LUDO – DOCUMENTAȚIE

Tăbăcaru Rareș Andrei  
Tret Elena  
Zaharia Cătălin Constantin

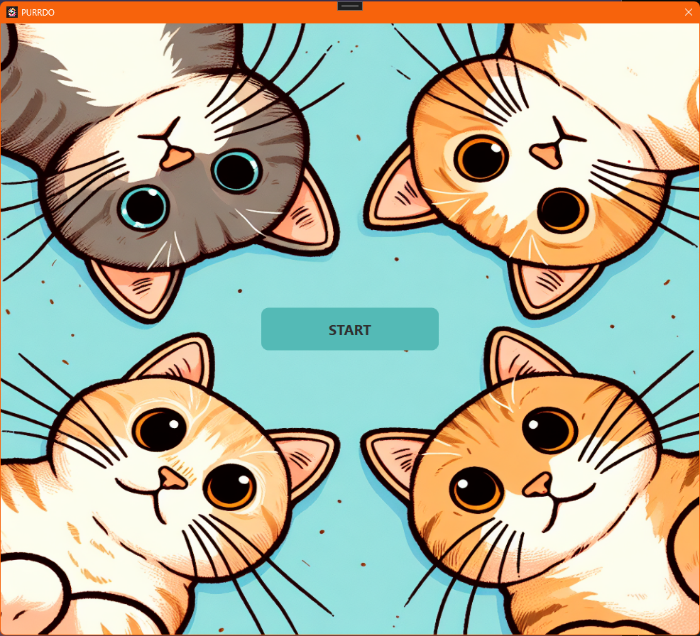
1. Prezentare generală

**Introducere:**

Proiectul Ludo aduce jocul clasic, cu același nume (sau cu denumirea de „Nu te supăra frate”) din varianta fizică, în lumea digitală. Jocul a fost creat utilizând limbajul C#, cu tehnologia WPF (Windows Presentation Foundation) utilizând arhitectura MVVM (Model-View-ViewModel).

**Instrucțiuni de joc:**

A screenshot of a game

Description automatically generated**** La pornirea jocului va apărea pagina de start. Prin apăsarea butonului de START se va ajunge la LOBBY unde jucătorii vor completa numele cu care vor să apară pe parcursul jocului.

Numărul minim de jucători este 2, iar numărul maxim este 4. Dacă nu se completează câmpurile cu minim 2 nume, jocul nu va putea începe. După ce se completează numărul necesar de jucători se apasă pe butonul de PLAY. Apoi se va ajunge la tabla de joc. Jucătorilor li se va atribui o culoare aleatoare ordinea în care dau cu zarul fiind: albastru, galben, verde, roșu.

A screenshot of a game

Description automatically generated

A screenshot of a computer

Description automatically generated

Pentru a scoate un pion, jucătorul trebuie să dea valoarea 6 cu zarul. Dacă va da altă valoare (și nu are niciun pion scos) va trebui să aștepte până îi vine din nou rândul. De fiecare dată când s-a dat cu zarul, pionii care se pot muta vor fi evidențiați. Dacă nu există o mutare posibilă, va apărea un mesaj.

A screenshot of a game

Description automatically generated A screenshot of a game

Description automatically generated

A screenshot of a game

Description automatically generated Pionii de aceeași culoare pot fi poziționați pe aceeași poziție pe tabla de joc. Pionii de culori diferite pot fi așezați pe acceași poziție doar daca se afla pe cele 4 zone colorate (cele în care ies prima oara pionii când sunt scoși pe tabla de joc). Dacă nu se află pe acea zonă, mutarea nu poate fi realizată, apărând un mesaj.

