**Mutliplayer Chess**

**[14.01.2021]**

|  |  |  |
| --- | --- | --- |
| **Îndrumător:** |  | **Student:** |
| **dr. ing. Daniel Morariu** |  | **Gavrilov Sabin**  **(211/1)** |

**Istoric Versiuni**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versiune** | **Descriere** | **Autor** |
| 12/09/2020 | 0.1 | In aceasta versiune, au fost facuta partea de functionare a sah-ului fara interfata grafica. | Gavrilov Sabin |
| 17/12/2020 | 0.2 | Versiunea aceasta jocul este funcitonal doar la nivel de un jucator, acesta nefiind prin server | Gavrilov Sabin |
|  |  | Chess.*getDisplay*().syncExec(new Runnable() {… | Sergiu Pop |
|  |  |  |  |

**Cuprins**

**ISTORIC VERSIUNI 2**

**CUPRINS 3**

**1** **SPECIFICAREA CERINȚELOR SOFTWARE 4**

**1.1 Introducere 4**

1.1.1 Obiective 4

1.1.2 Definiții, Acronime și Abrevieri 4

1.1.3 Tehnologiile utilizate 4

**1.2 Cerințe specifice 4**

**2** **FUNCȚIONALITATE 5**

**2.1 Descriere 5**

**2.2 Fluxul de evenimente 5**

2.2.1 Fluxul de bază 5

2.2.2 Pre-condiții 5

2.2.3 Post-condiții 5

**3** **FUNCȚIONALITATE**

**3.1 Descriere 5**

**3.2 Fluxul de evenimente 5**

3.2.1 Fluxul de bază 5

3.2.2 Pre-condiții 5

3.2.3 Post-condiții 5

**4** **IMPLEMENTARE 6**

**4.1 Diagrama de clase 6**

**4.2 Descriere detaliată 7**

**5** **BIBLIOGRAFIE 8**

# Specificarea cerințelor software

## Introducere

Joc de sah in retea

### Obiective

Implementarea unui joc de sah intre doi jucatori folosind comunicarea prin intermendiu unui server. Un obiectiv a fost acela ca in momentul selectarii unei piese sa se observe cumva faptul ca a fost selctata de catre juctorul care o muta(acum schimba culoarea label-ului selectat in rosu) . Un alt obiectiv este de a avea un istoric al mutarilor in care cei 2 jucatori sa poate vedea ce s-a mutat. Am tintit spre a face si un buton prin care sa fie integrata conexiunea la server si tot odata sa poate fi transmis si un mesaj de cerere start joc. Iar cand al 2-lea jucator da start sa poate fi trimisa inapoi un raspuns de start afirmativ si astfel sa poata incepe jocul.

Imbunnatatiri...

Transmiterea mutarii cum se face si pe interenet spre ex: din A4 -> G8. Mutari pe timp.

Grafica aplicatie:Animatie mutare, optiuni de undo, restart.

La inchiderea shell-ului sa apara celuilalt un mesaj cum ca jocul s-a sfarsit.

### Definiții, Acronime și Abrevieri

**JMS** - Java Message Service The Java Message Service (JMS) was designed to make it easy to develop business applications that asynchronously send and receive business data and events.

**Apache ActiveMQ** -- is the most popular open source, multi-protocol, Java-based messaging server.

**ActiveMQ** is a message broker which implements the **JMS** API and supports a number of cross language clients and network protocols

**API** is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other.

SWT(**Standard Widget Toolkit** ) - is the software component that delivers native widget functionality for the Eclipse platform in an operating system independent manner.

**Static method:**A method declared with the static keyword. Like static fields, static methods are associated with the class itself, not with any particular object created from the class. As a result, you don’t have to create an object from a class before you can use static methods defined by the class.

Chess.*getDisplay*().syncExec(new Runnable() {

*@Override*

public void run() {

}

}

* ce am in interiorul functiei run se executa pe thred-ul principal
* cu componentele grafice se poate lucra doar pe thred-ul principal

### Tehnologiile utilizate

ActiveMQ - Java-based messaging server Swt – o librarie pt widget-urile folosite Paint.net – am editat marimea pozelor si le-am centrat (60/60)

***Cerințe specifice***

In aceasta versiune de sah au fost implementate urmatoarele functionalitati:

Tabla este compusa dintr-o matrice de label-uri in care fiecare poate intercepta o actiune la selectarea unui label.

