**Mafia Game**

Inginerie Software

Realizat de:  
Morar Timotei Cristian

Tiuca Maria Daniela

Stroia Anton Calin

Contents

1. Introducere

1.1 Desfășurarea acțiunii

2. Implementare

2.1 Limbaj și IDE utilizat

2.2 Framework

2.3 Design Pattern

3. Diagrame

4. Instrucțiuni de utilizare

1. Introducere

1.1 Desfăsurarea acțiunii

Mafia Game este un joc de societate și strategie în care participanții sunt repartizați în diferite roluri, printre care se numără:

* **Killer** : Acest rol este de obicei asociat cu membrii mafiei, care trebuie să elimine treptat ceilalți jucători, în special pe detectiv, pentru a prelua controlul asupra jocului.
* **Detectiv**: Detectivul este un membru al orașului care încearcă să dezvăluie identitatea asasinilor. El poate investiga pentru a afla cine face parte.
* **Terapist**: Abilitatea terapistului este de a putea bloca un jucator in timpul convorbirii pe timpul zilei.
* **Doctor**: Este responsabil cu salvarea vieților în joc. El poate alege să protejeze pe cineva, oferindu-i șansa de a supraviețui în cazul unui atac al mafiei.
* **Cetățean**: Majoritatea jucătorilor vor fi cetățeni obișnuiți, care trebuie să își protejeze identitatea și să încerce să supraviețuiască. Ei pot să voteze pentru eliminarea unui jucător în fiecare rundă, încercând să scape de membrii mafiei.
* Primar: Are abilitatea de a decide daca votul pe timpul zilei se valideaza.

Jocul se desfășoară în runde, iar scopul este pentru oraș să identifice și să elimine membrii mafiei, în timp ce aceștia încearcă să preia controlul și să elimine cât mai mulți cetățeni sau membri importanți ai orașului. Comunicarea și strategia sunt esențiale pentru succesul echipei cetățenilor sau al mafiei, iar rezultatul depinde de modul în care jucătorii gestionează informațiile și interacțiunile lor.

