Șah în Embarcadero C++

11.01.2023

Îndrumător:

dr. ing. Daniel Morariu

Student:

Vilău Gheorghe-Marian

224/1

Istoric Versiuni

Data	Versiune	Descriere	Autor
22/nov/2022	0.1	Realizarea interfeței grafice (tabla de joc și piesele încărcate manual ca imagine)	Vilău Gheorghe-Marian
06/dec/2022	0.2	Realizarea clasei de bază Piese și a claselor derivate pentru fiecare tip de piesă (Tura, Cal, Nebun, Regina, Rege, Pion)	Vilău Gheorghe-Marian
23/dec/2022	1.0	Implementarea mutărilor și analiza șirului primit prin EditBox. S-a introdus încă o clasă Tabla pentru a stoca mișcarile de pe tabla de joc	Vilău Gheorghe-Marian
31/dec/2022	1.1	S-a renunțat la partea de mutări prin clasa Tabla și s-a creat un vector pentru a stoca mișcarile pentru fiecare piesă în parte. Probleme la capturarea piesei	Vilău Gheorghe-Marian
1/ian/2023	1.2	Realizarea mutărilor prin EditBox cu ajutorul clasei Piese și rezolvarea problemelor din versiunea 1.1	Vilău Gheorghe-Marian
2/ian/2023	1.3	Încercarea implementării metodelor pentru verificarea mutărilor.	Vilău Gheorghe-Marian
5/ian/2023	2.0	Implementarea rețelei. S-a renunțat la transmiterea mutării prin EditBox, acesta fiind înlocuit cu fereastra Memo din motive de implementare a codului.	Vilău Gheorghe-Marian

Cuprins

1 Specificarea cerințelor software

1.1 Introducere

Proiectul conține două aplicații, realizate în Embarcadero C++, una pentru server și cealaltă pentru client, corespunzătoare celor 2 jucători. Ambii au acces la toate piesele de pe tabla de șah și pot controla ambele culori. Regulile jocului rămân la latitudinea jucătorilor, programul fiind unul cu scop aplicativ pentru câteva dintre principiile programării pe obiecte.

Embarcadero C++ Builder este o aplicație de dezvoltare software care permite programatorilor să scrie, să compile și să execute programe scrise în limbajul de programare C++. Acesta este produs de compania Embarcadero Technologies și face parte din gama de produse RAD Studio, care oferă instrumente de dezvoltare software pentru diferite limbaje de programare și platforme.

1.1.1 Obiective

Obiectivul principal al acestui proiect a fost implementarea unui simulator pentru un joc de șah care poate fi disponibil în rețea.

Obiective realizate:

- Creearea interfeței grafice (a tablei de joc, a pionilor și a componentelor auxiliare pentru controlul jocului, butoane și casete text);
- Crearea clasei de baza **Piese** și a claselor derivate corespunzătoare fiecărui tip de piesă: **Tura**, **Cal**, **Nebun**, **Regina**, **Rege**, **Pion**;
- Realizarea încapsulării, a moștenirii și a polimorfismului pentru clasele create;
- Legarea imaginilor pieselor de clasa **Piese**;
- Interpretarea șirului primit pe linia nouă a blocului Memo și analizarea lui ca o mutare pe tabla de joc cu codificarea descrisă în capitolul 1.1.2;
- Realizarea mutărilor și eliminarea piesei adverse dacă este cazul;
- Implementarea rețelei și funcționalitatea ei (aici se putea îmbunătăți un pic codul prin realizarea automată a mutărilor).