Mutarea este realizata prin selectarea unui label, acestuia fiindui modificata culoarea si apoi se selecteaza label-ul destinatie.

In interfata este si un buton de start prin care cei 2 jucatorii pot trimite cererea de start unul catre altul, jocul incepand cand ambii trimit cererea, urmand ca cel ce a trimis-o primul sa incepa.

O alta functionalitate este tabelul din interfata care tine evidenta mutarilor oponentului.

Mesajele sunt trimise cu ajutorul unui server de mesaje numit „ActiveMQ” meajul trimis avand mai multe proprietati. Am trimis coordonatele de start si de dest ca si string prin mesaj, tipul operatiei (start\_init, start\_ok,move) si id-ul jucatorului.

Mesajul este trimis intr-un topic creat de mine prin ActiveMQ la care au acces ambii jucatori. La primire mesajului, cel care l-a trimis il ignora, iar celalat actioneaza conform protocolului creeat de mine.

# Funcționalitate

## Descriere

**Mutarea pieselor** este una dintre functionalitatile de baza, fiind realizata cu ajutorul unei metode din clasa PiecesBoard unde tin matricea cu ordinea pieselor curenta.

Aceasta functie primeste coordonatele de start si de denstinatie urmand ca daca acestea sunt diferite, piesa sa fie mutata in matricea de piese de la sursa la destinatie.

Iar ca piesa sa apara ca si mutata in interfata se apeleaza sa o alta metoda din clasa AppBoard numita: updateTableMove() care primestea si ea coordonatele piesei de start si coor de destinatie.

## Fluxul de evenimente

### Fluxul de bază

Aceasta functie de mutare, modifica pozitia piesei in matricea cu jocul curent din spatele alpicatiei.

Ea este apelata in momentul in care jucatorul dupa ce a selectat o piesa selectaza locul unde doreste ca piesa sa fie mutata. Apoi coordonatele sunt trimise celeilalte table de sah urmand sa fie folosite in aceeasi functie.

Totodata aceasta functie se mai apeleaza si din clasa ActiveMQUtil in functia connect() in listenerul de primire a mesajelor cand tipul mersajului este TYPE\_MOVE, pt ca mutarea sa fie realizata si in matricea de joc a celui de al doilea jucator.

public Piece[ ][ ]move(int x\_start, int y\_start, int x\_stop, int y\_stop) {

// validari .... doar daca coordonatele sunt diferite daca nu returneaza maticea neschimbata

piese[x\_stop][y\_stop] = piese[x\_start][y\_start];

piese[x\_start][y\_start] = null; // sau o piesa "spatiu"

return piese;

}

Pentru ca aceasta mutare sa fie actualizata si la nivel de interfata se apelaza metoda udpateTableMove() in momentul selectarii de catre jucator a locatiei in care vrea sa mute piesa pe actiunea de mouseDown(). In functie incearca luarea imaginii de pe sursa care apoi se verficia de != null dupa care aceasta este setata la labelul cu coordonatele destinatie, ea fiind pusa pe null in sursa. Funcita este apelata in listener-ul unui label, mutarea fiind actualizata instant pe interfata.

Totodata aceasta functie se mai apeleaza si din clasa ActiveMQUtil in functia connect() in listenerul de primire a mesajelor cand tipul mersajului este TYPE\_MOVE, pt ca mutarea sa fie realizata si in interfata de joc a celui de al doilea jucator.

public void udpateTableMove(int xSrc, int ySrc, int xDest, int yDest) {

Image srcImg = piese[xSrc][ySrc].getImage();

if (srcImg != null && !srcImg.isDisposed()) {

piese[xDest][yDest].setImage(srcImg);

}

piese[xSrc][ySrc].setImage(null);

}

Mutarea este realizata prin selectarea unui label, acestuia fiindui modificata culoarea si apoi se selecteaza label-ul destinatie de catre jucator.

Pentru ca jucatorul sa poata muta trebuie sa fie randul lui, adica, in tabelul de informatii scrie „Rabdl meu” si atunci „tabla” i permite jucatorului sa apese pe aceasta.

### Pre-condiții

Setari server ActiveMQ – ip, port – sunt „hard-coded” = puse direct in cod. Ar putea fi puse intr-un fisiere de configurare.

Pentru ca jocul sa functioneze trebuie ca jucatorul sa aiba minim o versiune de java 1.8.0 si trebuie sa instaleze Active MQ 5 si apoi sa il porneasca in consola prin comanda: C:\dev\apache-activemq-5.16.0>bin\activemq start (in functie de locatia bin.

