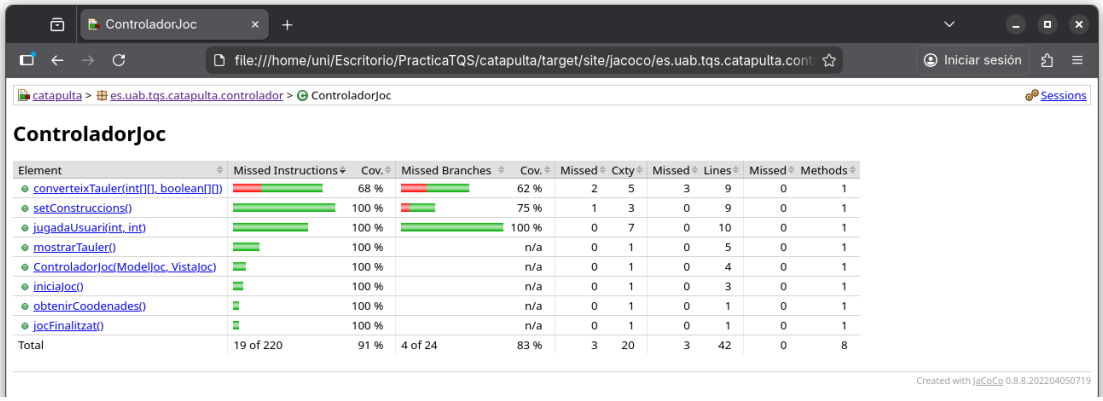
Funcionalitat: Retorna la coordenada de columna on comença la construcció en el tauler.

Localització: ModelConstruccio.java - ModelConstruccio.getColumnaInicial()

Test: ModelConstruccioTest.java - ModelConstruccioTest.testConstructor(). Test de caixa negra amb valors límit i frontera (coordenades negatives -5, coordenades 0, coordenades positives) utilitzant statement coverage.

Controlador

ControladorJoc.java



Funcionalitat 1: Iniciar el joc

Funcionalitat: Inicia el joc mostrant el tauler inicial a través de la vista i enviant un missatge de benvinguda a l'usuari per indicar que el joc ha començat.

Localització: ControladorJoc.java - ControladorJoc.iniciaJoc()

Test: ControladorJocTest.java - ControladorJocTest.testIniciaJoc(). Test de caixa blanca utilitzant mock objects (VistaJoc mockejada), statement coverage (100% de les sentències) i verificació d'interaccions amb verify() per assegurar que es crida mostraTauler() i mostraMissatges() amb els paràmetres correctes.

Funcionalitat 2: Processar jugada de l'usuari

Funcionalitat: Processa la jugada de l'usuari validant les coordenades introduïdes, verificant si la casella ja ha estat atacada, executant l'atac en el model i mostrant el missatge corresponent segons el resultat (atac correcte o aigua). Llança una excepció si les coordenades estan fora dels límits del tauler.

Localització: ControladorJoc.java - ControladorJoc.jugadaUsuari()

Test: ControladorJocTest.java - ControladorJocTest.testJugadaUsuari(). Test de caixa blanca combinant particions equivalents (casella buida, construcció present, casella ja atacada), valors límit i frontera (coordenades fora de límits (-1,0), (0,-1), (5,0), (0,5)), decision coverage (totes les branques if cobertes), condition coverage (condicions complexes de validació), path coverage (4 camins: out of bounds amb excepció, casella atacada, aigua, hit) i mock objects (verificació de missatges amb verify()).

Funcionalitat 3: Obtenir coordenades de l'usuari

Funcionalitat: Delega la tasca d'obtenir les coordenades introduïdes per l'usuari a la vista, retornant un array amb les coordenades x i y.

Localització: ControladorJoc.java - ControladorJoc.obtenirCoodenades()

Test: ControladorJocTest.java - ControladorJocTest.testObtenirCoodenades(). Test de caixa negra amb mock objects (Scanner mockejat per simular entrada d'usuari), particions equivalents (coordenades vàlides) i valors límit i frontera (coordenades (2,3), (0,4)) utilitzant statement coverage.

Funcionalitat 4: Establir construccions predefinides

Funcionalitat: Valida que el tauler estigui correctament inicialitzat i afegeix 5 construccions predefinides de mida 1x1 en posicions específiques del tauler: (0,0), (0,2), (2,2), (3,1) i (4,3).

Localització: ControladorJoc.java - ControladorJoc.setConstruccions()

Test: ControladorJocTest.java - ControladorJocTest.testSetConstruccions(). Test de caixa blanca amb particions equivalents (tauler vàlid, tauler buit), decision coverage (verificació de la branca if tauler.length == 0), condition coverage (condicions de validació del tauler), path coverage (camí correcte afegint 5 construccions, camí d'excepció amb tauler buit) i valors límit i frontera (tauler 0x0 que llança IllegalStateException).

Funcionalitat 5: Verificar si el joc ha finalitzat

Funcionalitat: Consulta al model si totes les construccions han estat demolides per determinar si el joc ha finalitzat.

Localització: ControladorJoc.java - ControladorJoc.jocFinalitzat()

Test: ControladorJocTest.java - ControladorJocTest.testJocFinalitzat(). Test de caixa negra amb particions equivalents (construccions intactes, algunes demolides, totes demolides), path coverage (camí de joc no finalitzat, camí de joc finalitzat), loop testing (iteració per destruir 5 construccions) i statement coverage.

Funcionalitat 6: Mostrar el tauler

Funcionalitat: Obté l'estat actual del tauler i de les caselles atacades del model, converteix aquesta informació en una representació de caràcters (~ per no atacada, O per aigua, X per construcció colpejada) i delega la visualització a la vista.

Localització: ControladorJoc.java - ControladorJoc.mostrarTauler()

Test: ControladorJocTest.java - ControladorJocTest.testMostrarTauler(). Test de caixa blanca amb mock objects (verificació de crides a vista.mostraTauler()), loop testing (iteració per convertir totes les caselles del tauler), statement coverage (100% de les sentències) i particions equivalents (tauler buit, tauler amb construccions).

Funcionalitat 7: Convertir tauler a representació de caràcters

Funcionalitat: Transforma la matriu numèrica del tauler i la matriu booleana d'atacs en una matriu de caràcters utilitzant la convenció: '~' per caselles no atacades, 'O' per atacs fallats (aigua) i 'X' per construccions colpejades.

Localització: ControladorJoc.java - ControladorJoc.converteixTauler()

Test: ControladorJocTest.java - ControladorJocTest.testMostrarTauler(). Test de caixa blanca amb loop testing (nested loops per recórrer tota la matriu), decision coverage (verificació de les 3 branques del if-else: no atacada, aigua, construcció), condition coverage (condicions ! atacades[i][j] i tauler[i][j] == 0) i statement coverage (100% de les sentències del mètode privat).

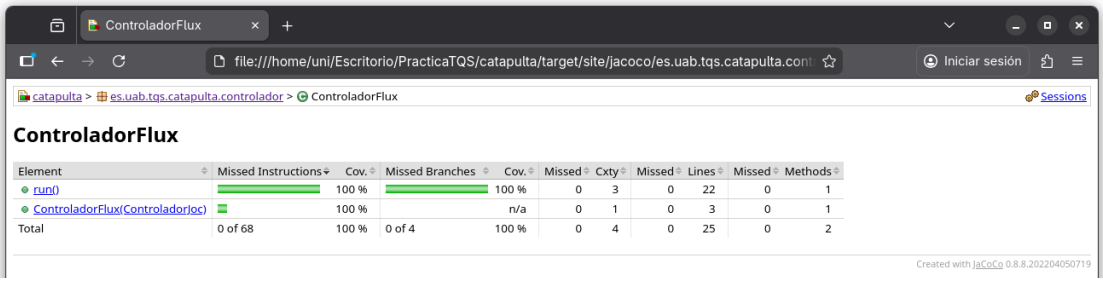
Funcionalitat 8: Inicialització del controlador

Funcionalitat: Crea una nova instància del controlador assignant les referències del model i la vista que utilitzarà per coordinar la lògica del joc.