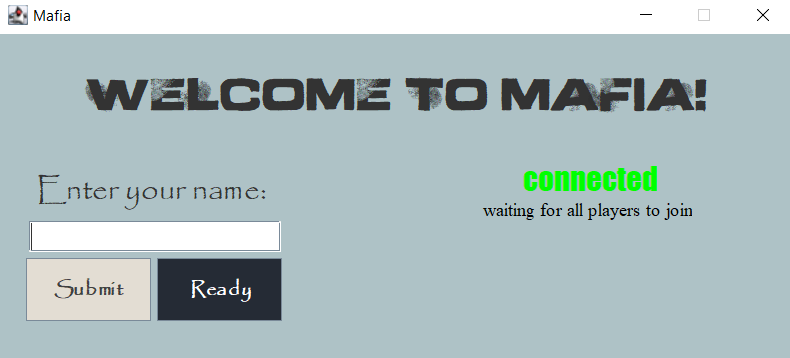


Figura 1.1 Lobby room waiting for other players to join

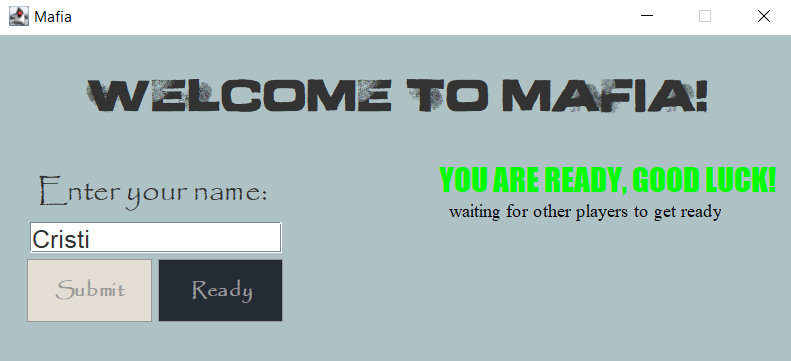


Figura 1.2 Lobby room waiting all players to get ready

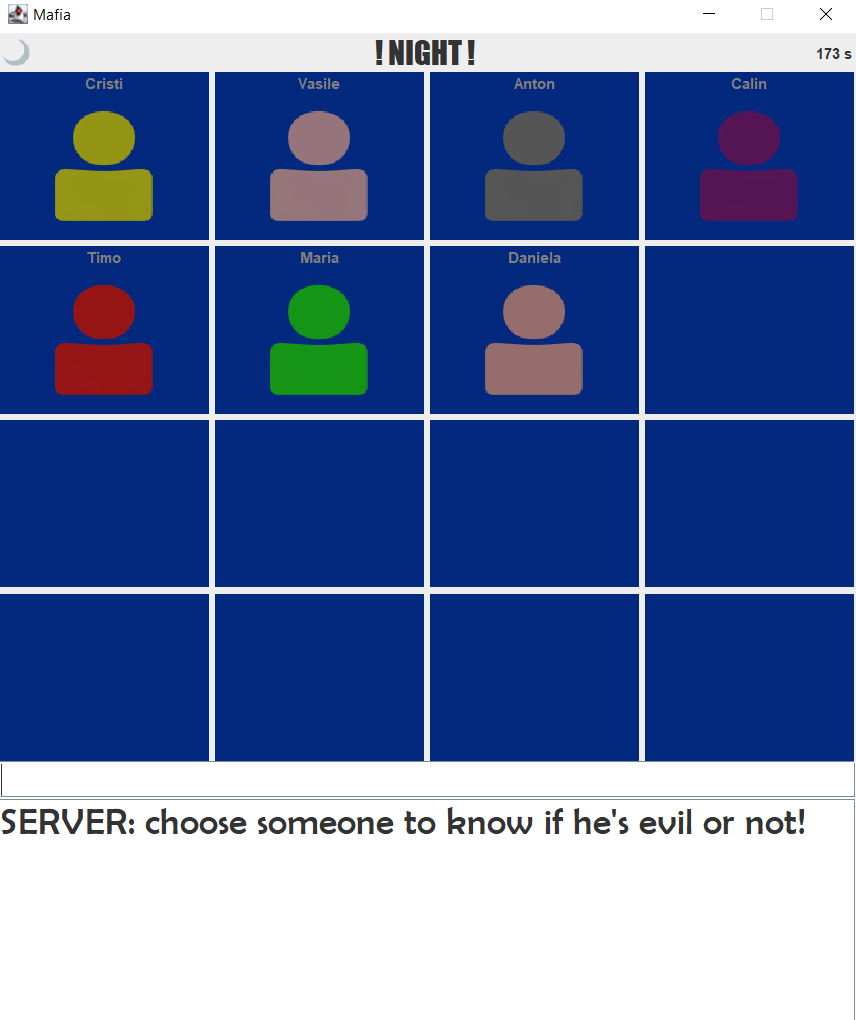


Figura 1.3 Game room waiting for players to do their action

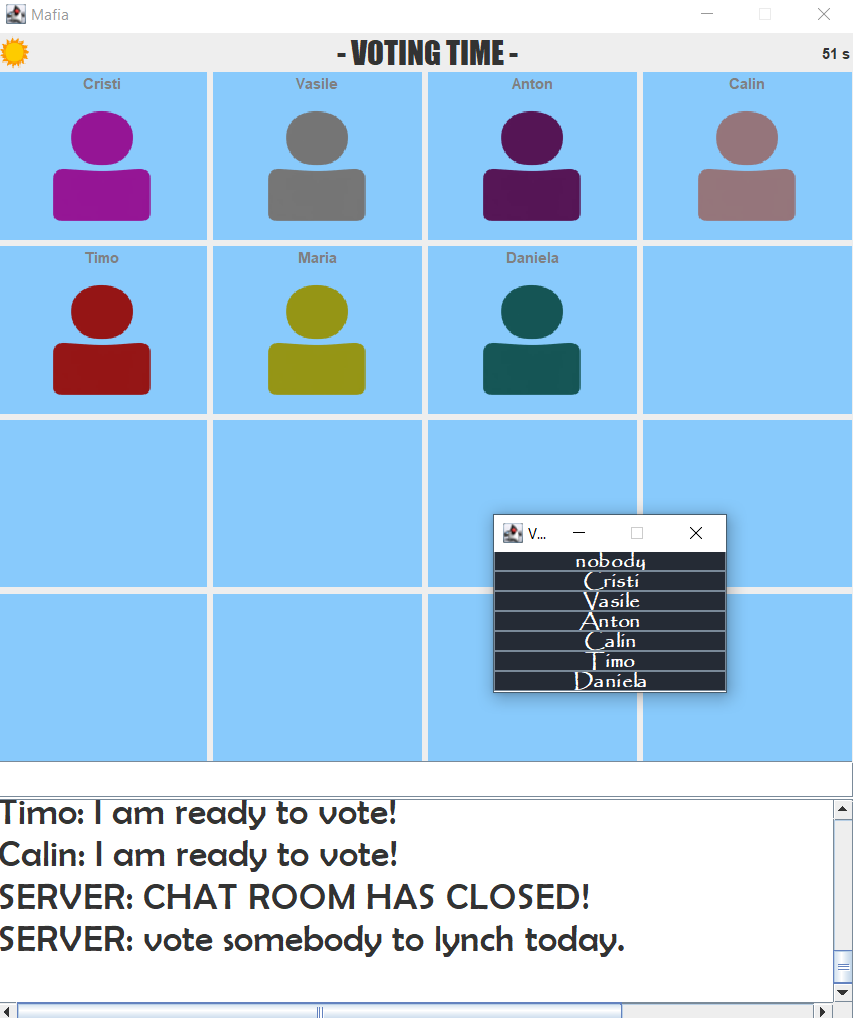


Figura 1.4 Game room during discussion time

Figura 1.4 Game room during voting time

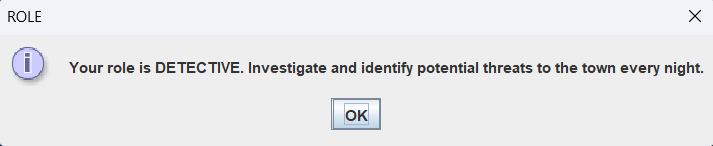


Figura 1.5 Every player is informed about his role.

1. Implementare

2.1 Limbaj și IDE utilizat

"Mafia-game" este un program implementat în limbajul de programare Java, folosind mediul de dezvoltare IntelliJ IDEA. Acesta reprezintă o simulare a jocului de societate "Mafia", unde jucătorii sunt împărțiți în două echipe - Mafioți și Locuitori - și își desfășoară acțiunile sub forma unui scenariu predefinit.

Caracteristici principale:

* Interfața grafică (GUI):

Utilizarea librăriilor Java Swing pentru a crea o interfață grafică interactivă și plăcută.

Ecrane pentru înscrierea și gestionarea jucătorilor, afișarea stării jocului și a acțiunilor disponibile.

* Gestionarea jocului:

Implementarea regulilor jocului "Mafia" pentru a asigura un joc autentic și echilibrat.

Crearea și distribuirea aleatoare a rolurilor (Mafiot, Cetățean, Detectiv, Doctor etc.) la începutul jocului.

Implementarea ciclurilor de zi și noapte, în care jucătorii iau decizii și își desfășoară acțiunile.

* Comunicarea între jucători:

Implementarea funcționalităților de chat sau mesagerie pentru a permite comunicarea între jucători în timpul jocului.