Obiective nerealizate:

- Verificarea mutărilor (la acest obiectiv am încercat ceva dar mergea numai pentru tură, nebun și regină);
- Ulterior verificării mutărilor apăreau alte 2 probleme. Prima este rocada ce ar trebui implementată prin butoane pentru fiecare tip de rocadă (mică sau mare) și pentru fiecare culoare. Cea de doua problemă e reprezentată de atacul pionilor și de „En Passant”. Aceasta ar putea fi implementată tot printr-un buton și o metodă în clasa **Piese**.
- Posibilitatea de a alege ce culoare dorește să fie serverul sau clientul și imposibilitatea de a muta piesele celeilalte culori;
- Implementarea unor condiții pentru șah și pentru șah mat;
- În momentul în care un pion ajunge în partea cealaltă a tablei să poată fi schimbat cu altă piesă.
- Un temporizator care să monitorizeze cât timp durează o mutare, asemănător competițiilor oficiale de șah.

1.1.2 Definiții, Acronime și Abrevieri

„En passant” este un termen utilizat în șah pentru a descrie o captură specială a pionilor. Acest lucru se întâmplă atunci când un pion se mută cu două pătrățele înainte de poziția sa inițială și un pion inamic este localizat pe o casă adiacentă la stânga sau la dreapta. Pionul inamic are opțiunea de a captura pionul „en passant” ca și cum ar fi mutat doar o casă.

Pentru realizarea acestui proiect s-au folosit anumite templateuri și convenții:

1. Pentru butoane s-a folosit următorul format: „btn” + textul afișat pe buton.
2. Pentru piese s-a folosit următorul format: „img” + nume piesă (Tura/ Cal/ Nebun/ Regina/ Rege/ Pion) + inițială culoare + numărul piesei în cazul în care sunt mai multe de același tip și de aceeași culoare.
3. Pentru membrul Culoare al clasei **Piese** s-a stabilit prin convenție să aibă 2 posibile valori: ‘N’ pentru piesele negre și ‘A’ pentru piesele albe.
4. În caseta de memo convenția de scriere a mutării are următorul format:
    a. Primul caracter este o literă de la ‘A’ la ‘H’ ce reprezintă coloana de unde pleacă piesa dorită;
    b. Al doilea caracter este o cifră între 1 și 8 ce reprezintă linia de unde pleacă piesa dorită;
    c. Al treilea caracter este o literă de la ‘A’ la ‘H’ ce reprezintă coloana unde ajunge piesa dorită;
    d. Ultimul caracter este o cifră între 1 și 8 ce reprezintă linia unde ajunge piesa dorită;

De asemenea, în inițializarea tablei de joc s-a ținut cont ca piesele să fie introduse în vector de la stânga la dreapta, de sus în jos.

În clasa **Piese** s-au folosit următorii membri:

- Xp, Yp care reprezintă coordonatele piesei, de tipul int. Ele sunt corespuzătoare proprietăților Left și Top. Această corespondență este vizibilă prin metoda Pozitie a aceleași clase;
- *Piesa de tipul TImage reprezintă, după cum sugerează și numele, imaginea unei piese de pe tabla de joc. Cu ajutorul acestui membru s-a realizat legătura între interfață și clasa **Piese**.
- Culoare de tipul char care reprezintă culoarea piesei.

1.1.3 Tehnologiile utilizate

Pentru realizarea interfeței grafice a acestui proiect s-a folosit inițial aplicația Canva Pro. Forma pieselor a fost corectată ulterior în Adobe Illustrator 2019.

Pentru implementarea codului s-a folosit mediul de programare Embarcadero C++ Builder 10.

Diagrama de clase a fost implementată în draw.io, iar descrierea detaliată în Larp.

Pentru a pune aplicațiile pe CD s-a folosit Nero Burning ROM 2015.

Documentația a fost realizată în Word 2016 și Google Docs.

1.2 Cerințe specifice

- Generarea tablei de joc și poziționarea pieselor;
- Analizarea mesajului primit ca mutare și decodificarea lui;
- Mutarea pieselor pe tablă și capturarea pieselor adversarului;
- Implementarea componentei de rețea și funcționalitatea ei (serverul apasă butonul „Trimite” pentru a trimite codificarea mutării către client, iar clientul trimite codificarea după ce a efectuat mutarea lui).

