BlecGec

16.1.2025

Îndrumător:

dr. ing. Daniel Morariu

Student:

Moldovan Andrei

C_23/2

Istoric Versiuni

Data	Versiune	Descriere	Autor
19.11.2024	1.0	Am creat clasele: cards, deck, player, dealer și folder-ul cu cărțile ce vor fi afișate pe ecran.	Moldovan Andrei
22.11.2024	1.1	Am creat 4 Windows Forms: Game_Menu, How_to_play, Rules și Game.	Moldovan Andrei
27.11.2024	1.2	Am lucrat la interfața grafica pentru forms-ul pe care se va juca jocul.	Moldovan Andrei
7.12.2024	1.3	Jocul este funcțional dar poate fi jucat o singura data.	Moldovan Andrei
5.1.2025	1.5	Jocul este complet, am implementat un buton de „Play again” (retratează jocul) și cărțile se afișează pe ecran corespunzător valorii ei.	Moldovan Andrei

Cuprins

ISTORIC VERSIUNI **2**

CUPRINS **3**

1 SPECIFICAREA CERINȚELOR SOFTWARE **5**

1.1 Introducere **5**

1.1.1 Obiective 6

1.1.2 Definiții, Acronime și Abrevieri 7

1.1.3 Tehnologiile utilizate 7

1.2 Cerințe specifice **8**

2 GESTIONAREA PACHETULUI **10**

2.1 Descriere **10**

2.2 Fluxul de evenimente **10**

2.2.1 Amestecarea cărților in pachet 10

2.2.3 Pre-condiții 11

2.2.4 Post-condiții 12

3 **CALCULUL SCORURILOR** **13**

3.1 DESCRIERE **13**

3.2.1 Fluxul de evenintente 13

3.2.2 Pre-condiții 14

3.2.3 Post-condiții 14

4 IMPLEMENTARE **15**

4.1 DIAGRAMA DE CLASE **15**

4.2 DESCRIERE DETALIATĂ **16**

5 BIBLIOGRAFIE 17

1 Specificarea cerințelor software

1.1 Introducere

Jocurile de noroc au captivat imaginația oamenilor de-a lungul secolelor, iar Blackjack este unul dintre cele mai populare jocuri de cărți din lume, datorită combinației dintre strategie și șansă.

Proiectul **BlecGec** își propune să ofere o implementare interactivă a jocului clasic de Blackjack, utilizând limbajul de programare **C#** și tehnologia **Windows Forms** pentru a crea o interfață grafică intuitivă și atrăgătoare. Jocul este destinat pasionaților de jocuri de cărți care doresc să experimenteze o versiune digitală a acestui joc popular, respectând regulile de bază și oferind funcționalități esențiale precum distribuirea cărților, calculul punctajului, gestionarea turelor și determinarea câștigătorului.

Cerințele proiectului includ implementarea logicii fundamentale a jocului de Blackjack, precum și crearea unei interfețe grafice prietenoase. Elementele cheie ale aplicației includ:

- Afișarea pachetului de cărți și gestionarea lor grafică (tragerea și afișarea cărților).
- Calcularea valorii mâinii jucătorului și dealer-ului, respectând regulile Blackjack.
- Interacțiunea utilizatorului prin acțiuni specifice, cum ar fi „Hit”, „Stand” sau „Play Again”.

Proiectul este organizat într-o structură modulară, fiind compus din mai multe clase, fiecare având un rol bine definit:

1. **Card** – Modelarea unei cărți individuale, cu proprietăți precum culoarea (Suit), rangul (Rank) și imaginea asociată.
2. **Deck** – Gestionarea pachetului de cărți, inclusiv inițializarea și amestecarea acestuia.
3. **Player** – Reprezentarea unui jucător, incluzând mâna de cărți și calculul punctajului.
4. **Dealer** – O extensie a clasei Player, care adaugă logica specifică dealer-ului (tragerea cărților până la un punctaj minim).
5. **Game** – Logica principală a jocului, care asigură coordonarea între jucător, dealer și interfața grafică.
6. **Game** (Interfața principală) – Oferă utilizatorului o interfață vizuală pentru a interacționa cu jocul, conectând logica jocului cu elementele grafice.

Prin această arhitectură modulară, proiectul adoptă un model logic și clar, în care componentele sunt bine delimitate și ușor de extins. Totodată, proiectul facilitează înțelegerea principiilor fundamentale ale programării orientate pe obiecte (OOP).