Trimiterea și primirea de mesaje anonime pentru a menține suspansul în timpul deliberărilor.

2.2 Framework

# Framework-ul Java Socket

Java Socket reprezintă un framework puternic și versatil pentru dezvoltarea aplicațiilor de rețea în limbajul de programare Java. Acesta oferă seturi bogate de clase și metode pentru comunicarea între procesele de pe diferite mașini, facilitând astfel dezvoltarea aplicațiilor distribuite și serviciilor de rețea.

### 1. Comunicare TCP și UDP

* Suport pentru protocoalele de comunicare TCP (Transmission Control Protocol) și UDP (User Datagram Protocol).
* Permite dezvoltatorilor să creeze aplicații care beneficiază de conexiuni stabile și sigure (TCP) sau de comunicare fără conexiune și cu latente reduse (UDP).

### 2. Crearea Serverelor și a Clienților

* Posibilitatea de a dezvolta atât servere cât și clienți, permitând interacțiunea bidirecțională între diferite dispozitive.
* Flexibilitate în configurarea și gestionarea conexiunilor pentru a susține diverse scenarii de utilizare.

### 3. Gestionarea Fluxurilor de Date

* Utilizarea claselor precum **Socket** și **ServerSocket** pentru manipularea fluxurilor de date dintre server și clienți.
* Oferă un model simplu și eficient pentru transmiterea informațiilor între dispozitive.

### 4. Asincronism și Fire de Execuție

* Posibilitatea de a implementa comunicare asincronă între clienți și servere, facilitând gestionarea mai multor conexiuni simultane.
* Utilizarea firelor de execuție (**Thread**) pentru a gestiona sarcini concurente și pentru a evita blocarea aplicației