2 Generarea tablei de joc și implementarea componentei de rețea

2.1 Descriere

În jocul de șah piesele au o anumită poziție de început pe tabla de joc. Reperul principal este dat de faptul că regina își păstrează culoarea. Există și variante de șah în care regele are această calitate, dar regulile implementate în aplicație au fost făcute după varianta standard a jocului.

Componenta de rețea este esențială pentru a putea juca de pe 2 interfețe diferite, câte una pentru fiecare jucător. Când unul dintre ei va introduce o mutare și va apasa butonul „Trimite”, în cazul serverului, sau butonul „Muta”, în cazul clientului, această mutare va apărea și în caseta Memo a celuilalt partener de joc.

2.2 Fluxul de evenimente

2.2.1 Fluxul de bază

Pentru a porni rețeaua este necesară deschiderea aplicației „SAH_Server.cbproj”. Cel de-al doilea jucător se va conecta prin intermediul aplicației „SAH_Client.cbproj”.

Clientul se conectează la serverul aflat pe același calculator, deoarece acesta are introdusă în proprietăți adresa 127.0.0.1. Conexiunea între cele 2 părți se va face pe portul 3000.

Atunci când utilizatorul apasă pe butonul „Start” se creează vectorul de piese, Tabla, obiect de tip **Piese***. Se parcurge vectorul până la jumătatea lui (16) și se inițializează fiecare element cu câte o piesă cu ajutorul constructorului clasei corespunzătoare tipului piesei (Tura, Cal, Nebun, Regina, Rege, Pion). Acesta se realizează cu ajutorul tratării uniforme a masivelor eterogene. Constructurii folosiți au următorii parametri: imaginea piesei, coordonatele Left și Top și un caracter pentru culoarea piesei conform convenției 3 din capitolul 1.1.2.

Serverul trimite informația către client doar după ce se apasă butonul „Trimite”. Acesta va trimite pe socket, către conexiunea 0 (adică singurul nostru client), ultima linie din Memo, ce reprezintă mutarea scrisă de jucător. Clientul primește această mutare datorită evenimetului OnRead reprezentat de metoda ClientSocketRead. Pe caseta Memo a clientului va apărea mutarea serverului.

Clientul trimite mutarea către server atunci când se apasă butonul „Muta” după ce s-a analizat aceasta. Pe caseta Memo a serverului va aparea această mutare. Pentru a o efectua, serverul la randul lui apasă butonul „Muta”.

2.2.2 Pre-condiții

Utilizatorul trebuie să pornească aplicația server și, ulterior, aplicația client. Apasă butonul „Start” pentru a începe jocul. Atunci când introduce mutarea, în cazul în care e server, va trebui să apese pe „Trimite”, iar dacă e client pe „Muta”.

2.2.3 Post-condiții

Utilizatorul poate observa tabla de joc și piesele aranjate, inițial, după regulile standard ale jocului și poate interacționa cu ele prin intermediul ferestrei Memo și a celor 2 butoane, în cazul serverului, sau a butonului, în cazul clientului.

3 Analizarea mesajului primit și decodificarea lui

3.1 Descriere

Tabla de șah are liniile notate de la 1 la 8, iar coloanele de la A la H, asfel se cunoaște adresa oricărui pătrățel de pe suprafața de joc. Pentru a codifica o mutare se specifică piesa mutată, locul de unde pleacă și locul unde trebuie să ajungă.

În aplicația de față acest lucru a fost ușor simplificat, fiind necesare doar pozițiile de start și stop ale piesei, nu și numele ei.

3.2 Fluxul de evenimente

3.2.1 Fluxul de bază

Utilizatorul pornește una dintre aplicațiile server sau client și apasă pe butonul „Start”. În acest moment piesele vor fi poziționate corect pe tabla de joc (tura neagră și unul dintre pionii albi se aflau în afara tablei de joc, iar pionii negri nu au ordinea din convenție, de la A la H, pe linia 7, crescătoare din punct de vedere a indicelui).