Aplicația **BlecGec** este potrivită atât pentru utilizatorii care doresc să învețe regulile Blackjack-ului, cât și pentru cei care doresc să își perfecționeze strategiile de joc, oferind o experiență distractivă și educativă.

1.1.1 Obiective

Implementarea regulilor de bază ale jocului de Blackjack:

- Distribuirea cărților către jucător și dealer dintr-un pachet amestecat.
- Calcularea punctajului mâinilor conform regulilor Blackjack (figurile valorează 10, Asul poate valora 1 sau 11).
- Validarea condițiilor de joc, cum ar fi depășirea punctajului de 21 (bust).

Gestionarea fluxului de joc:

- Alternarea între tura jucătorului și tura dealer-ului.
- Implementarea mecanismului prin care dealer-ul trage cărți până la un punctaj minim de 18.
- Determinarea câștigătorului în funcție de regulile jocului (cel mai apropiat de 21 fără a-l depăși).

Realizarea unei interfețe grafice intuitive:

- Afișarea grafică a cărților distribuite, utilizând imagini pentru fiecare carte.
- Afișarea punctajelor curente pentru jucător și dealer.
- Butoane pentru acțiuni precum „Hit”, „Stand” și „Play Again”.

Afișarea și logarea informațiilor despre joc:

- Afișarea mesajelor cu rezultatul jocului (de exemplu, „Dealer busted!”, „Player wins!”).
- Logarea în consolă a valorilor cărților și a acțiunilor întreprinse pentru depanare.

Extensii și funcționalități suplimentare (neimplementate încă / planificate pentru viitor):

- Salvarea și încărcarea stării curente a jocului.
- Implementarea unui mod multiplayer local (doi jucători).
- Adăugarea de niveluri de dificultate pentru dealer (strategie avansată).

1.1.2 Definiții, Acronime și Abrevieri

- **Card**: O carte individuală din pachet, care are un rang (`Rank`), o culoare (`Suit`) și o imagine asociată.
- **Deck**: Pachetul standard de 52 de cărți utilizat în joc.
- **Hand**: Totalitatea cărților pe care le are un jucător în timpul unei runde.
- **Bust**: Situația în care punctajul unui jucător depășește 21, ceea ce duce automat la pierderea rundei.
- **Hit**: Acțiunea prin care jucătorul alege să primească o carte suplimentară.
- **Stand**: Acțiunea prin care jucătorul decide să nu mai primească alte cărți și să rămână cu punctajul actual.
- **Dealer**: Entitatea din joc care reprezintă adversarul controlat de calculator. Dealer-ul are reguli stricte de joc (de exemplu, trage cărți până la valoarea minimă de 18).
- **UI (User Interface)**: Interfața grafică a utilizatorului, utilizată pentru a interacționa cu jocul.
- **PictureBox**: Componentă din Windows Forms folosită pentru afișarea imaginilor, cum ar fi cărțile de joc.
- **Shuffle**: Procesul de amestecare aleatorie a pachetului de cărți înainte de distribuirea lor.

1.1.3 Tehnologiile utilizate

- **C# (C Sharp)** - Limbajul principal folosit pentru dezvoltarea aplicației.
- **Windows Forms** - Framework-ul pentru crearea aplicațiilor desktop GUI (Graphical User Interface) în C#. Acesta este utilizat pentru gestionarea interfeței vizuale, cum ar fi feroneriile și butoanele.
- **Object-Oriented Programming (OOP)** - Conceptul de programare orientată pe obiecte este utilizat pentru organizarea aplicației în clase (de exemplu, `Card`, `Deck`, `Player`, `Dealer`, și `Game`), fiecare având atribute și metode proprii.
- **Image Handling (GDI+)** - Manipularea și afișarea imaginilor folosind `Image` și `PictureBox` din GDI+ pentru a vizualiza cărțile de joc.
- **File I/O (Input/Output)** - Citirea și manipularea fișierelor de imagini cu ajutorul `File.Exists` și `Image.FromFile` pentru a încărca cărțile dintr-un folder specificat.