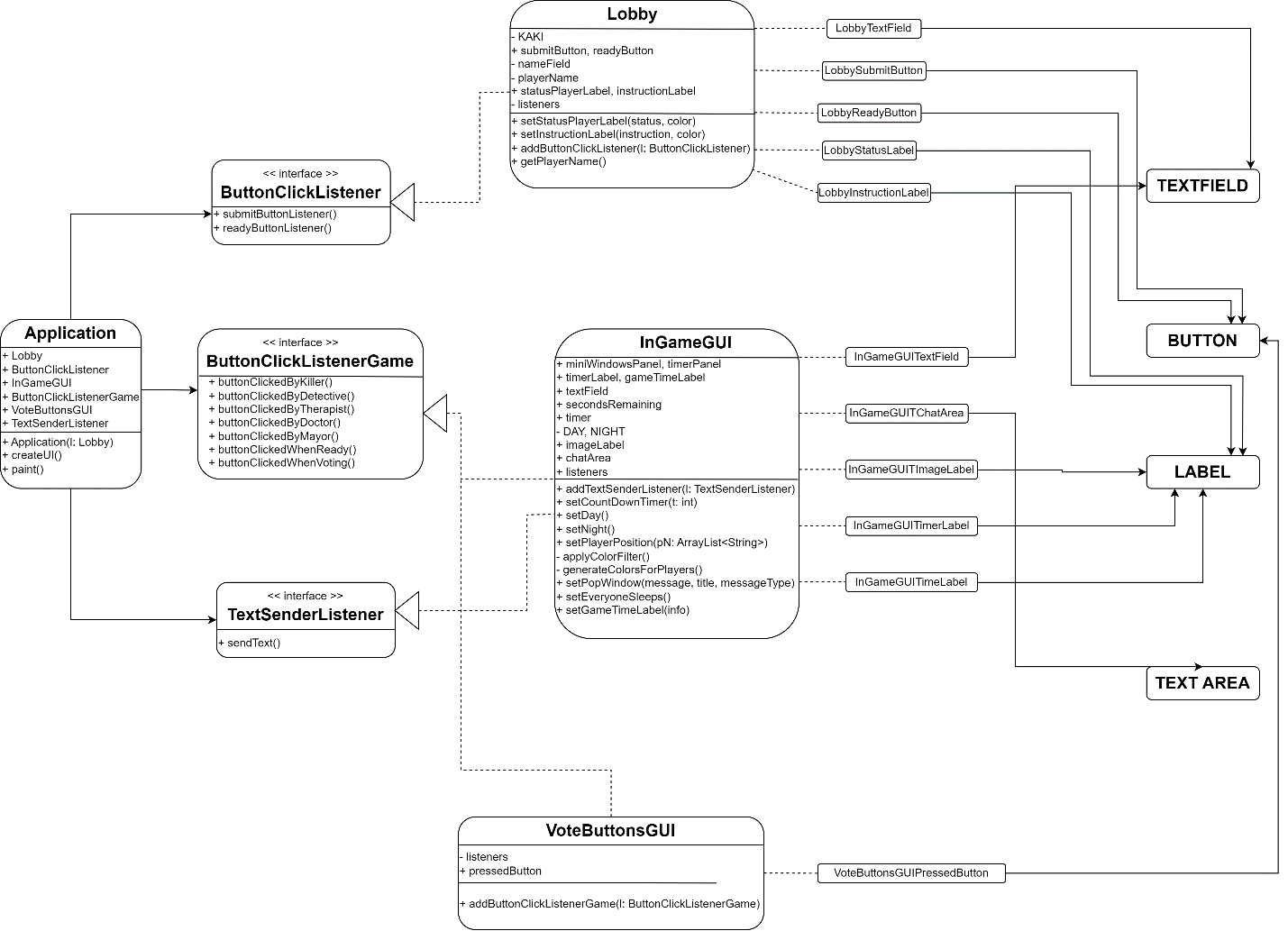
2.3 Design Pattern

Diagrama abstract factory pattern – Morar Timotei

Șablonul de proiectare abstract factory este un concept care încurajează crearea de familii de obiecte în mod coeziv și interschimbabil. Furnizează un strat de abstractizare pentru crearea obiectelor, permițând construirea sistemelor din componente variate și compatibile, fără a specifica clasele concrete ale acestora.

Șablonul abstractizează procesul de creare a obiectelor, permițând clienților să utilizeze interfețe de nivel înalt, fără a se preocupa de detaliile modului în care obiectele sunt instanțiate.

Prin bazarea pe interfețe în loc de clase concrete, șablonul promovează flexibilitatea. Devine mai ușor să se introducă noi familii de obiecte sau să se modifice cele existente fără a afecta restul sistemului.