A screenshot of a game

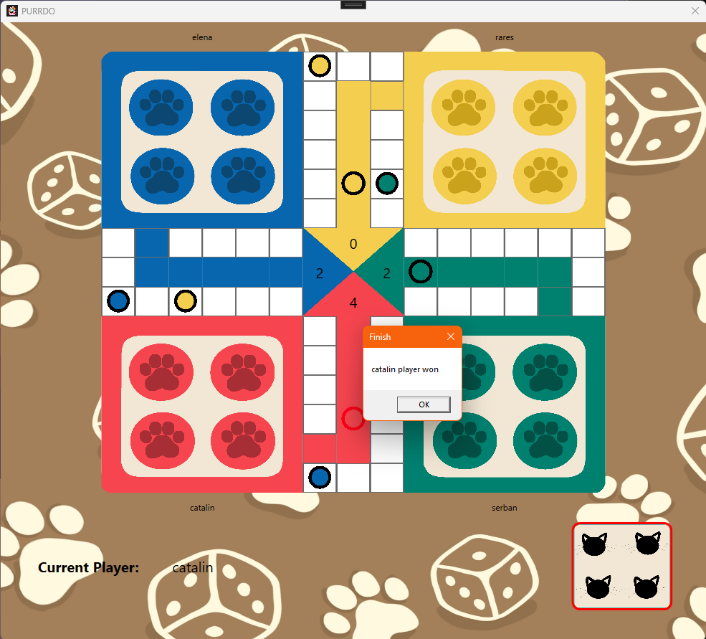
Description automatically generated

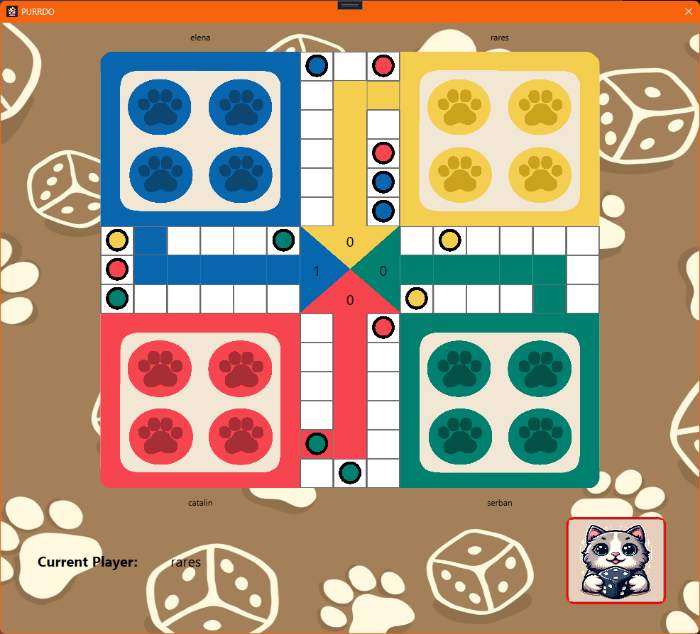
A screenshot of a game

Description automatically generatedA screenshot of a game

Description automatically generated

Scopul jocului este de a aduce toți cei 4 pioni în bazele din centrul tablei. Odată ce un jucător aduce toți pionii în centru, jocul se va termina și pagina va fi redirecționată la lobby unde se poate începe un nou joc.





1. Design Patterns

**OBSERVER**

A screenshot of a computer

Description automatically generated

Interfața „IgameEngineObserver” definește observatorii care vor fi notificați despre diferite evenimente care au loc în cadrul clasei GameEngine. Metodele din această interfață sunt responsabile pentru a notifica observatorii despre diverse evenimente care au loc în joc, furnizând informații relevante despre aceste evenimente (cum ar fi valoarea zarului, mutările pionilor, schimbările de scor etc.).

Interfața „IgameEngineSubject” definește subiectul care este responsabil pentru gestionarea și notificarea observatorilor. Metodele Attach și Detach sunt folosite pentru a atașa și detașa observatori de subiect. Prin Attach, observatorii sunt înregistrați pentru a fi notificați despre evenimente, în timp ce Detach este folosit pentru a elimina înregistrările observatorilor.

Utilizarea acestor interfețe înseamnă că un obiect care implementează IGameEngineSubject va putea să se înregistreze și să notifice observatorii (IGameEngineOberver) despre schimbările și evenimentele relevante care au loc în cadrul jocului.

**SINGLETON**

**A computer screen with text on it

Description automatically generated**

Acest design pattern a fost implementat pentru a crea cate o singură instanță pentru anumite obiecte. De exemplu: pentru fereastra „Main Window” s-a utilizat acest design pattern pentru a asigura existența unei singure insanțe a ferestrei pe parcursul întregului ciclu de viață al aplicației.

**DEPENDENCY INJECTION**

**A computer screen shot of a program

Description automatically generated**

Dependency Injection este un pattern de design care facilitează gestionarea dependențelor între componente.

„ServiceCollection” este utilizat pentru a înregistra dependențele (serviciile) folosite în aplicație. De exemplu, „AddSingleton” este folosit pentru a înregistra servicii ca obiecte singleton.

Atunci când un serviciu este solicitat prin „GetService” sau prin crearea directă cu „CreateService”, dependency injection este utilizat pentru a furniza dependențele necesare pentru a instanția acel serviciu.

Atributul „InjectAttribute” este folosit pentru a marca constructorii pentru care se dorește o gestionare specială a dependențelor. În cazul în care constructorul este marcat cu acest atribut, ServiceCollection îl identifică și îl utilizează pentru crearea serviciului.

1. Unit Tests

A screenshot of a computer

Description automatically generated

Testele au fost realizate pentru clasele Board, GameEngine (folosind teste simple) și Observer (folosind tehnica de mocking). Acestea sunt organizate ca fiind TestClass (pentru clase), TestInitialize (pentru a verifica dacă se inițializează corect obiectele) și TestMethod (pentru metodele prezente în clasele respective).