In primul rand utlilizatorul trebuie sa apese pe butonul de start urmand ca o cerere sa fie trimisa catre un posibil alt jucator care la randul lui cand primeste cererea trimite un raspuns inapoi ca si confirmare dupa care jocul a inceput. Jucatorul stie cand trebuie sa puna prima mutare deoarece pe interfata am creat un tabel de informatii unde va scrie cand e randul jucatorului si apoi cand primeste mutarea adversarului poate muta.

### Post-condiții

Dupa ce utilizatorul a apasat pe detinatia piesei, label-ul sursa care avea culoarea rosu cand era selectata va recapata culoarea lui initiala. Jucatorul va trebui sa astepte alt mesaj pe tabelul de informatii cum ca e randul lui.

# Funcționalitate

## Descriere

**Afisarea pieselor pe tabla de joc** se realizeaza cu ajutorul unei matrici de componente Label din libraria SWT. Aceste matrice este parcursa si fiecare label este asezat in pozitia lui pe un composite de pe shell, fiecare label avand o culoare specifica. Apoi se parcurge matricea intitiala din clasa PiecesBoard care este matricea cu piesele actulaizata la fiecare mutare si la fiecare label i atribuie imaginea piesei in functie de locatia acesteia iar in caz ca nu este piese i se atribuie null la imagine.

# *Fluxul de evenimente*

### Fluxul de bază

Operatia de afisare a pieselor pe tabla se realizeaza cu ajutorul functiei updateTableMove().

Aceasta functie a fost specifiacta si mai sus deoarece are si rolul in a afisa mutarea.

Matricea de label-uri este defapt ceea ce jucatorul vede ca si table de sah. Matricea e parcursa si fiecare label primeste cate o culoare si o dimensiune. Apoi pentru ca piesele sa fie afisate pe aceasta “tabla” intervine o functie de initializare a tablei de sah numita initChessTable() care pune in fiecare label piesa corespunzatoare din matricea curenta de piese a jocului. Mai exact seteaza fiecarui label o imagine sau null (daca img lipseste) :

public void initChessTable() {

// matricea de piese

Piece[][] board = appBoard.getBoard();

// tabla

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

if (board[i][j] != null) {

piese[i][j].setImage(board[i][j].loadImage());

} else {

if (piese[i][j].getImage() != null) {

piese[i][j].getImage().dispose();

}

}

}

}

}

La deshiderea aplicatiei tabla cu piesele este pe interfata. Apoi cand urmeaza mutarile , pt fiecare mutare de piese table este actualizata prin functia updateTableMove() care se poate vedea mai sus. Imaginea piesei este luata de pe label-ul sursa si mutata pe cea destinatie.