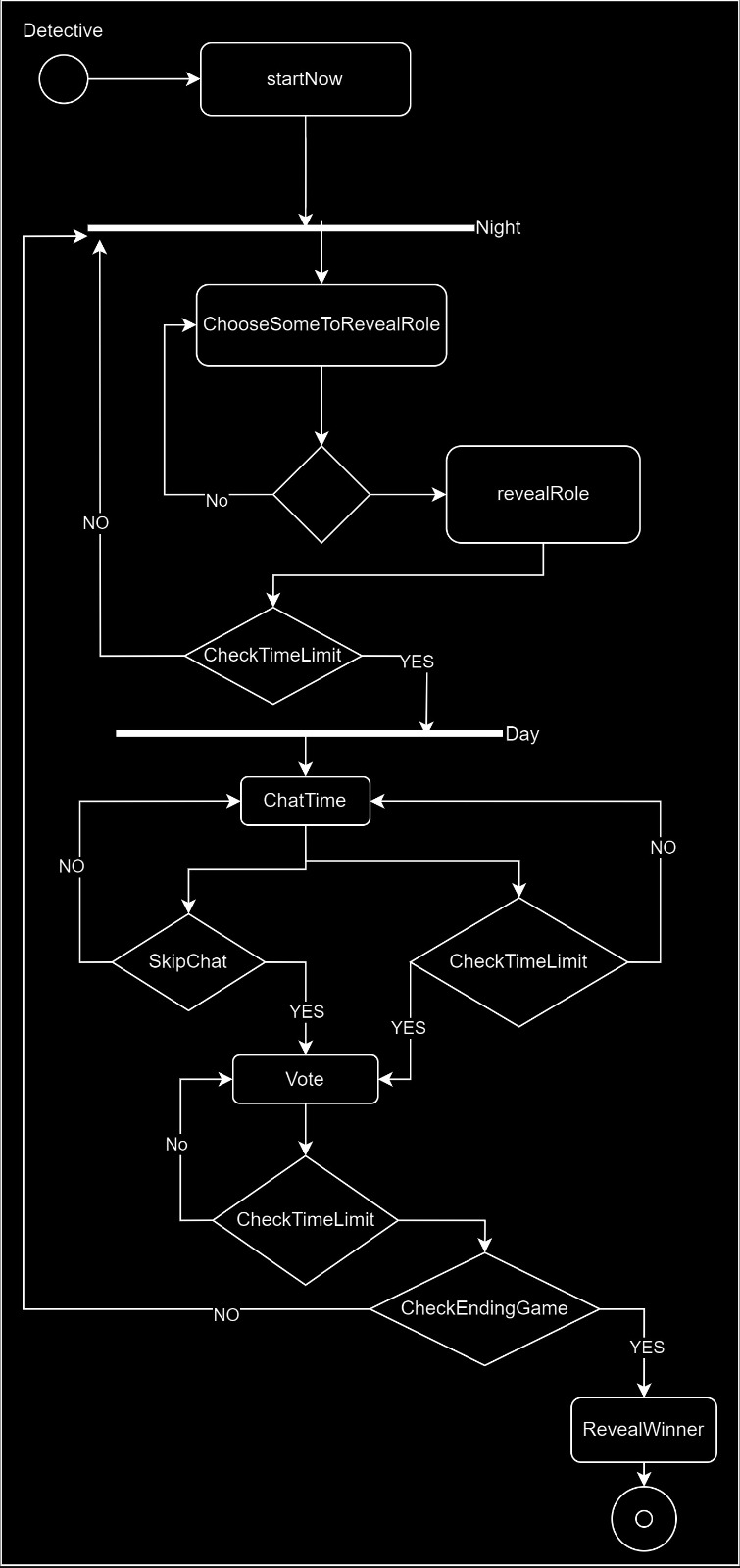
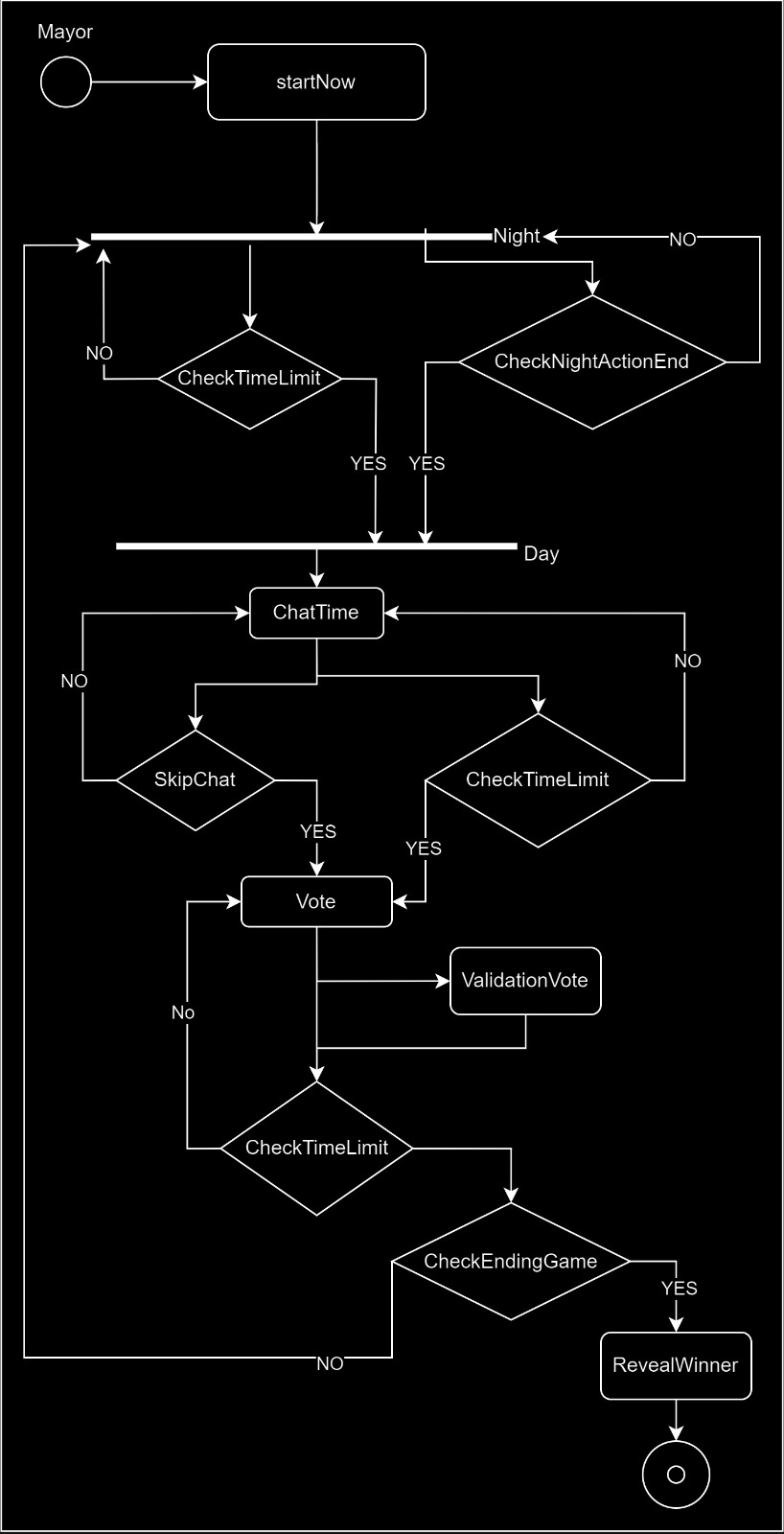