Localització: ControladorJoc.java - ControladorJoc.ControladorJoc()

Test: ControladorJocTest.java - ControladorJocTest.setUp(). Test de caixa negra amb statement coverage (100% del constructor) executat al @BeforeEach de tots els tests.

ControladorFlux.java



Funcionalitat 1: Executar el bucle principal del joc

Funcionalitat: Gestiona el flux complet del joc des de la inicialització fins a la finalització. Mostra el missatge de benvinguda, estableix les construccions al tauler, inicia el joc i executa el bucle principal que obté coordenades de l'usuari, processa jugades, actualitza el tauler i verifica si el joc ha finalitzat. Gestiona excepcions d'IndexOutOfBoundsException per coordenades invàlides i mostra el missatge final amb el nombre d'intents realitzats.

Localització: ControladorFlux.java - ControladorFlux.run()

Test: ControladorFluxTest.java - ControladorFluxTest.testRun(). Test de caixa blanca utilitzant mock objects (ControladorJoc completament mockejat amb when() i doThrow()), path coverage (3 camins principals: excepció en primera iteració amb catch, jugada OK amb joc no finalitzat, jugada OK amb joc finalitzat), loop testing (bucle while amb 4 iteracions), decision coverage (verificació de les branques try-catch i if jocFinalitzat), condition coverage (condició while jocActiu) i statement coverage. Utilitza doThrow().doReturn() per simular

comportaments seqüencials i verify() amb times() per verificar el nombre exacte de crides a cada mètode del controlador.

Funcionalitat 2: Inicialització del controlador de flux

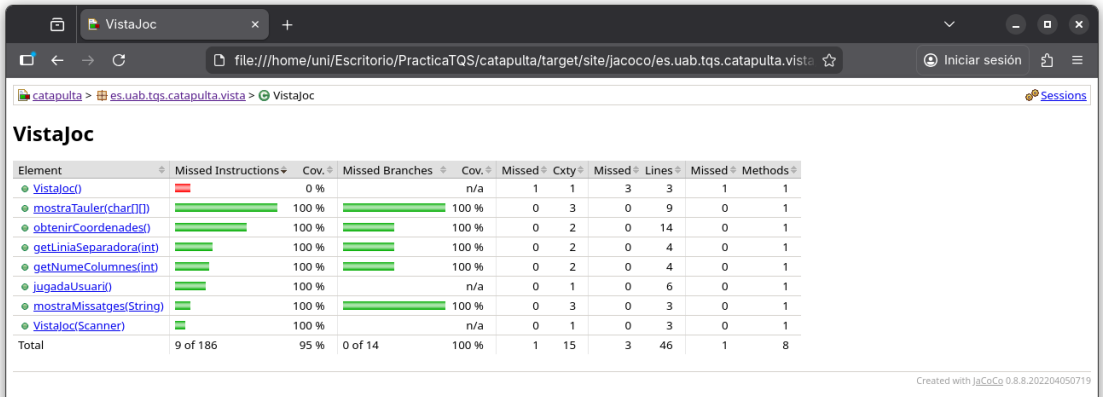
Funcionalitat: Crea una nova instància del controlador de flux assignant la referència del ControladorJoc que utilitzarà per gestionar la lògica del joc durant l'execució del bucle principal.

Localització: ControladorFlux.java - ControladorFlux.ControladorFlux()

Test: ControladorFluxTest.java - ControladorFluxTest.testRun(). Test de caixa negra amb statement coverage (100% del constructor) executat dins del mètode testRun() en crear la instància de ControladorFlux amb el controlador mockejat.

Vista

VistaJoc.java



Funcionalitat 1: Mostrar el tauler de joc

Funcionalitat: Mostra el tauler de joc a la consola amb format visual incloent capçalera amb números de columnes, línies separadores i el contingut de cada casella. Utilitza nested loops per recórrer tota la matriu i mostrar cada fila amb els seus separadors.

Localització: VistaJoc.java - VistaJoc.mostraTauler()

Test: VistaJocTest.java - VistaJocTest.testMostraTauler(). Test de caixa blanca amb loop testing (nested loops per iterar files i columnes del tauler), particions equivalents (tauler 2x2 amb diferents caràcters), statement coverage (100% de les sentències del mètode), pairwise testing (combinacions de dimensions i caràcters: (2x2, 'A'/'B'), (3x3, 'X'/'Y'/'Z')) i verificació de sortida capturant System.out per comprovar que conté números de columnes, separadors i caràcters del tauler.

Funcionalitat 2: Mostrar missatges a l'usuari

Funcionalitat: Mostra missatges informatius a l'usuari per consola. Valida que el missatge no sigui null ni buit abans de mostrar-lo, ignorant missatges invàlids.

Localització: VistaJoc.java - VistaJoc.mostraMissatges()

Test: VistaJocTest.java - VistaJocTest.testMostraMissatges(). Test de caixa blanca amb particions equivalents (missatge null, string buit, missatge vàlid), decision coverage (verificació de les branques if missatge != null i !missatge.isEmpty()), condition coverage (condicions del if compost amb operador &&) i statement coverage, capturant System.out per verificar que només es mostra el missatge vàlid.

Funcionalitat 3: Obtenir coordenades de l'usuari

Funcionalitat: Demana a l'usuari que introdueixi les coordenades (fila i columna) per atacar, validant que siguin números vàlids. Gestiona excepcions InputMismatchException quan l'usuari introdueix text en lloc de números, netejant el buffer i repetint la petició fins obtenir entrada vàlida.

Localització: VistaJoc.java - VistaJoc.obtenirCoordenades()

Test: VistaJocTest.java - VistaJocTest.testObtenirCoordenades(). Test de caixa blanca amb mock objects (Scanner mockejat amb when().thenThrow().thenReturn()), loop testing (bucle while amb gestió d'excepcions múltiples), decision coverage (verificació de les branques try-catch i while !coordenadesValides), path coverage (camí amb entrada vàlida directa, camí amb InputMismatchException i retry), particions equivalents (entrada vàlida vs invàlida) i valors límit i frontera (coordenades (2,3), (0,4)), verificant amb verify() el nombre de crides a nextInt() i nextLine().

Funcionalitat 4: Processar jugada de l'usuari (mètode auxiliar)

Funcionalitat: Demana a l'usuari les coordenades per atacar llegint directament dos valors enters del Scanner sense validació. Mètode auxiliar utilitzat principalment per testing.

Localització: VistaJoc.java - VistaJoc.jugadaUsuari()

Test: VistaJocTest.java - VistaJocTest.testJugadaUsuari(). Test de caixa negra amb mock objects (Scanner mockejat amb when().thenReturn(4, 0)), valors límit i frontera (coordenades (4,0)) i statement coverage, verificant amb verify() que es crida nextInt() dues vegades i que la sortida conté els textos "Fila" i "Columna".

Funcionalitat 5: Generar capçalera amb números de columnes

Funcionalitat: Genera una cadena de text amb els números de columnes del tauler separats per espais, utilitzada com a capçalera visual del tauler mostrat.

Localització: VistaJoc.java - VistaJoc.getNumeColumnes()

Test: VistaJocTest.java - VistaJocTest.testMostraTauler(). Test de caixa blanca amb loop testing (bucle for per iterar des de 0 fins a numColumnes), statement coverage (100% de les sentències) i verificació indirecta capturant System.out per comprovar que apareixen els números "0", "1", "2" en la sortida del tauler.

Funcionalitat 6: Generar línia separadora

Funcionalitat: Genera una línia de separació amb guions ('-') d'una longitud determinada pel nombre de columnes del tauler multiplicat per 2 més 1, utilitzada per separar visualment les files del tauler.