- **Randomization** - Utilizarea algoritmului de amestecare a cărților prin clasa `Random` pentru a amesteca pachetul de cărți.
- **Event-Driven Programming** - Aplicația utilizează evenimente pentru a interacționa cu utilizatorul, cum ar fi click pe butoane pentru a da "hit", "stand", sau a începe un nou joc.
- **Exception Handling** - Manevrarea excepțiilor pentru a preveni erori, cum ar fi `FileNotFoundException` atunci când fișierul imaginii nu este găsit.
- **Inheritance (Moștenire)** - Clasa `Dealer` moștenește clasa `Player`, având aceleași caracteristici, dar cu funcționalități suplimentare specifice dealerului, cum ar fi logica pentru jocul acestuia.
- **List Collection** - Utilizarea listei (`List<Card>`) pentru a stoca cărțile unui jucător, dealer și pachetul de cărți.
- **UI Updates** - Actualizarea interfeței grafice cu informații actualizate despre joc, cum ar fi suma cărților pentru jucător și dealer.
- **Control Management** - Crearea și gestionarea controalelor de tip `PictureBox` pentru a afișa cărțile pe ecran și a le organiza corespunzător.
- **Boolean Flags** - Folosirea unui flag (`PlayerTurn`) pentru a gestiona ordinea jocului între jucător și dealer.

1.2 Cerințe specifice

Implementarea regulilor de bază ale jocului de Blackjack:

- S-a realizat mecanismul de adăugare a cărților, în funcție de alegerea acestora (apăsarea butonului "Hit" pentru a adăuga o carte sau "Stand" pentru a încheia tura).
- S-a implementat logica privind imposibilitatea depășirii valorii de 21.
- S-a configurat finalizarea jocului atunci când un jucător sau dealerul ajunge la 21 sau depășește valoarea.

Gestionarea fluxului de joc și a turelor:

- Alternarea jucătorilor (Player și Dealer) după ce jucătorul face alegerea (Hit sau Stand), dealerul va juca conform regulilor.
- După încheierea unei runde, jocul se poate **reseta cu un nou pachet de cărți**, iar jocul poate continua pentru un alt jucător sau cu aceeași configurație.

Interfață grafică funcțională:

- Se creează o **zonă vizuală** pentru jucător și dealer, unde piesele (cărțile) sunt afișate dinamic în funcție de pozițiile lor.
- Se vor afișa **valorile cărților trasă** pentru jucător și dealer. Valorile cărților vor fi actualizate pe ecran după fiecare alegere a jucătorului (Hit sau Stand).
- Fiecare dată când un jucător adaugă o carte la mâna sa sau când dealerul ia o carte, interfața va trebui să se **actualizeze** și să reflecte noile valori ale cărților.

Afișarea și logarea informațiilor despre joc:

- Vor fi afișate valorile cărților din mâna jucătorului și dealerului. De asemenea, se vor înregistra evenimentele, cum ar fi depășirea valorii de 21 (bust) sau câștigarea rundei.
- După fiecare rundă, când rezultatul este determinat (jucătorul sau dealerul câștigă), se va afișa un **MessageBox** cu rezultatul și scorul actualizat.

2 Gestionarea pachetului de cărți

2.1 Descriere

Aceasta modalitate ne permite crearea pachetului de cărți de joc și amestecarea acestuia. Fiecare carte este reprezentata în clasa cards care ii atribuie cărții o valoare, un tip dar și imaginea corespunzătoare ei.

2.2 Fluxul de evenimente

- ➔ La inițializarea jocului este creat un obiect de tip Deck
- ➔ Pachetul de cărți este amestecat automat folosind metoda Shuffle
- ➔ Cărțile sunt trase și adăugate la player respectiv dealer prin metoda AddCard

2.2.1 Amestecarea cărţilor in pachet

La inițializarea jocului se creează o instanța a clasei deck. Metoda InitializeDeck este cea care creează toate cele 52 de cărți din pachet (13 valori de câte 4 tipuri diferite).

```
private void InitializeDeck()
        {
            string[] suits = { "hearts", "diamonds", "clubs", "spades" };
            string[] ranks = { "ace", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"jack", "queen", "king" };




            foreach (var suit in suits)
            {
                foreach (var rank in ranks)
                {
                    string imageLocation = Path.Combine($"{rank}_of_{suit}.png");
                    cards.Add(new Card(rank, suit, imageLocation));
                }
            }
        }
```

Metoda InitializeDeck se asigura ca pachetul a fost creat corect pentru a avea fix 52 de cărți diferite, fiind 13 cărți de 4 tipuri.