Diagrama player-role(mayor/detective) pattern – Tiuca Maria

Design patternul Player-Role din Java abordează problema modelării entităților care pot îndeplini diferite roluri în funcție de context. Un rol este definit ca un set particular de proprietăți asociate unui obiect într-un context specific, iar obiectul respectiv poate juca roluri diferite în contexte diferite. Întrebarea cheie a acestui design pattern este cum să modelezi cel mai eficient entitățile care pot schimba rolurile sau pot deține mai multe roluri.

Forțele care motivează utilizarea acestui design pattern includ dorința de a îmbunătăți încapsularea prin capturarea informațiilor asociate fiecărui rol într-o clasă separată. Este important să evităm multiplele moșteniri și să nu permitem unei instanțe să schimbe clasa. Aceasta se datorează faptului că dorim să evităm complexitatea adusă de multiplele moșteniri și să păstrăm o structură flexibilă a codului.

Design patternul Player-Role oferă o soluție eficientă pentru modelarea entităților într-un sistem software, permițându-le să își schimbe rolurile sau să dețină mai multe roluri într-un mod flexibil și modular. Acest lucru contribuie la îmbunătățirea încapsulării și menținerea unei structuri de cod coerente și ușor de înțeles.

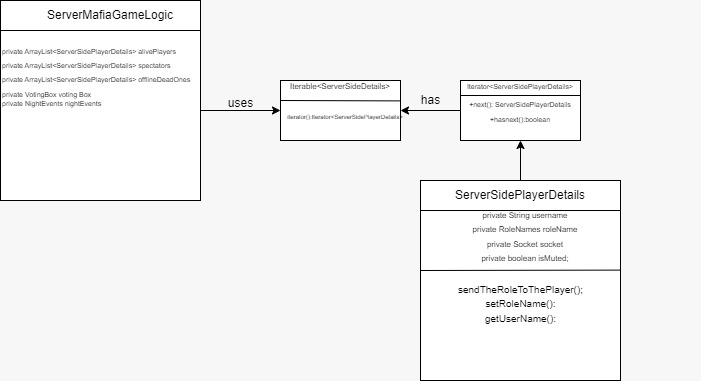


Diagrama iterator pattern – Stroia Calin

In Java atunci cand o clasa implementeaza interfata "Iterable", este necesar sa furnizeze o implementare a metodei iterator().Această metodă returnează un obiect de tip Iterator, care poate fi utilizat pentru a itera prin elementele colecției.În cazul proiectului, clasa ServerMafiaGameLogic implementează Iterable<ServerSidePlayerDetails> și furnizează o implementare pentru metoda "iterator()".

Această metodă returnează un iterator pentru lista de jucători vii. Astfel, clasa ServerMafiaGameLogic respectă design pattern-ul "Iterator".De asemenea, se observă că metoda givePlayersTheirRoles() utilizează iteratorul pentru a parcurge jucătorii vii și a le atribui roluri. Aceasta este o utilizare specifică a design-ului de Iterator.