Localització: VistaJoc.java - VistaJoc.getLiniaSeparadora()

Test: VistaJocTest.java - VistaJocTest.testMostraTauler(). Test de caixa blanca amb loop testing (bucle for per generar la línia de guions), statement coverage (100% de les sentències) i verificació indirecta capturant System.out per comprovar que apareixen els separadors "--" en la sortida del tauler.

Funcionalitat 7: Inicialització de la vista amb Scanner per defecte

Funcionalitat: Crea una nova instància de la vista inicialitzant el Scanner amb System.in per permetre l'entrada d'usuari des de la consola en l'execució real del joc.

Localització: VistaJoc.java - VistaJoc.VistaJoc()

Test: No testejat. Constructor utilitzat només en execució real (Main.java). No és necessari per a la cobertura de tests ja que és codi de producció sense lògica complexa.

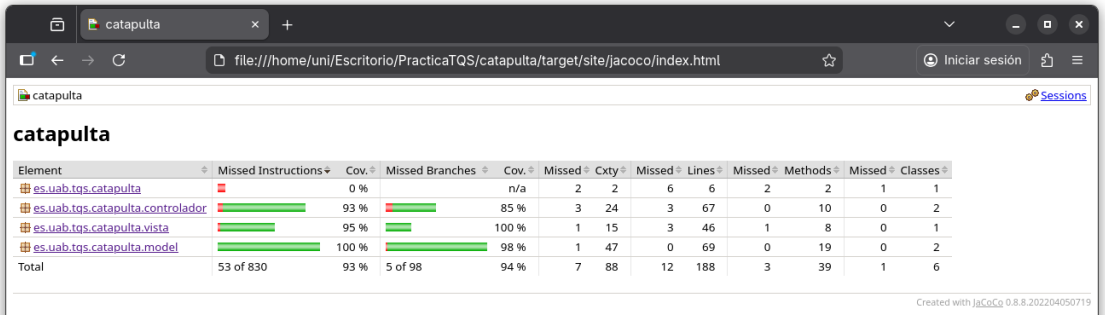
Funcionalitat 8: Inicialització de la vista amb Scanner injectat

Funcionalitat: Crea una nova instància de la vista acceptant un Scanner com a paràmetre, permetent la injecció de dependències per facilitar el testing amb mocks.

Localització: VistaJoc.java - VistaJoc.VistaJoc(Scanner)

Test: VistaJocTest.java - Utilitzat en tots els tests (testMostraTauler, testMostraMissatges, testObtenirCoordenades, testJugadaUsuari). Test de caixa negra amb mock objects (Scanner mockejat injectat al constructor) i statement coverage (100% del constructor).

Test Coverage



Model

ModelJoc.java

```
1   package es.uab.tqs.catapulta.model;
2
3   import java.util.ArrayList;
4   import java.util.List;
5
6   // Model del joc: manté l'estat del tauler, les construccions i els atacs efectuats.
7   public class ModelJoc {
8
9       // Estat del tauler: 0 buit, >0 identifica la construcció col·locada.
10      private int[][] tauler; // Matriu que representa el tauler del joc (0=Buit, >0=ID construcció)
11      private List<ModelConstruccio> construccions; // Llista de construccions col·locades
12      private boolean[][] atacades; // Matriu que marca caselles ja atacades
13
14      // Crea un joc amb dimensions donades i inicialitza tauler i marcatge d'atacs.
15      public ModelJoc(int width, int height) {
16          this.tauler = new int[height][width];
17          this.atacades = new boolean[height][width];
18          this.construccions = new ArrayList<>();
19          inicialitzaTauler();
20      }
21
22      private void inicialitzaTauler() {
23          // Omple el tauler amb buits i neteja l'historial d'atacs.
24          for (int i = 0; i < tauler.length; i++) {
25              for (int j = 0; j < tauler[i].length; j++) {
26                  tauler[i][j] = 0; // 0 representa un espai buit
27                  atacades[i][j] = false;
28              }
29          }
30      }
31
32      public void addConstruccio(ModelConstruccio construccio) {
33          // Afegeix una construcció; cas límit: null -> excepció.
34          if (construccio == null) {
35              throw new IllegalArgumentException("La construcció no pot ser null");
36          }
37
38          int id = construccions.size() + 1;
39          construccions.add(construccio);
40
41          // Marca les cel·les que ocupa; part exterior al tauler es descarta silenciosament.
42          for (int i = 0; i < construccio.getMidaVertical(); i++) {
43              for (int j = 0; j < construccio.getMidaHoritzontal(); j++) {
44                  int fila = construccio.getFilaInicial() + i;
45                  int columna = construccio.getColumnaInicial() + j;
46
47                  if (fila >= 0 && fila < tauler.length && columna >= 0 && columna < tauler[0].length) {
48                      tauler[fila][columna] = id;
49                  }
50              }
51          }
52      }
53
54      public boolean atac(int x, int y) {
55          // Partició frontera: coordenades fora del tauler -> atac invàlid.
56          if (x < 0 || x >= tauler.length || y < 0 || y >= tauler[0].length) {
57              return false; // Fora de límits
58          }
59
60          // Partició equivalent: casella ja atacada -> no es repeteix.
61          if (atacades[x][y]) {
62              return false; // Ja ha estat atacada
63          }
64
65          atacades[x][y] = true;
66          int construccioId = tauler[x][y];
67
68          if (construccioId > 0) {
69              // Hi ha una construcció en aquesta posició; es registra el cop.
70              ModelConstruccio construccio = construccions.get(construccioId - 1);
71              construccio.rebeCop();
72              return true; // Atac correcte
73          }
74
75          return false; // Casella buida
76      }
77
78      public boolean estaCasellAtacada(int x, int y) {
79          // Consulta segura: fora de rang es considera no atacat.
80          if (x < 0 || x >= tauler.length || y < 0 || y >= tauler[0].length) {
81              return false;
82          }
83          return atacades[x][y];
84      }
85
86      public boolean existeixConstruccioEnPosicio(int x, int y) {
87          // Consulta segura: fora de rang es considera sense construcció.
88          if (x < 0 || x >= tauler.length || y < 0 || y >= tauler[0].length) {
89              return false;
90          }
91          return tauler[x][y] > 0;
92      }
93
94      public boolean totesConstruccionsDemolides() {
95          // Condició de victòria: totes les construccions han estat demolides.
96          for (ModelConstruccio construccio : construccions) {
97              if (!construccio.estaDemolida()) {
98                  return false;
99              }
100         }
101         return true;
102     }
103
104     public int[][] getTauler() {
105         return this.tauler;
106     }
107
108     public List<ModelConstruccio> getConstruccions() {
109         return this.construccions;
110     }
111
112     public boolean[][] getAtacades() {
113         return this.atacades;
114     }
115 }
```