Pentru a introduce o mutare utlizatorul va folosi caseta Memo. Pe un rând nou al acesteia el va scrie, conform convenției 4 din capitolul 1.1.2, el va scrie mutarea dorită.

În cazul serverului, pentru a trimite mutarea către client, utilizatorul trebuie să apese pe butonul „Trimite”, iar pentru a o efectua pe butonul „Muta”. Pentru client procesul este mai simplu, acesta doar apasă pe butonul „Muta” și va trimite către server mutarea după ce aceasta a fost deja analizată.

După ce se apasă butonul „Muta”, ultima linie din caseta Memo este stocată într-un string „sir” și trimisă către clasa **Piese** pentru analizare, cu ajutorul unui obiect numit „piesa”.

În metoda AnalizaMutare din clasa **Piese** se descompune acest șir, ținând cont de convenția din capitolul 1.1.2. O eroare posibilă apare în cazul în care șirul nu are exact 4 caractere sau, dacă are mai multe, primele 4 nu respectă tiparul. Astfel:

- Primul caracter este corespunzător primei coloane și va primi rezultatul int al metodei Coloana. În cadrul acestei metode se face, printr-un switch, conversia de la literă la cifră. Caracterul ‚A’ corespunde numărului 0, iar caracterul ‚H’ numărului 7.
- Al doilea caracter se leagă de prima linie. Numerotarea pe tabla de joc începe de jos în sus, dar matricea simulată a tablei începe de sus și se continuă în jos. De asemenea, în matrice indexarea se face de la 0. Pentru a obține întregul ce reprezintă prima linie, aceea de plecare a piesei, din 8 scădem numărul format prin convertirea caracterului la int.
- Caracterele de destinație au mod asemănător de formare, acestea fiind stocate în variabile cu indice 2.

Pentru că pătrățelele pe tabla de joc încep de la coordonatele 230 și 80, corespunzătoare Left și Top, iar un pătrat are latura de 48, atunci se va face conversia către coordonate numerelor extrase din șir:

```
*l1=80+*l1*48;
*c1=230+*c1*48;
*l2=80+*l2*48;
*c2=230+*c2*48;
```

S-au folosit pointeri pentru ca aceste valori să iasă modificate din metoda AnalizaMutare.

3.2.2 Flux alternativ - implementare cu libraria <sstream>

Această implementare ar fi fost un exemplu mult mai bun pentru programarea pe obiecte, dar pentru că vorbim doar de 4 caractere a fost mult mai simplu să le analizez caracter cu caracter, după cum am prezentat mai sus.

Pentru a parsa șirul din Memo utilizând librăria <sstream> puteam utiliza clasa stringstream care permite citirea dintr-un string ca și cum ar fi un stream. De asemenea, pentru a stoca datele ar fi trebuit folosit un vector, din librăria <vector>, deoarece nu se știe cât de lung ar putea fi șirul.

3.2.3 Pre-condiţii

Utilizatorul trebuie să:

- pornească una dintre aplicațiile server (SAH_Server.cbproj) sau client (SAH_Client.cbproj),
- apese butonul „Start”,
- introducă în caseta Memo mutarea dorită după regula expusă în capitolul 1.1.2. și să
- apese butonul „Muta”.

Singura problemă observată ce ar putea apărea aici ar fi să se pornească aplicația de client înaintea celei de server.

3.2.4 Post-condiţii

Pe interfață utilizatorul ar trebui să observe mutarea piesei în funcție de șirul introdus. Acestă metodă face o analiză internă a șirului în clasa **Piese** și va trimite prin parametri de referință coordonatele de plecare și de destinație a piesei.

În cazul în care șirul are mai puțin de 4 caractere, dar care corespund regulii atunci piesa va dispărea de pe tabla de joc (cel mai probabil este trimisă către un set de coordonate mai mare decât dimensiunea interfeței).

În cazul în care doar datele de plecare ale piesei sunt scrise corect acesta va ajunge în afara suprafeței de joc, în cele mai multe cazuri din simulări, undeva pe cadranul tablei de joc.