3. Diagrame

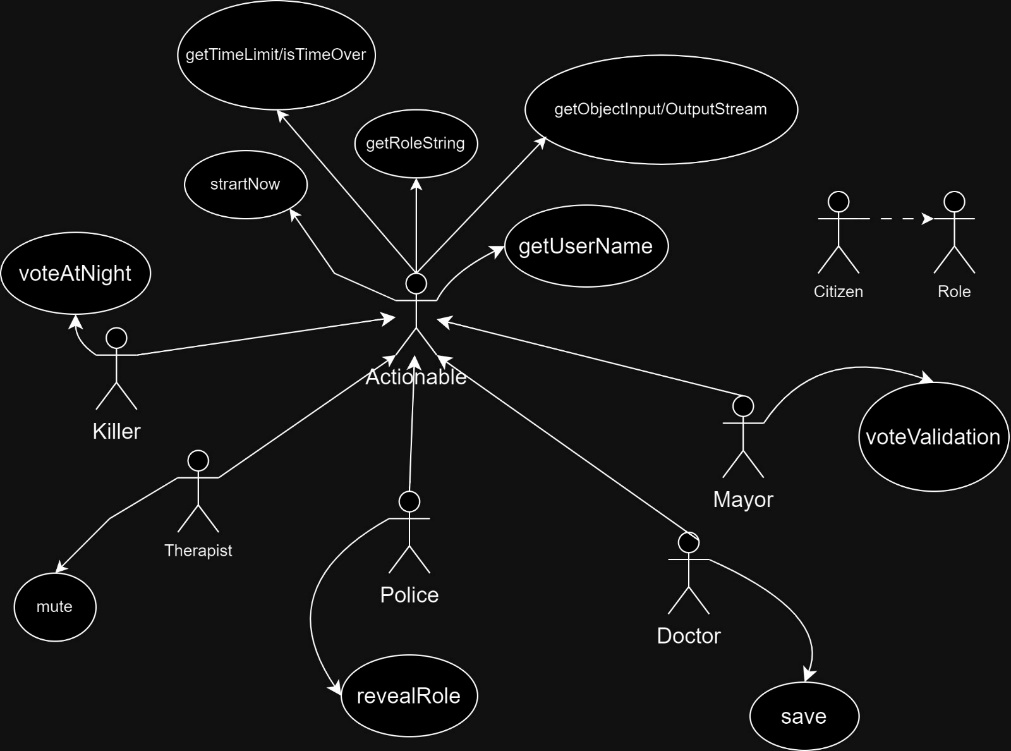


Diagrama de USE-CASE SCENARIO

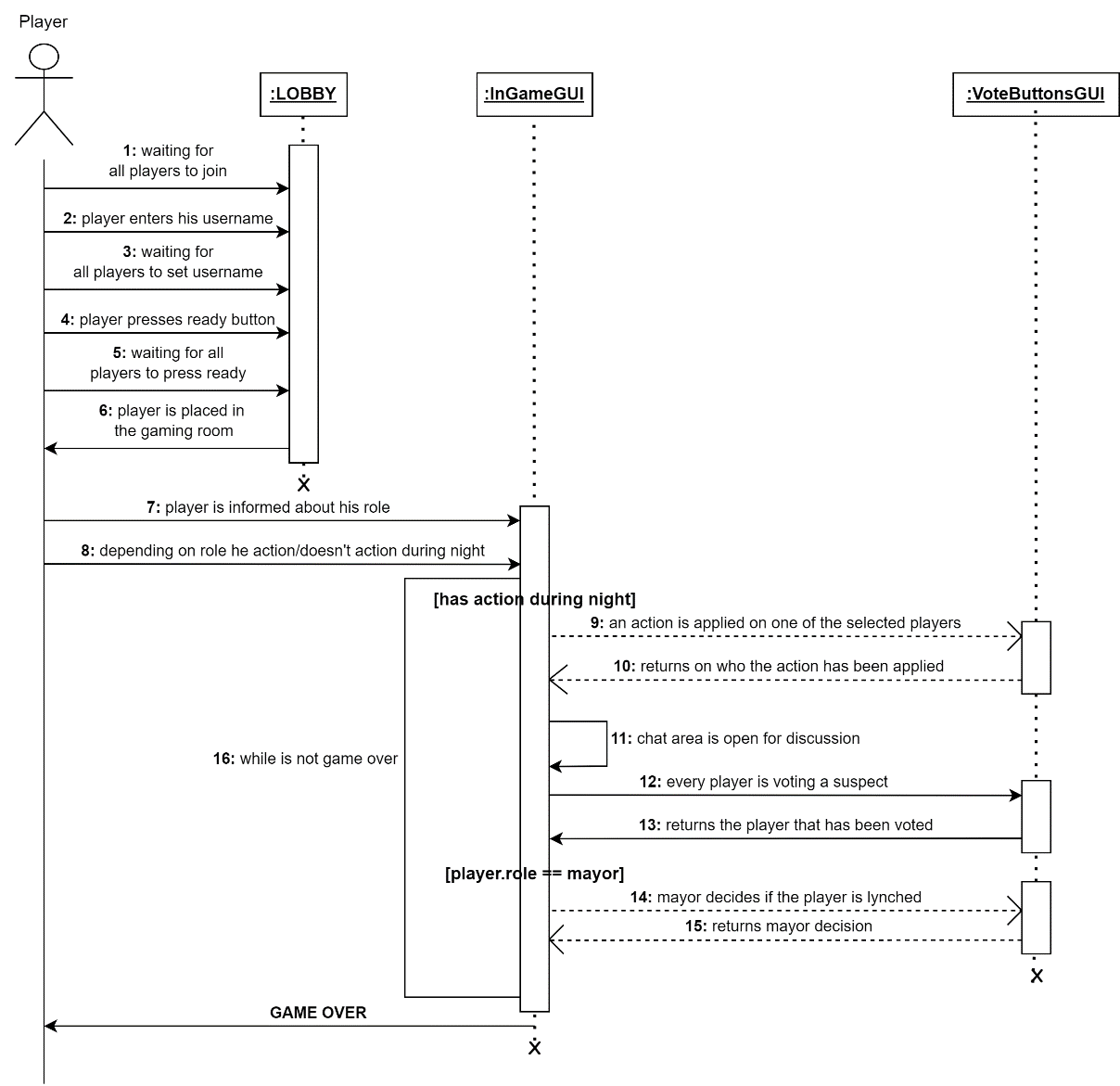


Diagrama de secventa pentru GUI

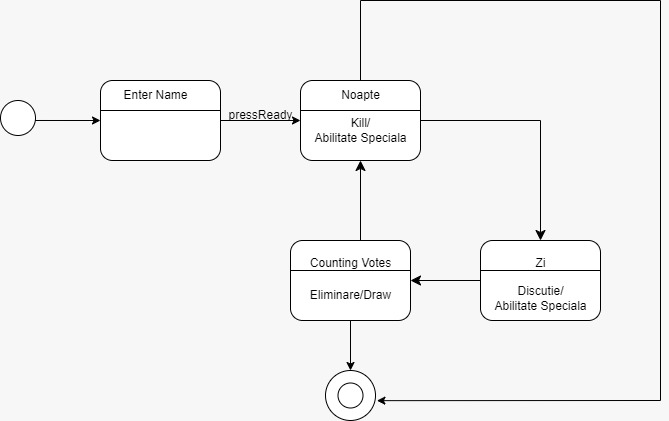
Morar Timotei

A diagram of a company

Description automatically generated

Diagrame player-role

Tiuca Maria



A black and white diagram

Description automatically generated

Diagrama de stare

Stroia Calin

4. Instrucțiuni de utilizare

Pentru a începe acest joc captivant, primul pas esențial este stabilirea numărului de jucători. Această etapă este crucială, deoarece va influența echilibrul și dinamica viitoare a partidei. Numărul de jucători trebuie să se încadreze în intervalul specificat, adică între 3 și 16.

Odată stabilit numărul de participanți, urmează să pornim serverul introducând acest număr. Serverul este centrul de control al jocului și asigură conectivitatea între jucători. Este important ca toți participanții să se conecteze la aceeași rețea de internet.