ModelConstruccio.java

```
1.  package es.uab.tqs.catapulta.model;
2.
3.  // Representa una construcció al tauler amb mida i estat de demolició.
4.  public class ModelConstruccio {
5.
6.      // Dimensions i localització de la peça al tauler.
7.      private int midaVertical;       // Altura de la construcció
8.      private int midaHoritzontal;    // Amplada de la construcció
9.      private int filaInicial;        // Fila inicial de la construcció
10.     private int columnaInicial;     // Columna inicial de la construcció
11.
12.     // Estat de danys i demolició.
13.     private int copsRebuts;         // Nombre de cops rebuts
14.     private boolean estaDemolida;   // Estat de la construcció
15.
16.     // Constructor principal: defineix mida i posició de la construcció.
17.     public ModelConstruccio(int midaVertical, int midaHoritzontal, int filaInicial, int columnaInicial) {
18.         this.midaVertical = midaVertical;
19.         this.midaHoritzontal = midaHoritzontal;
20.         this.filaInicial = filaInicial;
21.         this.columnaInicial = columnaInicial;
22.         this.copsRebuts = 0;
23.         this.estaDemolida = false;
24.     }
25.
26.     // Constructor per defecte: peça mínima 1x1 a la cantonada superior esquerra.
27.     public ModelConstruccio() {
28.         this(1, 1, 0, 0);
29.     }
30.
31.     public int getMidaVertical() {
32.         return midaVertical;
33.     }
34.
35.     public int getMidaHoritzontal() {
36.         return midaHoritzontal;
37.     }
38.
39.     public int getCopsRebuts() {
40.         return copsRebuts;
41.     }
42.
43.     public boolean estaDemolida() {
44.         return estaDemolida;
45.     }
46.
47.     public int getFilaInicial() {
48.         return filaInicial;
49.     }
50.
51.     public int getColumnaInicial() {
52.         return columnaInicial;
53.     }
54.
55.     public void rebeCop() {
56.         // Incrementa el comptador i comprova cas límit: suficients impactes per demolir.
57.         if (!estaDemolida) {
58.             copsRebuts++;
59.             int copsNecessaris = midaVertical * midaHoritzontal;
60.             if (copsRebuts >= copsNecessaris) {
61.                 estaDemolida = true;
62.             }
63.         }
64.     }
65. }
```

Controlador

ControladorJoc.java

```
1.  package es.uab.tqs.catapulta.controlador;
2.
3.  import es.uab.tqs.catapulta.model.ModelJoc;
4.  import es.uab.tqs.catapulta.model.ModelConstruccio;
5.  import es.uab.tqs.catapulta.vista.VistaJoc;
6.
7.  // Coordina la interacció entre el model del joc i la vista de consola.
8.  public class ControladorJoc {
9.      // Estat intern del joc i canal de sortida/entrada amb l'usuari.
10.     private ModelJoc model;
11.     private VistaJoc vista;
12.
13.     // Injecció de dependències per treballar amb el model i la vista.
14.     public ControladorJoc(ModelJoc model, VistaJoc vista) {
15.         this.model = model;
16.         this.vista = vista;
17.     }
18.
19.     // Inicialitza la partida i mostra l'estat inicial del tauler.
20.     public void iniciaJoc() {
21.         mostrarTauler();
22.         vista.mostraMissatges("Joc iniciat. Comença a atacar!");
23.     }
24.
25.     public boolean jugadaUsuari(int x, int y) {
26.         // Processa una jugada d'usuari (ús habitual i proves unitàries).
27.         int[][] tauler = model.getTauler();
28.
29.         // Cas límit (frontera): coordenades fora del tauler -> excepció controlada.
30.         if (x < 0 || x >= tauler.length || y < 0 || y >= tauler[0].length) {
31.             throw new IndexOutOfBoundsException("Coordenades fora de límits");
32.         }
33.
34.         // Partició equivalent: casella ja atacada, no es torna a processar.
35.         if (model.estaCasellAtacada(x, y)) {
36.             return false;
37.         }
38.
39.         boolean atac = model.atac(x, y);
40.
41.         // Feedback de caixa negra a l'usuari segons l'èxit de l'atac.
42.         if (atac) {
43.             vista.mostraMissatges("Atac correcte! Has colpejat una construcció.");
44.         } else {
45.             vista.mostraMissatges("Mala sort! No hi ha res en aquesta posició.");
46.         }
47.
48.         return atac;
49.     }
50.
51.     public int[] obtenirCoodenades() {
52.         // Entrada d'usuari delegada a la vista (pot ser simulada en tests).
53.         return vista.obtenirCoordenades();
54.     }
55.
56.     public void setConstruccions() {
57.         // Configura l'escenari inicial amb construccions 1x1 per a un tauler 5x5.
58.         int[][] tauler = model.getTauler();
59.
60.         // Precondició: el tauler ha d'estar inicialitzat abans de col·locar peces.
61.         if (tauler.length == 0 || tauler[0].length == 0) {
62.             throw new IllegalStateException("El tauler no està inicialitzat correctament");
63.         }
64.
65.         // Distribució fixa per facilitar proves reproductibles (particions equivalents).
66.         model.addConstruccio(new ModelConstruccio(1, 1, 0, 0));  // Posició (0, 0)
67.         model.addConstruccio(new ModelConstruccio(1, 1, 0, 2));  // Posició (0, 2)
68.         model.addConstruccio(new ModelConstruccio(1, 1, 2, 2));  // Posició (2, 2)
69.         model.addConstruccio(new ModelConstruccio(1, 1, 3, 1));  // Posició (3, 1)
70.         model.addConstruccio(new ModelConstruccio(1, 1, 4, 3));  // Posició (4, 3)
71.     }
72.
73.     public boolean jocFinalitzat() {
74.         // Condició d'aturada: totes les construccions estan demolides.
75.         return model.totesConstruccionsDemolides();
76.     }
77.
78.     public void mostrarTauler() {
79.         // Converteix l'estat intern a símbols i el mostra via la vista.
80.         int[][] tauler = model.getTauler();
81.         boolean[][] atacades = model.getAtacades();
82.         char[][] taulerChar = converteixTauler(tauler, atacades);
83.         vista.mostraTauler(taulerChar);
84.     }
85.
86.     private char[][] converteixTauler(int[][] tauler, boolean[][] atacades) {
87.         // Tradueix l'estat del model a símbols: ~ sense atacar, O aigua, X encert.
88.         char[][] taulerChar = new char[tauler.length][tauler[0].length];
89.         for (int i = 0; i < tauler.length; i++) {
90.             for (int j = 0; j < tauler[i].length; j++) {
91.                 if (!atacades[i][j]) {
92.                     taulerChar[i][j] = '~'; // Casella no atacada
93.                 } else if (tauler[i][j] == 0) {
94.                     taulerChar[i][j] = 'O'; // Atac fallat (casella buida)
95.                 } else {
96.                     taulerChar[i][j] = 'X'; // Construcció colpejada
97.                 }
98.             }
99.         }
100.        return taulerChar;
101.    }
102. }
```

ControladorFlux.java

```java
package es.uab.tqs.catapulta.controlador;

// Controla el flux principal del joc de Catapulta des de la consola.
public class ControladorFlux {
    // Controlador de la lògica del joc (configuració, jugades i finalització).
    private final ControladorJoc controlador;

    // Injecció del controlador per preparar el flux d'execució.
    public ControladorFlux(ControladorJoc controlador) {
        this.controlador = controlador;
    }

    // Execució principal: inicia el joc, demana jugades i compta intents fins acabar.
    public void run() {
        System.out.println("Benvingut a Catapulta!");
        controlador.setConstruccions();
        controlador.iniciaJoc();
        controlador.mostrarTauler();

        boolean jocActiu = true;
        int intents = 0;

        // Bucle de partida: rep jugades de l'usuari fins eliminar totes les construccions.
        while (jocActiu) {
            // Coordenades facilitades per l'usuari via consola.
            int[] coords = controlador.obtenirCoodenades();
            int x = coords[0];
            int y = coords[1];
            try {
                // Processa la jugada, incrementa intents i refresca el tauler.
                controlador.jugadaUsuari(x, y);
                intents++;
                controlador.mostrarTauler();
                if (controlador.jocFinalitzat()) {
                    // Cas d'èxit: totes les construccions han estat destruïdes.
                    System.out.println("\n¡Felicitats! Has destruït totes les construccions en " + intents + " intents!");
                    jocActiu = false;
                }
            } catch (IndexOutOfBoundsException e) {
                // Coordenades fora del tauler: s'informa i es demanen noves jugades.
                System.out.println("Error: " + e.getMessage());
            }
        }

        System.out.println("Joc finalitzat. Fins aviat!");
    }
}
```