A screen shot of a computer program

Description automatically generated

Fiecare metodă a fost verificată, luându-se în considerare cât mai multe cazuri posibile. Spre exemplu: se poate observa verificarea metodelor „ReleasePawnFromBase” (care scoate un pion din bază și îl pune pe tile-ul inițial corespunzător culorii jucătorului) și „MoveInPlayPawn” (care mută un anumit pion un număr de pași). La început se crează tabla de joc, apoi se așează pionul pe poziția de start (0), urmând ca apoi să se realizeze un anumit număr pe pași (2), în final ajungând pe zona cu indicele 2.

A screen shot of a computer program

Description automatically generated

Testele cu mocking permit izolarea componentelor pe care dorim să le testăm (în cazul de față, funcționalitatea de notificare a observer-ului) fără a depinde de implementarea reală a acestor componente. De exemplu, s-a testat dacă atunci când un anumit eveniment are loc în joc (cum ar fi o mutare sau o schimbare pe tablă), observatorii sunt notificați corespunzător.

1. Diagrame

**DIAGRAMA CAZURILOR DE UTILIZARE**

A diagram of a company

Description automatically generated

Ca și actori avem jucătorul și jocul în sine. Cazurile de utilizare sunt reprezentate de regulile jocului. „Pornește jocul” include „Denumire jucători” care de asemenea include „Deschidere joc” deoarece una nu se poate realiza fără cealaltă (de exemplu: jocul nu poate începe dacă jucătorii nu și-au introdus numele). Avem și câteva cazuri speciale; atunci când se dorește mutarea unui pion există momente în care unele mutări sunt invalide (de exemplu: atunci când un jucător are toți pionii în bază, acesta trebuie să dea valoarea 6 cu zarul; dacă se dă o valoare diferită de 6, atunci mutarea devine invalidă).

Actorii sunt în aproape aceleași relații cu cazurile de utilizare deoarece dacă jucătorul ia o anumită decizie, jocul va face ca acțiunea respectivă să fie posibilă (de exemplu: jucătorul apasă pe butonul de aruncare cu zarul, însă jocul face posibilă returnarea unei valori cuprinse între 1 si 6 inclusiv).

**DIAGRAMA DE CLASE**

A diagram of a computer program

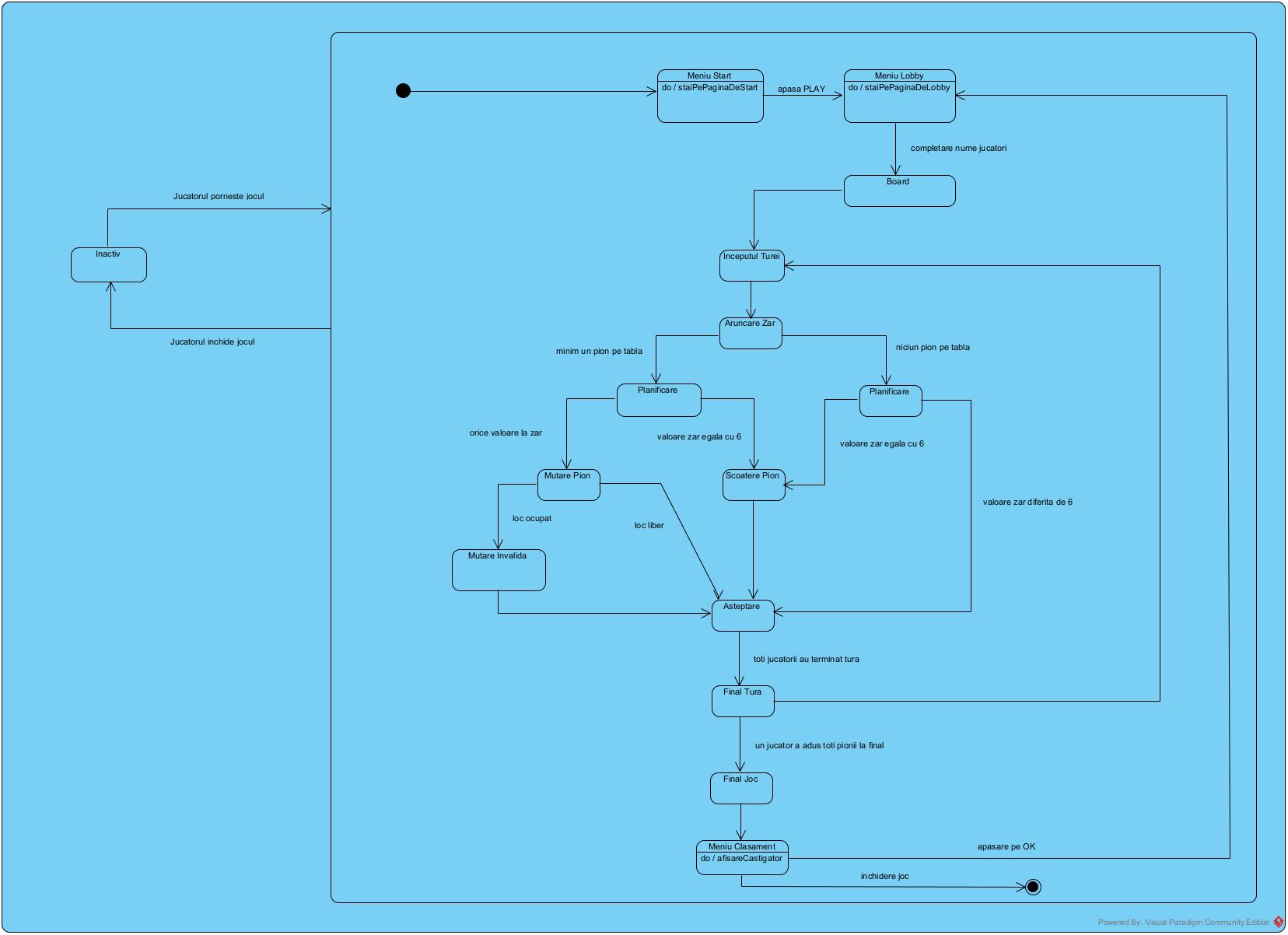
Description automatically generated with medium confidence