### 3.2.2 Pre-conditii

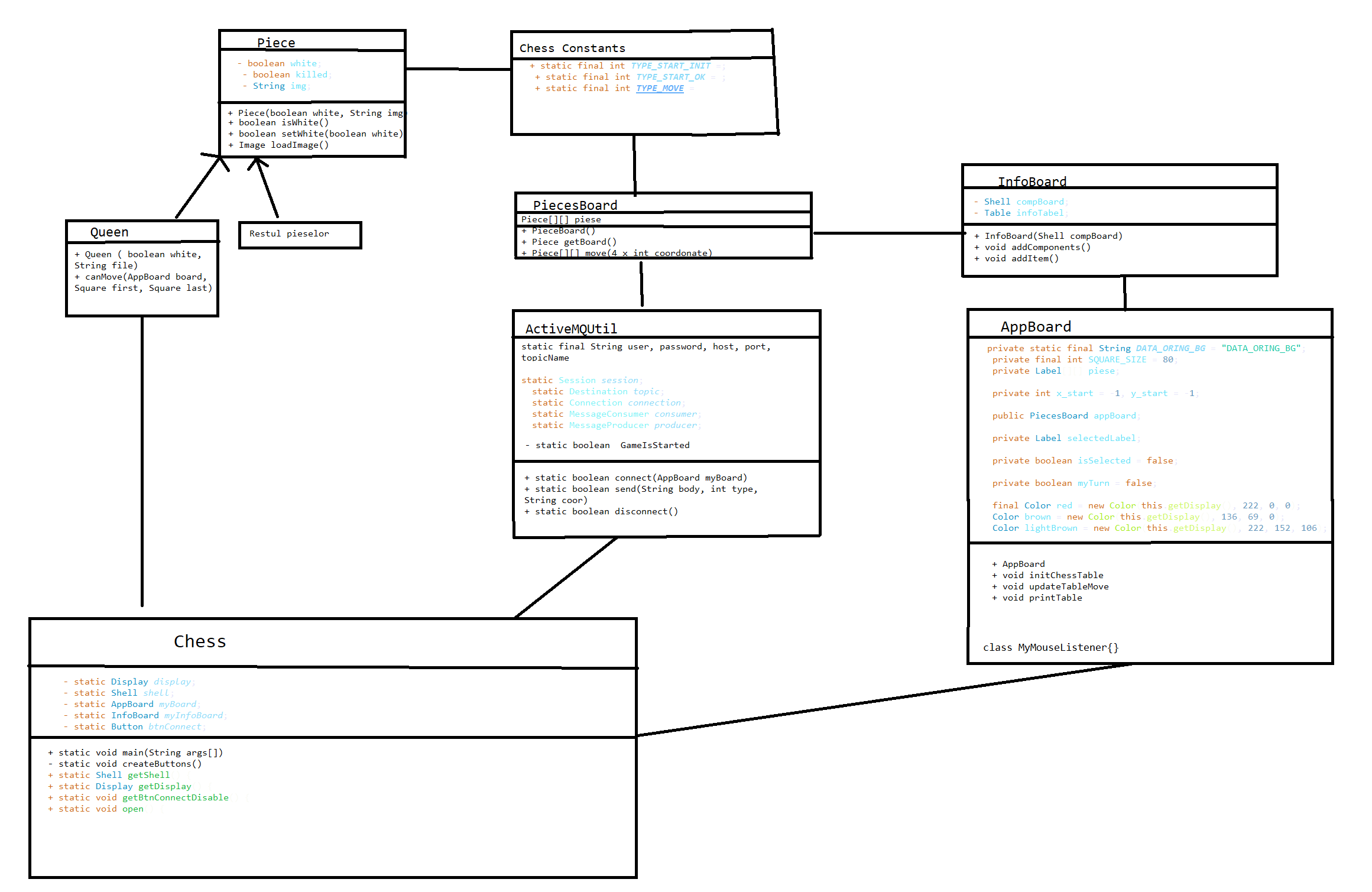
Utilizatorul trebuie doar sa deschida aplicatia ca table si piesele sa apara iar ca sa joace si tabla sa fie modificata in timp real trebuie sa inceapa jocul aceasta insemnand ca ambii jucatori sa aibe aplicatia deschisa sis a apese butonlul de start.

## Post-conditii

Jucatorul dupa ce deschide aplicatia va vedea tabla, urmand sa dea start joc si sa astepte dupa celalalt jucator.

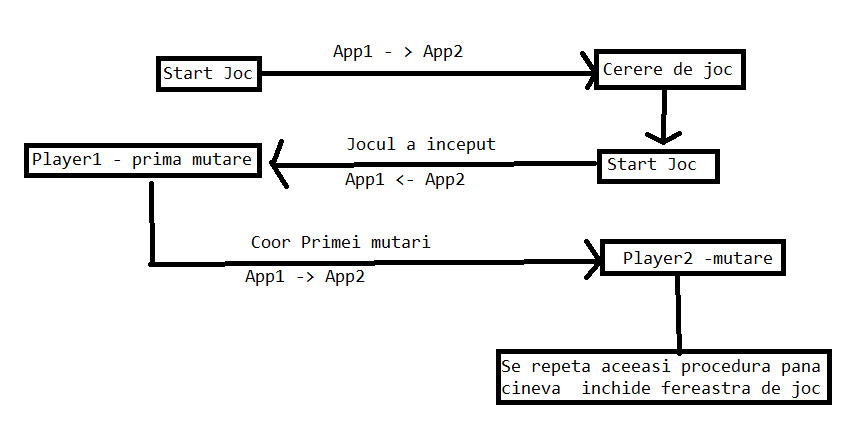
# Implementare

## Diagrama de clase



## Descriere detaliată

:



# Bibliografie

jms - Java Message Service - <https://www.oracle.com/technical-resources/articles/java/intro-java-message-service.html>

ActiveMQ - <http://activemq.apache.org/>

Piese de sah - https://commons.wikimedia.org/wiki/Category:PNG\_chess\_pieces/Standard\_transparent