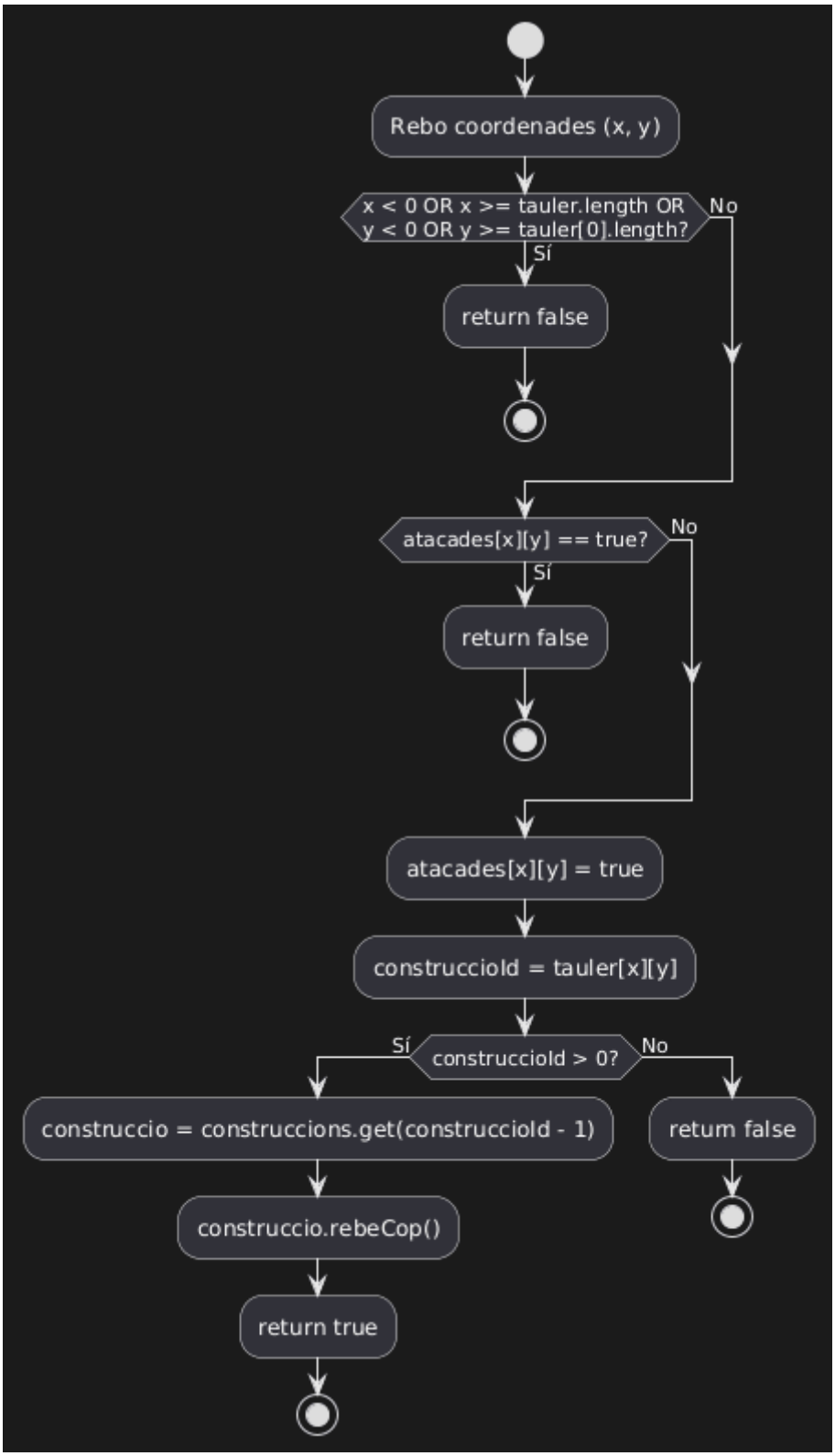
Vista

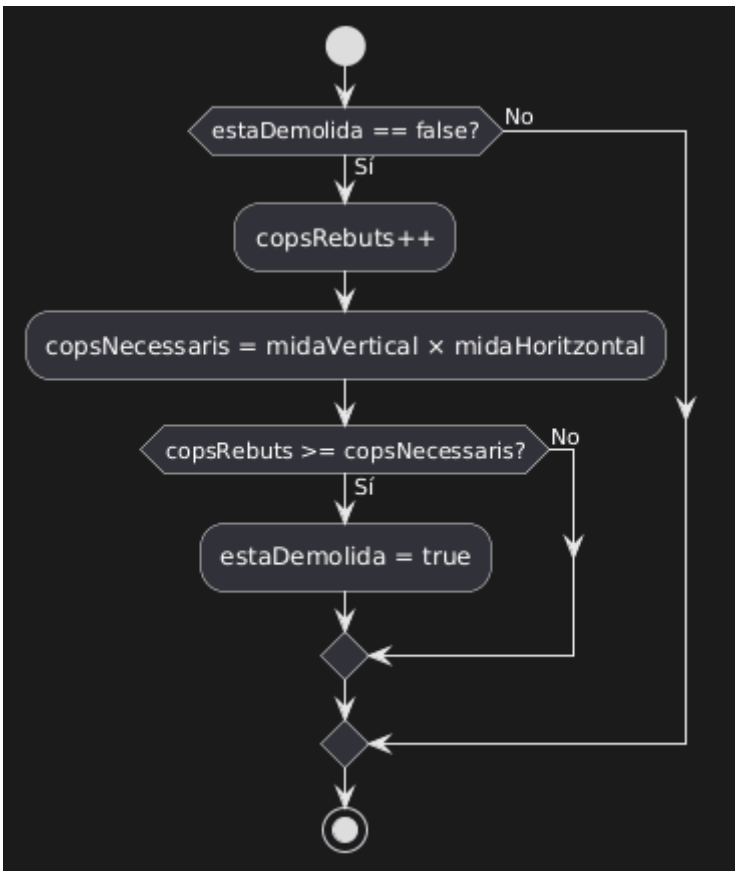
VistaJoc.java

```
1.  package es.uab.tqs.catapulta.vista;
2.
3.  import java.util.Scanner;
4.
5.  // Vista de consola: mostra el tauler i recull interaccions amb l'usuari.
6.  public class VistaJoc {
7.      // Canal d'entrada per llegir coordenades (substituïble en tests).
8.      private Scanner scanner;
9.
10.     public VistaJoc() {
11.         this.scanner = new Scanner(System.in);
12.     }
13.
14.     // Constructor per testing amb Scanner mockejat.
15.     public VistaJoc(Scanner scanner) {
16.         this.scanner = scanner;
17.     }
18.
19.     public void mostraTauler(char[][] tauler) {
20.         // Dibuixa el tauler en text: capçaleres de columnes, separadors i files.
21.         System.out.println("\n   " + getNumeColumnes(tauler[0].length));
22.         System.out.println("   " + getLiniaSeparadora(tauler[0].length));
23.
24.         for (int i = 0; i < tauler.length; i++) {
25.             System.out.print(i + " |");
26.             for (int j = 0; j < tauler[i].length; j++) {
27.                 System.out.print(tauler[i][j] + "|");
28.             }
29.             System.out.println();
30.             System.out.println("   " + getLiniaSeparadora(tauler[0].length));
31.         }
32.     }
33.
34.     public void mostraMissatges(String missatge) {
35.         // Presenta feedback a l'usuari si el text no és buit.
36.         if (missatge != null && !missatge.isEmpty()) {
37.             System.out.println(missatge);
38.         }
39.     }
40.
41.     public int[] obtenirCoordenades() {
42.         // Demana coordenades d'atac; bucle fins a entrada vàlida (gestiona InputMismatch).
43.         int[] coordenades = new int[2];
44.         boolean coordenadesValides = false;
45.
46.         while (!coordenadesValides) {
47.             try {
48.                 System.out.println("\nIntrodueix les coordenades per atacar:");
49.                 System.out.print("Fila (x): ");
50.                 coordenades[0] = scanner.nextInt();
51.                 System.out.print("Columna (y): ");
52.                 coordenades[1] = scanner.nextInt();
53.                 coordenadesValides = true;
54.             } catch (java.util.InputMismatchException e) {
55.                 System.out.println("Error: Introdueix números vàlids!");
56.                 scanner.nextLine(); // Neteja buffer per tornar a llegir.
57.             }
58.         }
59.
60.         return coordenades;
61.     }
62.
63.     public void jugadaUsuari() {
64.         // Demana una jugada directa (no retorna valors; útil per compatibilitat).
65.         System.out.println("\nIntrodueix les coordenades per atacar:");
66.         System.out.print("Fila (x): ");
67.         int x = scanner.nextInt();
68.         System.out.print("Columna (y): ");
69.         int y = scanner.nextInt();
70.     }
71.
72.     private String getNumeColumnes(int numColumnes) {
73.         StringBuilder sb = new StringBuilder();
74.         for (int i = 0; i < numColumnes; i++) {
75.             sb.append(i).append(" ");
76.         }
77.         return sb.toString();
78.     }
79.
80.     private String getLiniaSeparadora(int numColumnes) {
81.         StringBuilder sb = new StringBuilder();
82.         for (int i = 0; i < numColumnes * 2 + 1; i++) {
83.             sb.append("-");
84.         }
85.         return sb.toString();
86.     }
87. }
```