4 Efectuarea mutărilor pe tabla de joc

4.1 Descriere

În jocul de șah piesele au anumite reguli după care se mută pe tabla de joc, spre exemplu tura se poate muta doar pe aceeași linie sau aceeași coloană pe care se află, iar nebunul pe diagonale. După cum am specificat mai sus, pentru o mutare se notează piesa și pozițiile de plecare și destinației.

Pe baza prezumției de nevinovăție și a aspectului că pe o tablă de joc reală rămâne la latitudinea celor 2 jucători dacă respectă regulile clasice sau creează altele noi în cadrul acestei aplicații mutările nu sunt verificate, deși s-a încercat implementarea unor astfel de reguli.

4.2 Fluxul de evenimente

4.2.1 Fluxul de bază

După ce utilizatorul a introdus mutarea în caseta Memo și a apăsat pe butonul „Muta”, ultima linie este stocată într-un string „sir”. Prin obiectul piesa al clasei **Piese** aceasta este analizată. După apelul metodei AnalizaMutare variabilele lin1, col1, lin2, col2 se vor încărca, prin transmitere prin referință, cu coordonatele piesei, de plecare și de destinație.

Ulterior, se caută prin intermediul metodei Cautare dacă aceste coordonate corespund uneia dintre cele 32 de piese. Această metodă primește vectorul de piese și coordonatele dorite ca parametri. În implementarea ei s-a folosit o simplă parcurgere a șirului, verificându-se element cu element dacă s-a găsit piesa. În caz afirmativ metoda returnează poziția în vector unde se află aceasta, în caz contrar -1.

Metoda se folosește atât pentru poziția de start, rezultatul fiind stocat în variabila l, cât și pentru poziția de destinație.

Următorul pas este verificarea poziției de destinație. Dacă aici există o piesă atunci pentru aceasta se va apela metoda Captura din clasa **Piese**, cu un singur parametru, piesa respectivă. Dacă această piesă este neagră ea va fi trimisă la coordonatele 672 Left și 416 Top, undeva în paralel cu pionul alb H2, exact pe poziția unde se afla acesta înaintea apăsării butonului de start. Dacă piesa e albă ea va fi trimisă la 59 Left și 80 Top, paralel cu tura neagră A8, de asemenea, exact pe poziția dinaintea apăsării butonului de start.

Se verifică și poziția de pornire pentru a ne asigura că a fost introdusă corect (o piesă trebuie să aibă cel puțin un loc de plecare). Prin metoda getPiesa vom modifica coordonatele piesei, iar prin setX și setY vom modifica Xp și Yp pentru clasa **Piesa**. Aceste metode au fost necesare deoarece Xp, Yp și Piesa sunt membri protected.