Între clasele Player și Pawn există o relație de agregare deoarece una este compusă din cealaltă, dar cea din urmă poate exista și independentă. Aceeași relație se aplică și între Pawn și Tile deoarece un Tile poate exista și fără să aibe un Pawn pe el.

Între clasele Board și Tile există o relație de compoziție deoarece entitățile depind în totalitate unele de altele. Un Board nu poate exista dacă nu este altcătuit din mai multe Tile-uri.

Între enumerații si clasele unde sunt utilizate (PlayerColor – Player, PawnState - Pawn) există o relație de asociere, iar între interfețe este prezentă moștenirea folosindu-se relația de generalizare.

**DIAGRAMA DE STARI**



Această diagramă reprezintă stările și tranzițiile prin care trece jocul de Ludo în timpul unei sesiuni de joc. În interiorul unor stări se pot afla și unele activități, spre exemplu: la starea Meniu Start apare do/staiPePaginaDeStart care ar avea următoarea semnificație „cât timp obiectul stă pe starea Meniu Start se execută staiPePaginaDeStart”.

La final, jucătorul poate readuce jocul în starea de Meniu Lobby pentru a mai juca o rundă sau poate să închidă jocul, moment în care se revine la inactivitate.

**DIAGRAMA DE ACTIVITĂȚI**

A diagram of a diagram

Description automatically generated

Această diagramă reprezintă secvențele de acțiuni și decizii pe care jucătorii le pot lua în timpul jocului. De la o acțiune la alta poate exista o tranziție (de exemplu: săgeata reprezentată de la „Adăugare nume” și „Pornire joc”), dar și decizii (simbolul romb cu descrieri pe liniile săgeților). Un exemplu de decizie este prezentă la acțiunea de „Planificare” unde jucătorul are pioni pe tabla de joc și a dat valoarea 6 cu zarul. Acesta are de ales între a muta un pion deja existent pe tablă sau a scoate un pion din bază.

Simbolul romb fără descriere reprezintă reunirea sau întoarcerea la o etapă comună, după ce s-au realizat două sau mai multe bifurcații, astfel încât să se continue cu acțiuni comune sau în paralel, după deciziile luate anterior.

**DIAGRAMA DE SECVENȚĂ**

A screenshot of a diagram

Description automatically generated

Această diagramă ilustrează din punct de vedere cronologic transmiterea de mesaje între obiecte (de exemplu, poate ilustra cum un jucător aruncă zarul și cum sistemul răspunde cu rezultatul obținut.).

Fragmentele alternative (segmentele notate cu denumirea „alt”) reprezintă deciziile pe care jucătorul trebuie să le ia. Spre exemplu: după aruncarea cu zarul jucătorul trebuie să decida ce mutare vrea să facă (sau este nevoit să facă) în funcție de zonele în care se află pionii (aceleași cazuri ca în diagramele anterioare).

Mesajul „Revenire începe tura” este unul care se întâmplă în momentul în care toți jucătorii au dat pe rând cu zarul.

**DIAGRAMA DE COLABORARE**

A close-up of a white background

Description automatically generated

Această diagramă este o convertire a diagramei de secvență, care evidenția mesajele dintre obiecte din punct de vedere al timpului, în a le evidenția din punct de vedere al spațiului. De exemplu, poate ilustra cum jucătorii și tabla de joc comunică între ele pentru a efectua mutările și pentru a actualiza starea jocului.