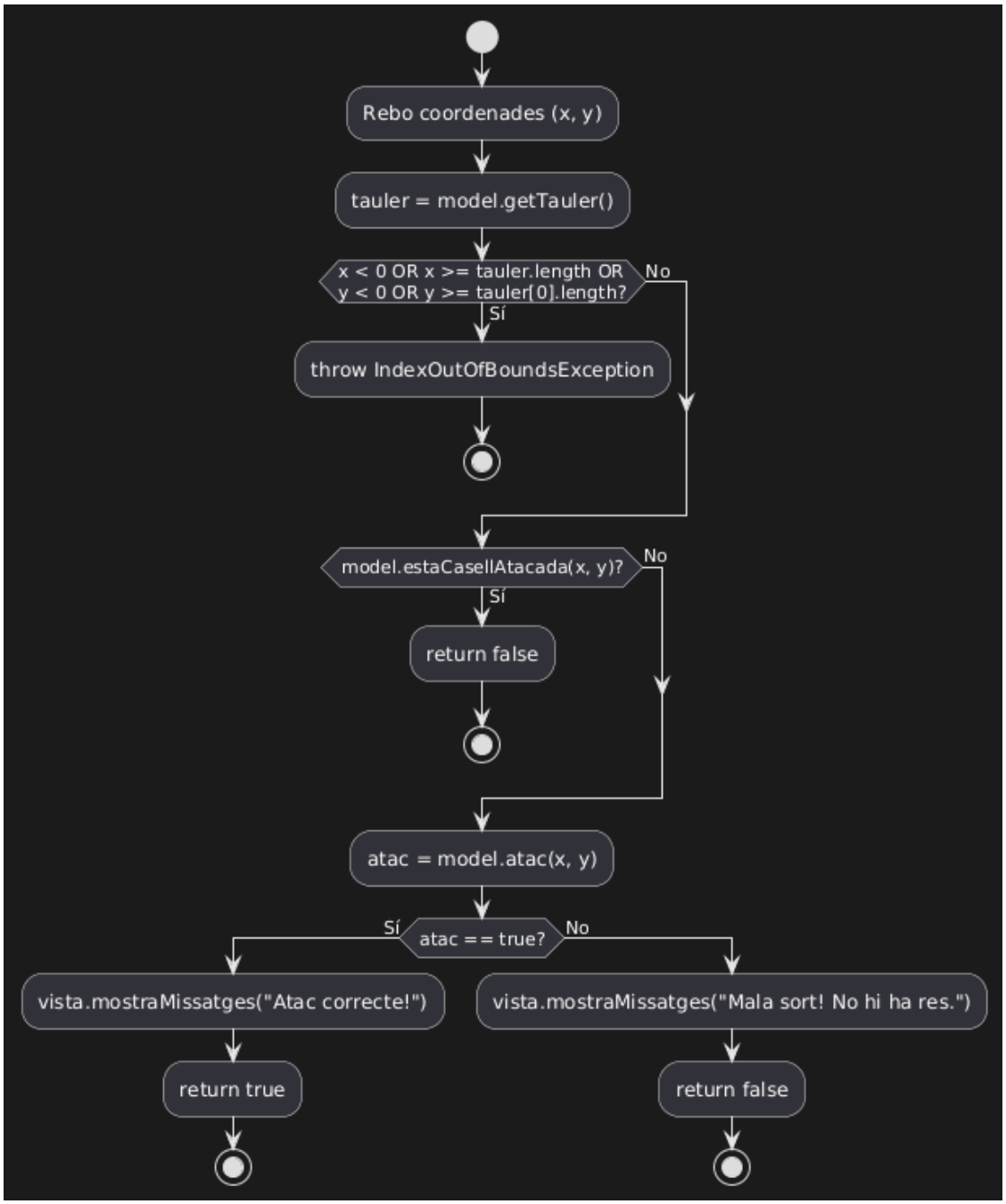
Diagrames de flux

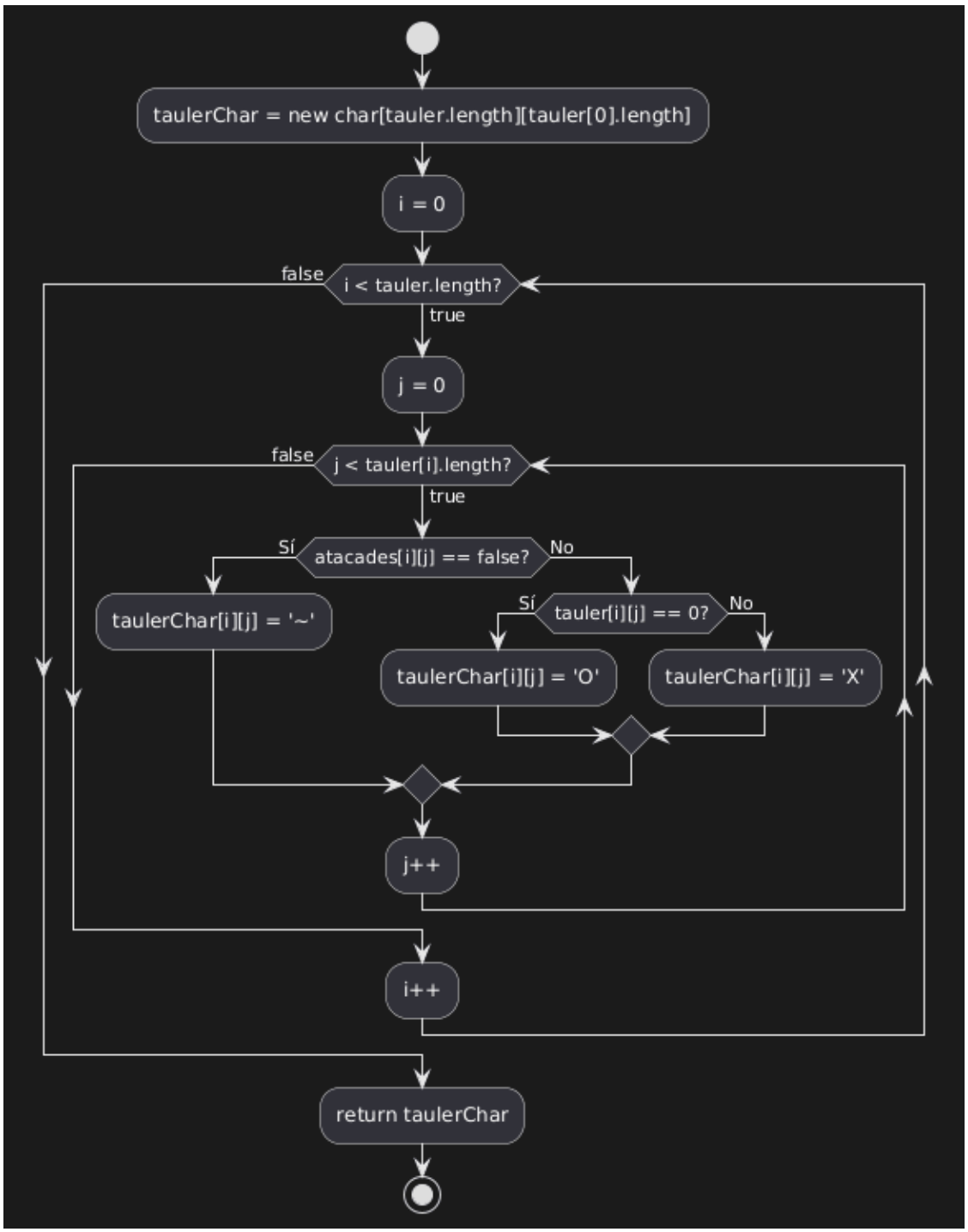
Model

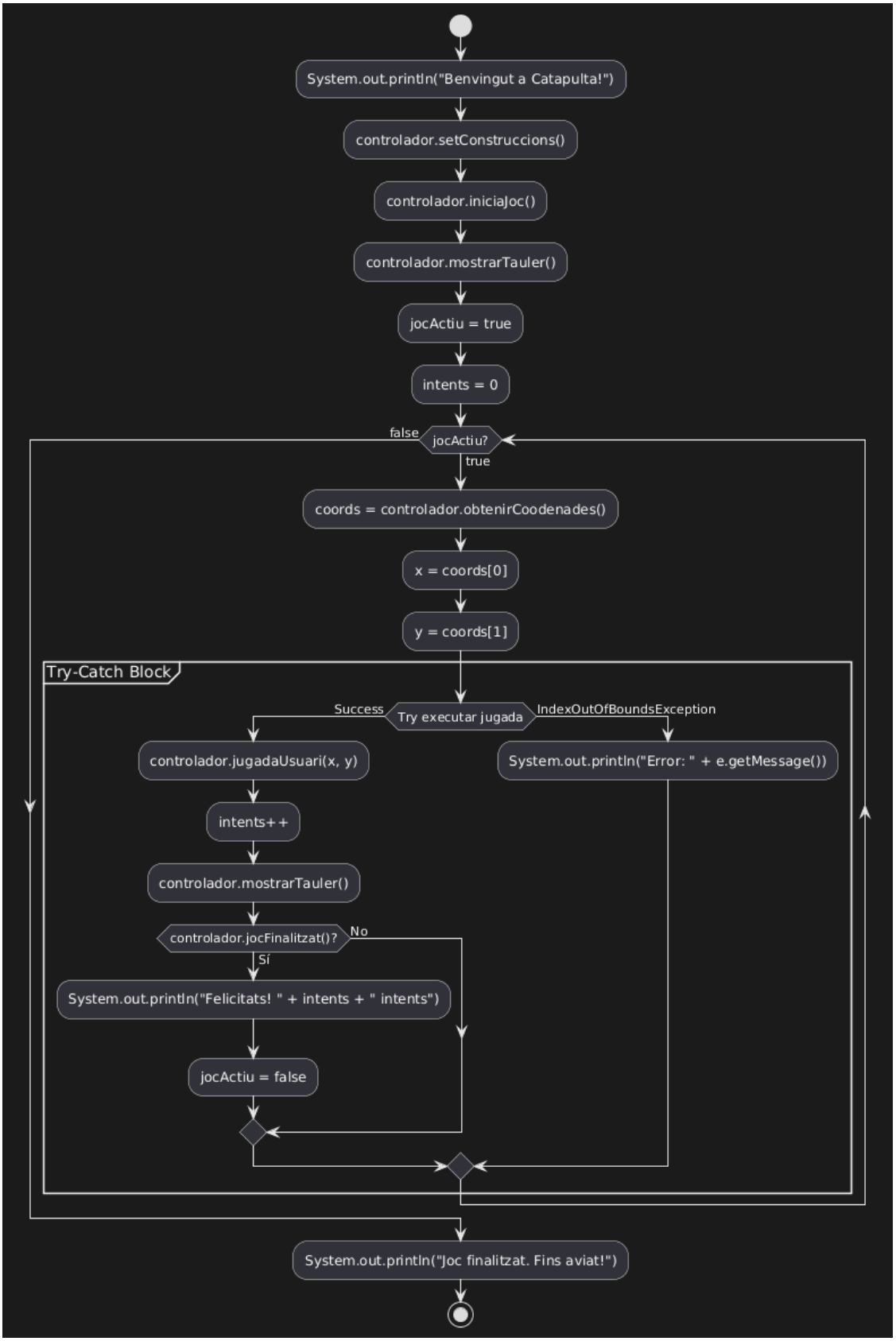



estaDemolida == false?
No
Sí
copsRebuts++
copsNecessaris = midaVertical × midaHoritzontal
copsRebuts >= copsNecessaris?
No
Sí
estaDemolida = true