În cadrul metodei se inițializează o listă goala de cărți unde vor fi adăugate pe rând cărțile după ce sunt create. Se definesc doua tipuri de date Suits (culoarea) și Rank (valoarea).Pentru fiecare suit și rank se creează un obiect Card care este adăugat în lista de cărți.

După ce pachetul a fost creat acesta urmează să fie amestecat folosind metoda Shuffle.

```
public void Shuffle()
        {
            Random rng = new Random();
            int n = cards.Count;

            for (int i = n - 1; i > 0; i--)
            {
                int j = rng.Next(i + 1);
                (cards[i], cards[j]) = (cards[j], cards[i]);
            }
        }
```

Metoda `Shuffle` implementează un algoritm de amestecare bazat pe metoda **Fisher-Yates** (modernă), care amestecă în mod aleatoriu elementele unei liste. Obiectivul principal al acestei metode este de a rearanja cărțile din lista cards într-o ordine aleatorie. Procesul este unul simplu. Se creează un obiect de tip Random pentru a genera numere aleatorii, parcurgem lista de la ultimul element către primul pentru a garanta ca fiecare element din lista este amestecat, pe urma la fiecare pas alege un indice aleatoriu intre 0 si elementul curent (inclusiv) și la final permuta elementul curent cu cel de la indicele ales. Acest algoritm este standard pentru amestecarea datelor și garantează un rezultat nepredictibil la fiecare execuție.

2.2.2 Pre-condiții

După pornirea jocului și apăsare butonului de Start Game se va deschide fereastra jocului. Programul inițializează și amesteca pachetul de cărți, urmând sa împartă jucătorului și dealerului câte 2 cărți, a doua carte a dealerului fiind necunoscuta.

2.2.3 Post-condiții

Dacă jucătorul vede în masa sa 2 cărți reprezentata de un panel, cât și mana dealerului, înseamnă ca funcționalitatea s-a terminat corect și jocul poate continua.

3 Calculul scorurilor

3.1 Descriere

Calcularea scorurilor reprezintă o componentă esențială a jocului. Atât jucătorul, cât și dealerul, au un scor calculat în funcție de valorile cărților din mână. În mod special, Asul poate avea valoarea de 1 sau 11, în funcție de context, astfel încât să se evite depășirea limitei de 21. Această flexibilitate a valorii Asului permite menținerea unui echilibru optim în joc și asigură strategii variate pentru jucători.

3.2 Fluxul de evenimente

→ La începerea jocului sunt definite doua variabile playerSum si dealerSum care reține valoarea mâinii fiecăruia

→ La fiecare carte primita suma din label se actualizează automat

3.2.1 Calcularea scorului

După fiecare extragere de carte ,metoda AddCard din clasa Player adaugă cartea in mana jucatorului.Metoda GetHandValue este apelata pentru a evalua noul scor al jucătorului. După care este afișat pe interfața grafica.

```
public int GetHandValue()
        {
            int value = 0;
            int aceCount = 0;

            foreach (var card in Hand)
            {
                if (int.TryParse(card.Rank, out int numericValue))
                {
                    value += numericValue;
                }
                else if (card.Rank == "ace")
                {
                    value += 11;
                    aceCount++;
                }
                else
```

```
            {
                value += 10;
            }
        }



        while (value > 21 && aceCount > 0)
        {
            value -= 10;
            aceCount--;
        }

        return value;
    }
```

Daca jucătorul are mai mulți ași in mana ,aceștia sunt calculați dinamic, alternând valoare lor intre 1 si 11, pentru a menține scorul cat mai aproape de 21.