4.2.2 Pre-condiții

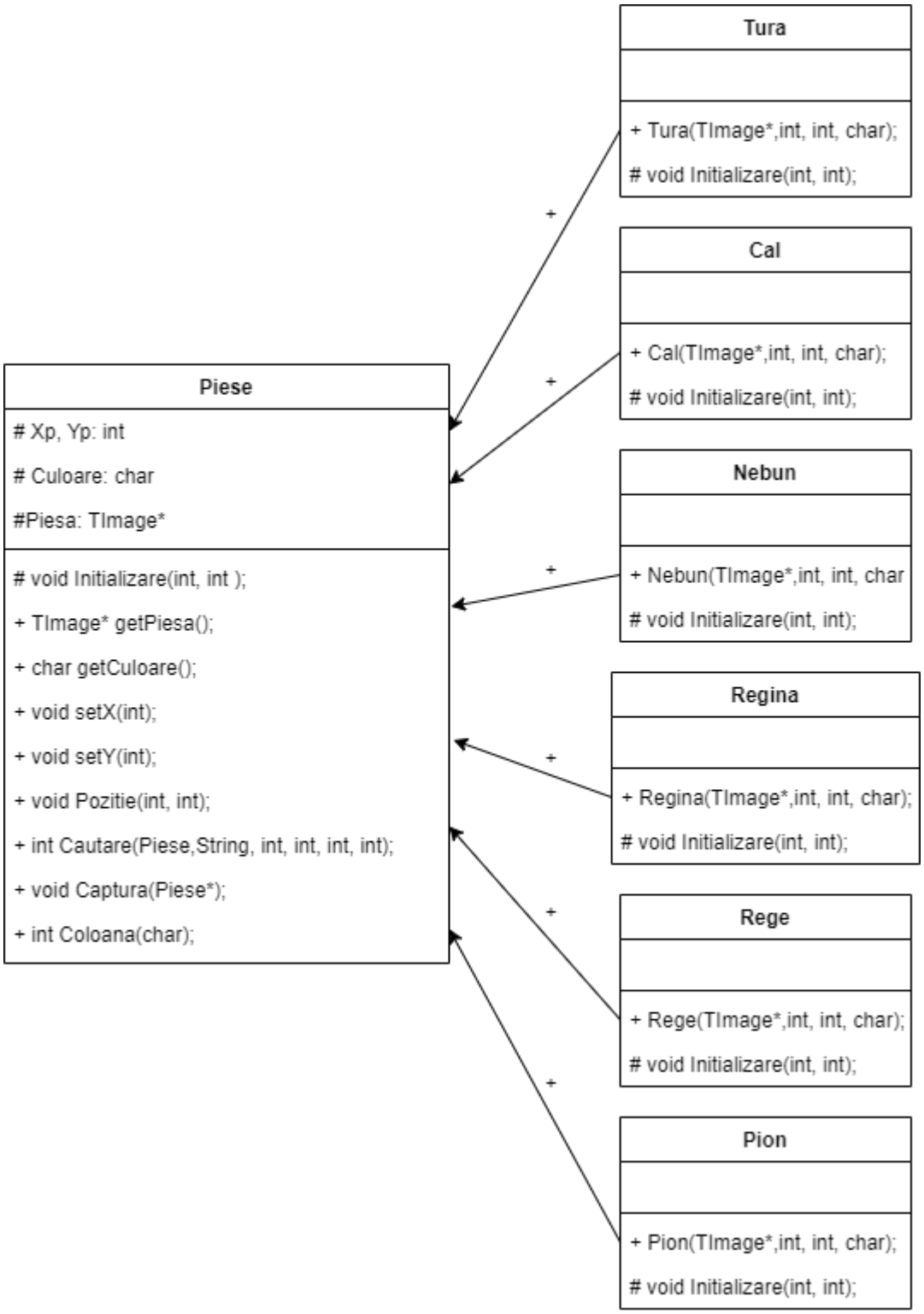
Aceleași pre-condinții ca la capitolul 3. Aceste funcționalități fac parte din funcționarea aceluiași buton „Muta”.