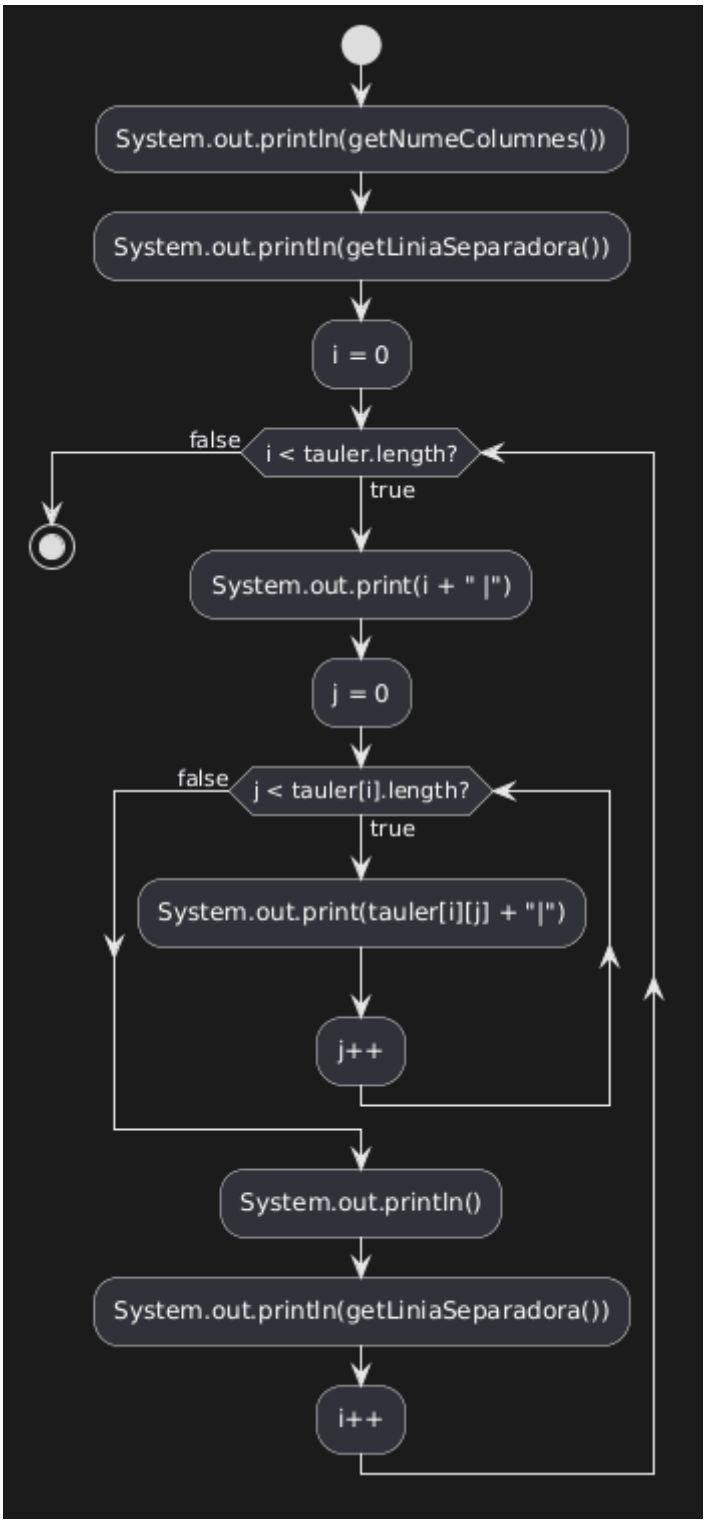
Controlador

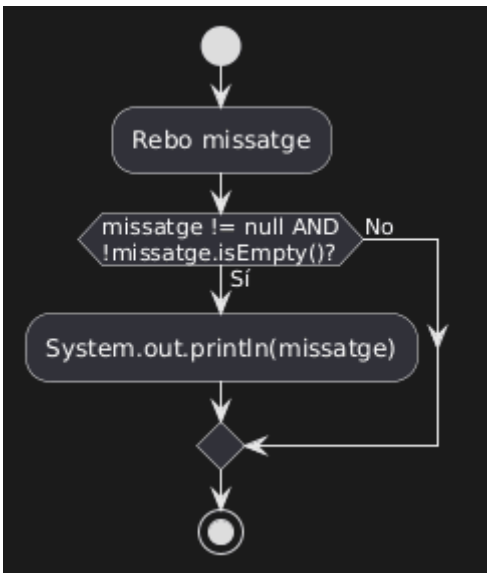


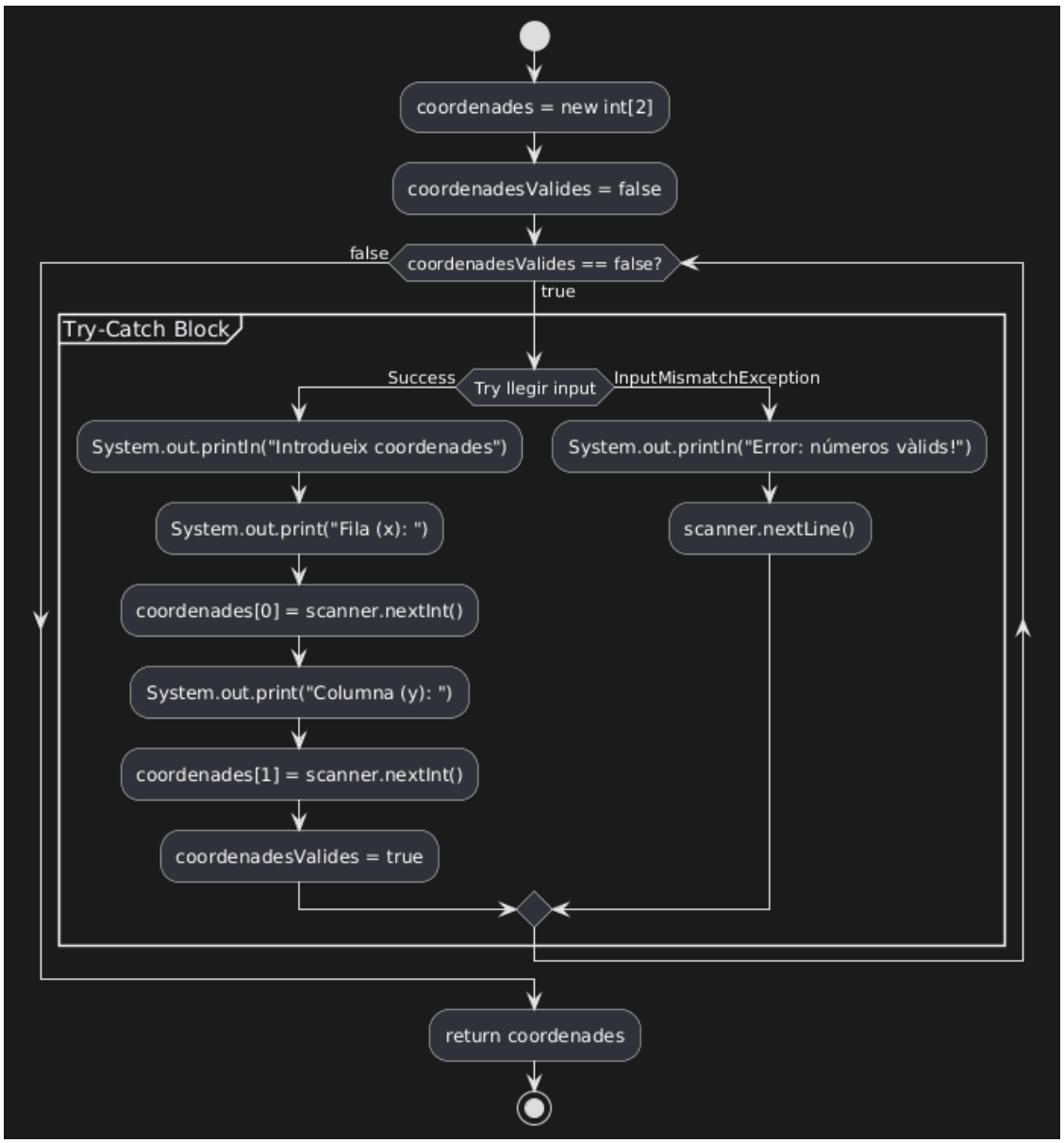

taulerChar = new char[tauler.length][tauler[0].length]
i = 0
false
i < tauler.length?
true
j = 0
false
j < tauler[i].length?
true
Sí
atacades[i][j] == false?
No
taulerChar[i][j] = '~'
Sí
tauler[i][j] == 0?
No
taulerChar[i][j] = 'O'
taulerChar[i][j] = 'X'
j++
i++
return taulerChar


System.out.println("Benvingut a Catapulta!")
controlador.setConstruccions()
controlador.iniciaJoc()
controlador.mostrarTauler()
jocActiu = true
intents = 0
false
jocActiu?
true
coords = controlador.obtenirCoodenades()
x = coords[0]
y = coords[1]
Try-Catch Block
Success
Try executar jugada
IndexOutOfBoundsException
controlador.jugadaUsuari(x, y)
System.out.println("Error: " + e.getMessage())
intents++
controlador.mostrarTauler()
controlador.jocFinalitzat()?
No
Sí
System.out.println("Felicitats! " + intents + " intents")
jocActiu = false
System.out.println("Joc finalitzat. Fins aviat!")

Vista


System.out.println(getNumeColumnes())
System.out.println(getLiniaSeparadora())
i = 0
false
i < tauler.length?
true
System.out.print(i + " |")
j = 0
false
j < tauler[i].length?
true
System.out.print(tauler[i][j] + "|")
j++
System.out.println()
System.out.println(getLiniaSeparadora())
i++


Rebo missatge
missatge != null AND
!missatge.isEmpty()?
No
Sí
System.out.println(missatge)


coordenades = new int[2]
coordenadesValides = false
false
coordenadesValides == false?
true
Try-Catch Block
Success
Try llegir input
InputMismatchException
System.out.println("Introdueix coordenades")
System.out.print("Fila (x): ")
coordenades[0] = scanner.nextInt()
System.out.print("Columna (y): ")
coordenades[1] = scanner.nextInt()
coordenadesValides = true
System.out.println("Error: números vàlids!")
scanner.nextLine()
return coordenades