3.2.2 Pre-condiții

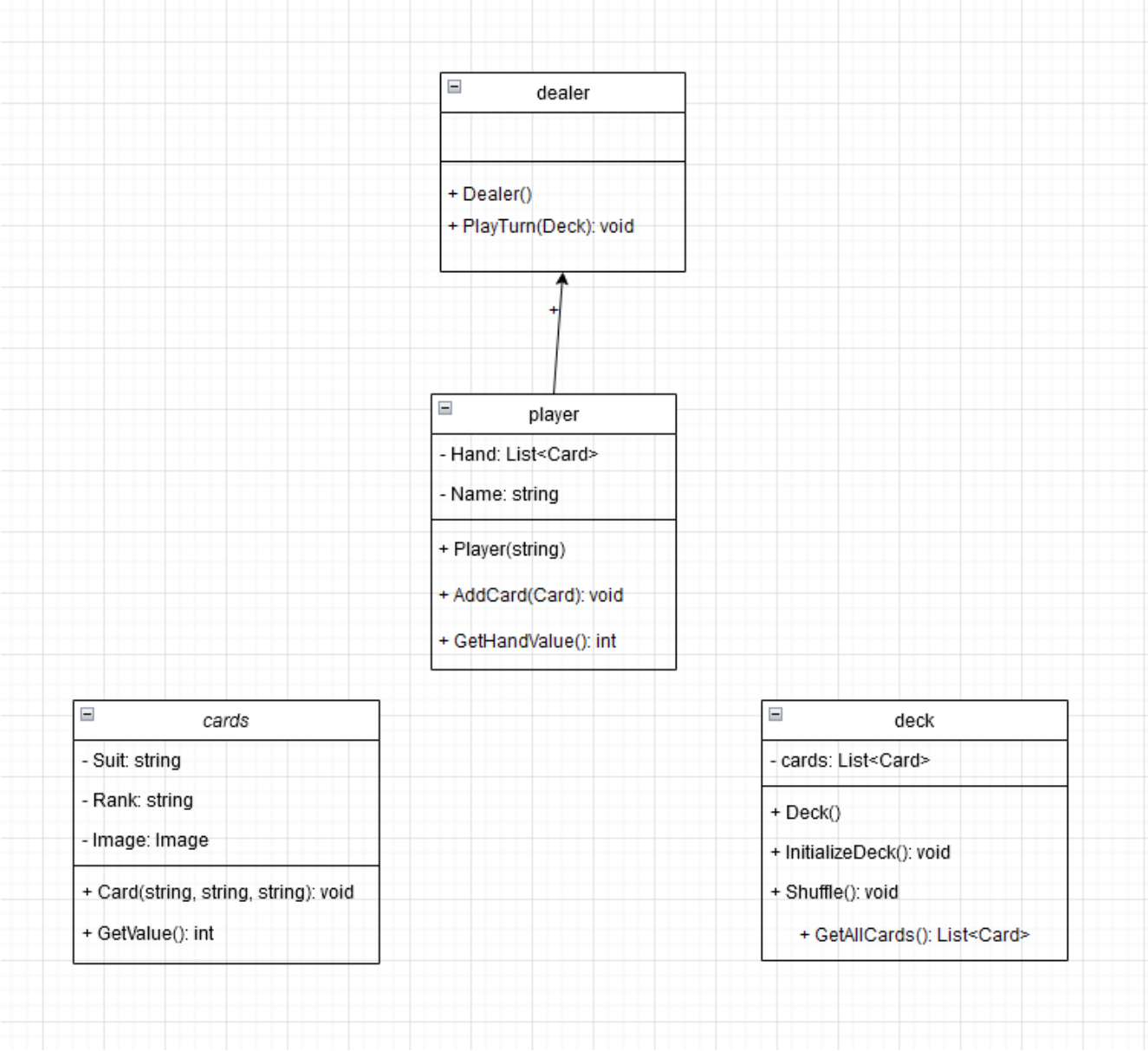
Jucătorul sau dealerul trebuie să aibă cel puțin doua cărți în mână pentru a calcula un scor valid. Cărțile trebuie adăugate corect mâinii. Metoda GetHandValue trebuie sa tina cont de valoare dinamica a asului.