4.2.3 Post-condiții

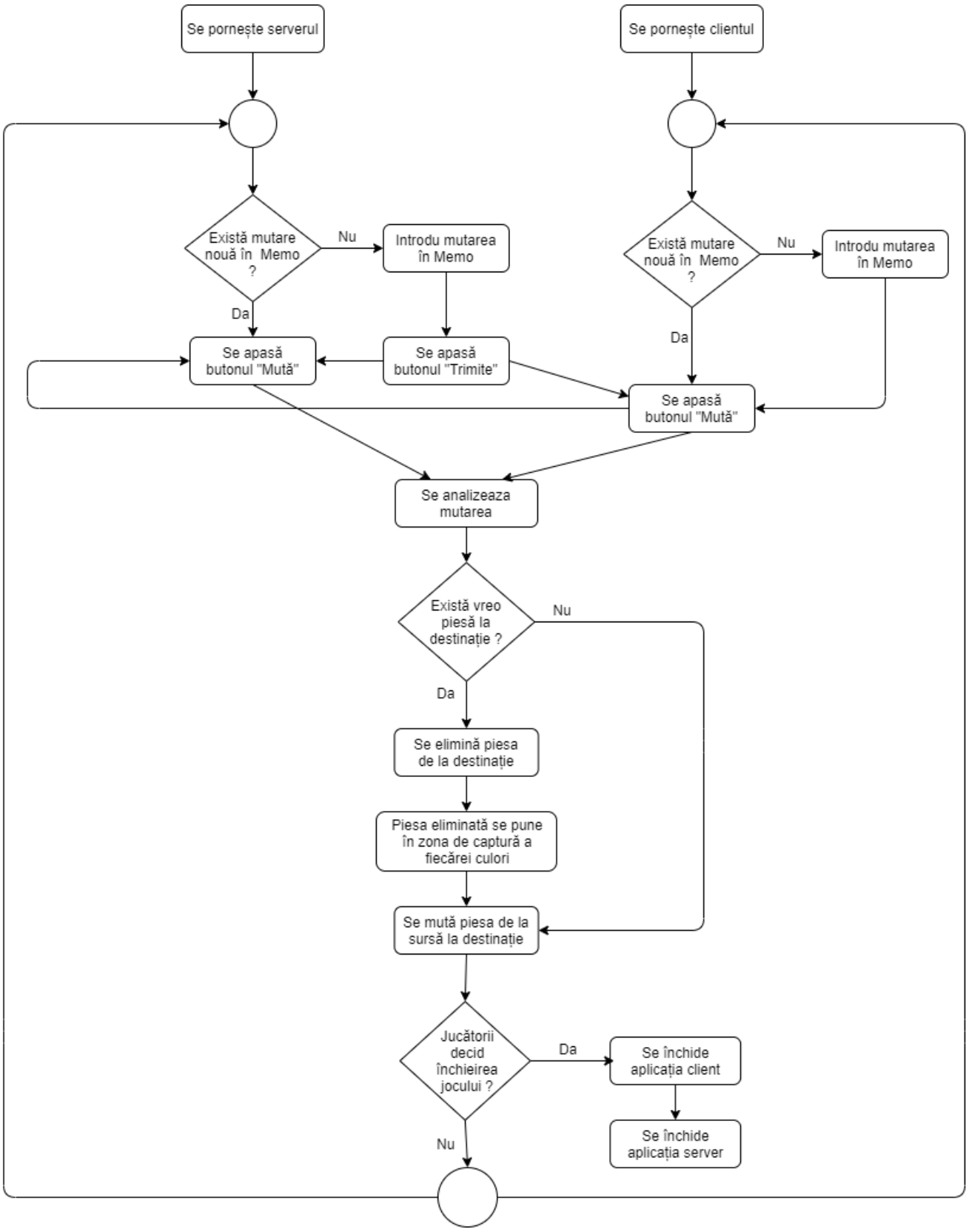
Aceleași post-condiții ca la capitolul 3. În versiunea anterioară, 1.1 au existat probleme la mutarea pieselor. Chiar dacă mutarea era introdusă corect, atunci când venea vorba de captură comportamentul era imprevizibil. Se captura fie piesa ce se afla la acea poziție, fie cea de la destinație. Aceste bug-uri au fost rezolvate începând cu versiunea 1.2.

5 Implementare

5.1 Diagrama de clase



5.2 Descriere detaliată



6 Bibliografie

Programare C++ Builder – Windows Controls http://www.functionx.com/cppbuilder/

Dezvoltare aplicație cu interfață grafică - http://docwiki.embarcadero.com/RADStudio/Tokyo/en/VCL

Programare orientată pe obiecte: principii – Macarie Breazu. –Sibiu: Editura Universității „Lucian Blaga" din Sibiu, 2002

Laboratoare – Conf. dr. ing. Daniel Morariu

C++ reference - https://en.cppreference.com/w/

Stack Overflow- https://stackoverflow.com/

Geek for geeks- https://www.geeksforgeeks.org/

Github - https://github.com/

ChatGPT - https://chat.openai.com/chat