Fiecare jucător este apoi invitat să se conecteze la server și să își introducă numele într-un câmp specific. Acest pas este esențial pentru a personaliza participarea fiecărui jucător la joc. După ce au completat această informație, aceștia trebuie să confirme prin apăsarea butonului "Submit".

Odată ce toți jucătorii s-au înscris, următorul pas este inițierea jocului prin apăsarea butonului "Ready". Această acțiune declanșează distribuirea aleatoare a rolurilor, care includ Mafioți, Detectivi, Doctori și Terapeuți. Această fază a jocului este crucială pentru crearea unei atmosfere imprevizibile și echitabile.

Mecanica jocului este determinată printr-un ciclu repetitiv de noapte și zi. În timpul nopții, Mafia decide cine va fi omorât, Detectivul investighează un jucător, Doctorul alege pe cine să salveze, iar Terapeutul blochează chat-ul unui participant. Dimineața aduce cu sine actualizarea numărului de jucători și începerea sesiunii de vot, unde cetățenii decid cine este criminalul.

Jocul se poate încheia prin victoria cetățenilor în urma demascării crimininalului sau poate sa continue prin repetarea acestui ciclu până când numărul de mafioți devine mai mic sau egal decât numărul de cetățeni, moment în care este determinat câștigătorul și anume “Mafia”.