3.2.3 Post-condiții

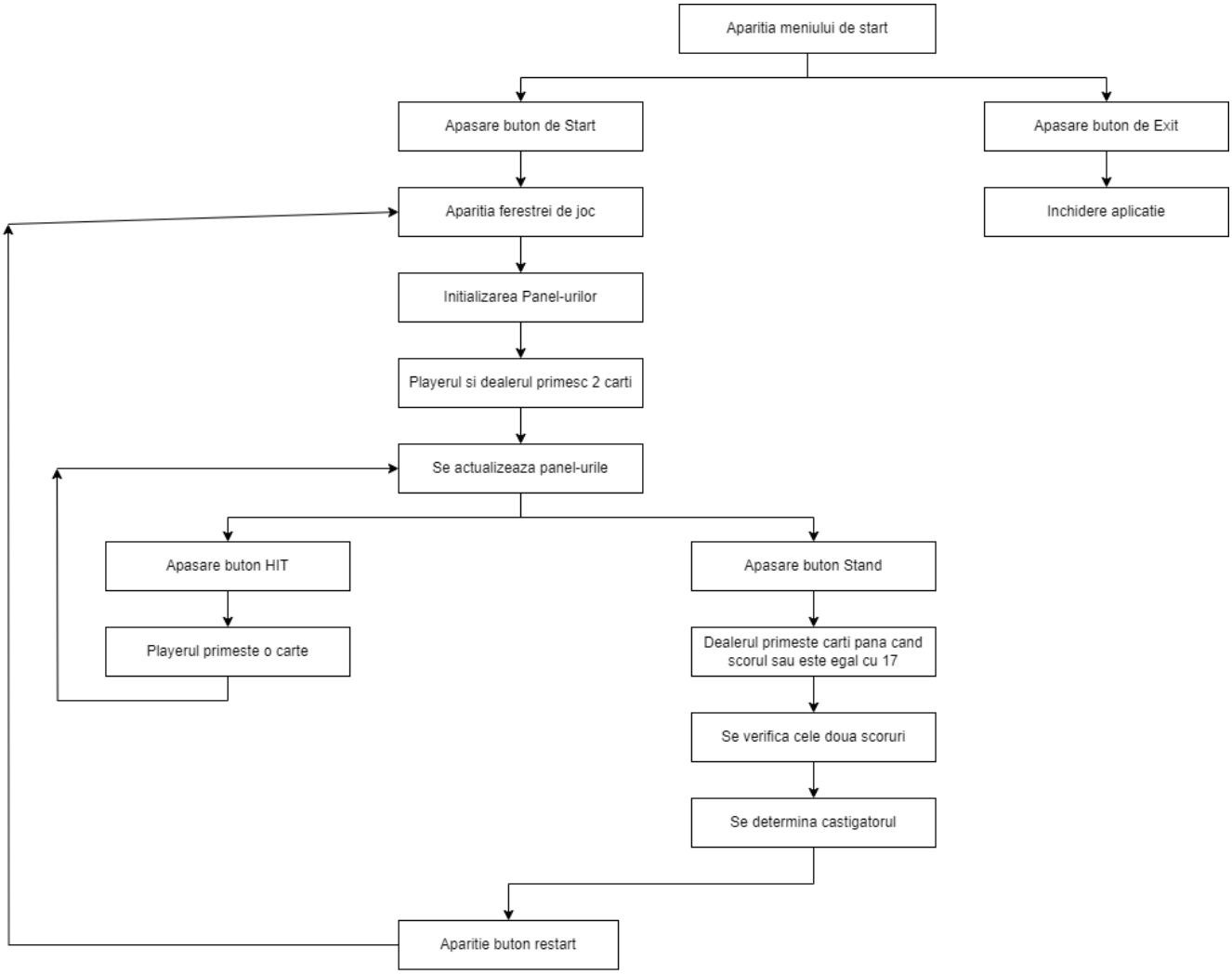
Scorul calculat este afișat pe interfața grafica cu ajutorul unui label pentru fiecare jucator.Dupa ce jucătorul apasă butonul Stand, se încheie tura sa ,iar apoi urmează sa fie calculat automat scorul dealerului. Jucătorul primește feedback vizual imediat asupra stării jocului (ex. „Ai pierdut” dacă scorul depășește 21).

4 Implementare

4.1 Diagrama de clase



4.2 Descriere detaliată



5 Bibliografie

https://github.com/Ionshard/BlackJack

https://github.com/pda87/Blackjack

https://chatgpt.com/

https://www.youtube.com/watch?v=I-QZ4I_RibQ&ab_channel=IvanTop

https://stackoverflow.com/questions/16463599/popup-window-in-winform-c-sharp

https://www.c-sharpcorner.com/article/working-with-popup-notification-in-windows-forms/

https://stackoverflow.com/questions/34316096/i-have-created-a-black-jack-game-in-c-for-one-player-how-can-i-create-it-for

https://www.csharp-tutorial.hu/csharp/blackjack-game-part-1/

https://www.csharp-tutorial.hu/csharp/blackjack-game-part-2/

https://www.csharp-tutorial.hu/csharp/blackjack-game-part-3/\

https://www.youtube.com/watch?v=r0KTuGn-nhw&list=PLBT4d1188Q3sEyYZxTfunthVr1GvLpBeK

https://www.youtube.com/watch?v=78G04i_v-Rc&ab_channel=Micky