**Poor Man’s Go**

**17/01/2025**

|  |  |  |
| --- | --- | --- |
| **Îndrumător:** |  | **Student:** |
| **dr. ing. Daniel Morariu** |  | **Badea, Octavian-Andreas (C\_21/1)** |

Istoric Versiuni

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versiune** | **Descriere** | **Autor** |
| 07/11/2024 | 0.1 | Ilustrarea inițială a tablei de joc pentru diverse dimensiuni | Badea Andreas |
| 10/11/2024 | 0.2 | Crearea claselor CellPos și CellPos\_ExtraPrecision;  Calcularea poziției mouse-ului pe tablă (coordonatele în pixeli și pe grila de joc);  Afișarea unei pietre la poziția respectivă pe interfața grafică | Badea Andreas |
| 12/11/2024 | 0.3 | Rezolvarea unui bug cauzat de evenimentul MouseMove când se ține apăsat butonul de Left-Click și se mută cursorul în afara tablei | Badea Andreas |
| 16/11/2024 | 0.4 | Implementarea unui comparator de egalitate pentru tipul Point;  Crearea clasei Go\_Board | Badea Andreas |
| 20/11/2024 | 0.5 | Crearea clasei Go\_String;  Punerea unei pietre pe tablă | Badea Andreas |
| 22/11/2024 | 0.6 | Crearea unei ferestre cu meniu de start | Badea Andreas |
| 04/12/2024 | 0.7 | Eliminarea unei instanțe Go\_String când rămâne fără libertăți;  Actualizarea tablei în cazul descris | Badea Andreas |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versiune** | **Descriere** | **Autor** |
| 09/12/2024 | 0.8 | Redimensionarea elementelor de pe interfața grafică după dimensiunea ferestrei de joc (exceptând textul) | Badea Andreas |
| 13/12/2024 | 0.9 | Mică refactorizare pentru a elimina dependențe statice | Badea Andreas (realizat după sfatul profesorului îndrumător) |
| 21/12/2024 | 0.9 | Calcularea scorului de la tură la tură | Badea Andreas |
| 05/01/2025 | 0.10 | Conectarea în rețea a 2 (fără vreun transfer de date între ture) | Badea Andreas |
| 11/01/2025 | 0.11 | Regres la versiunea 0.9 deoarece am făcut conexiunea sincronă... | Badea Andreas |
| 14/01/2025 | 0.12 | Punerea unui buton care cedează tura inamicului;  Afișarea unui mesaj la sfârșitul jocului în funcție de diferența dintre scoruri | Badea Andreas |

Cuprins

Istoric Versiuni 2

Cuprins 4

1 Specificarea cerințelor software 5

1.1 Introducere 5

1.1.1 Obiective 5

1.1.2 Definiții, Acronime și Abrevieri 6

1.1.3 Tehnologiile utilizate 7

1.2 Cerințe specifice 8

2 Previzualizarea mișcărilor legale & Calcularea scorului tură-de-tură 9

2.1 Descriere 9

2.2 Fluxul de evenimente 9

2.2.1 Fluxul de bază (Previzualizare & Scor) 9

2.2.2 Pre-condiții 11

2.2.3 Post-condiții 12

3 Implementare 13

3.1 Diagrama de clase 13

4 Bibliografie 14

# Specificarea cerințelor software

## Introducere

Intitulat “Poor Man’s Go”, proiectul este încercarea autorului de a translata jocul de “Go” într-un format software, aplicând concepte de Programare Orientată pe Obiecte și încercând să se abordeze dificultățile ce vor apărea într-o astfel de sarcină folosind cunoștințele dobândite (precum detalii legate de tehnologiile utilizate și șabloanele de proiectare unde este cazul).

Totuși, scopul aplicației nu va fi de a reproduce exhaustiv jocul de “Go”, urmărind literă de literă regulamente recunoscute, precum cel chinezesc, respectiv japonez, ci de a îl folosi drept o oportunitate de a dezvolta propriile competențe.

### Obiective

Obiectivul principal este de a aduce proiectul într-o stare funcțională, definită de posibilitatea ca 2 jucători să interacționeze cu aplicația prin interfața grafică astfel încât să simuleze interpretarea autorului a acestui joc, ceea ce va presupune realizarea unei subcategorii de obiective:

* Crearea unei ferestre cu meniul principal
* Permiterea jucătorilor să aleagă dimensiunea tablei
* Conectarea prin rețea a 2 jucători, având o relație server-client drept analog pentru o relație provocator-contestant (sau gazdă-invitat)
* Generarea unei interfațe grafice interactive pentru tabla de joc
* Previzualizarea mișcărilor legale
* Crearea unei clase care să gestioneze pietre adiacente la momentul punerii piesei
* Calcularea scorului tură-de-tură
* Crearea unui buton care permite unui jucător să își cedeze tura
* Impunerea condiției de sfârșire a partidei folosind respectivul buton
* Adăugarea unui cronometru care indică timpul rămas jucătorului să realizeze o mișcare
* Înregistrarea unui joc prin codificarea mutărilor folosind notații specifice, asemănător cu șahul

### Definiții, Acronime și Abrevieri

Termeni ai jocului:

* Piatră/Stone – piesă de culoare neagră sau albă, de formă biconvexă
* Goban – tabla de joc, a cărui celule pe care vor fi plasate piesele sunt colțurile pătratelor ilustrate pe ea
* Libertate/Liberty – celulă goală, adiacentă unei pietre într-una din cele 4 direcții cardinale de bază
* String (Go) – totalitatea pietrelor de aceeași culoare, vecine între ele tot în cele 4 direcții cardinale
* Captură/Capture – înconjurarea unui String de către alte String-uri adversare și/sau marginea tablei astfel încât nu mai are libertăți și este considerată a fi “prinsă” și scoasă de pe tablă
* Teritoriu – celulă goală, înconjurată doar de pietre de aceeași culoare și marginea Goban-ului
* Ochi – libertatea unui String care nu poate fi ocupată de inamic fără a executa o mișcare sinucigașă de autocapturare (cum ar indica și numele, pot fi create pentru un String în număr de 2 cel puțin)

Termeni folosiți în scrierea documentației:

* Tablă de joc –face referire la interfața grafică (GUI) cu care jucătorii interacționează în mod direct
* Grilă de joc – folosit când este vorba despre mecanismele propriu-zise din spatele interfeței grafice (deci a “tablei de joc”), cărora li se aplică operațiile necesare pentru a stimula interacțiunea om-mașină, intermediară a ulterioarei interacțiuni jucător-jucător
* Coordonate matriciale/pe tabla de joc – perechile de indecși la care vor fi marcate marginile tablei, spațiile goale și pietrele negre/albe într-un tablou bidimensional
* Coordonate în pixeli/pe grila de joc – perechile de indecși care denotă poziția cursorului pe tabla de joc

Clase:

* Go\_Board – pentru gestionarea datelor referitoare la grila de joc
* Go\_String – pentru gestionarea pietrelor care ar forma un String (Go)
* PlayerWindow\_Safe – clasă modelată pe baza șablonului de proiectare “Proxy”
* Point\_IEqualityComparer – implementează interfața IEqualityComparer<T> pentru tipul-structură “Point”
* CellPos – menține coordonate matriciale și dimensiunea unei celule
* CellPos\_ExtraPrecision – clasă moștenitoare lui “CellPos”, menține coordonate pe ecran, coordonate matriciale derivate calculate cu acestea; redefinește anumite metode din clasa de bază

Metode:

* GetBlack\_WhiteScoreDifference() – calculează și returnează diferența dintre scorul jucătorului cu pietre negre și jucătorului cu pietre albe
* ReadjustFields() – recalculează parametrii folosiți pentru ilustrarea interfeței grafice pe baza dimensiunii unei celule și lungimea tablei în coordonate matriciale
* DrawEmptyCell() – desenează la coordonatele matriciale o celulă goală în funcție de poziția ei pe tablă

### Tehnologiile utilizate

#### Proiect:

**1. Mediul de dezvoltare:**

* **Visual Studio** Community 2019

**2. Limbaj de programare:**

* **C#** -limbajul principal utilizat pentru a crea proiectul
* **XML** - folosit pentru a scrie documentație în cod unde s-a considerat că este cazul

**3.Framework-uri și biblioteci:**

* **.NET Framework** (versiunea .NET Framework 4.7.2) - platformă pentru rularea aplicațiilor Windows Forms.
* **Windows Forms** - utilizată pentru crearea interfeței grafice.

#### Documentație:

**1. Editor de text:**

* **Word 2016**

**2. Diagramele:**

* **Visual Studio Class Designer**

## Cerințe specifice

Funcționalități implementate până la momentul predării:

* Crearea unei ferestre cu meniul principal
* Permiterea jucătorilor să aleagă dimensiunea tablei
* Generarea unei interfațe grafice interactive pentru tabla de joc
* Previzualizarea mișcărilor legale (crearea unui obiect Go\_String fără salvarea pe tabla și grila de joc a modificărilor aduse de acesta)
* Crearea unei clase care să gestioneze pietre adiacente la momentul punerii piesei (este vorba de clasa Go\_String și modul în care ea modifică conținulul grilei de joc pentru a putea extrage informații relevante)
* Calcularea scorului tură-de-tură (egal cu nu
* Crearea unui buton care permite unui jucător să își cedeze tura
* Impunerea condiției de sfârșire a partidei folosind respectivul buton

# Previzualizarea mișcărilor legale & Calcularea scorului tură-de-tură

## Descriere

Prima funcționalitate aleasă va afișa la poziția cursorului pe tabla de joc o piatră de culoarea jucătorului dacă la poziția respectivă poate fi realizată mișcarea.

Motivul alegerii funcționalității de previzualizare este simplu: constituie singurul mod prin care jucătorul poate să interacționeze cu tabla de joc în ceea ce îl privește informația grafică valabilă într-o tură, exceptând punerea unei pietre. În cazul în care lipsea, jucătorul ar trebui să descopere singur dacă o mutare dată este legală, iar în caz contrar ar trebui să țină minte pozițiile la care nu poate pune o piatră și să reia verificarea cu o altă celulă, ceea ce ar fi evident agasant și ar putea duce la mutări accidentale.

După punerea unei pietre și gestionarea ei (adică la sfârșitul turei), va urma procesul de determinare a pietrelor care au fost capturate și a teritoriului fiecărui jucător. Formula generalizată este:

scor[jucător] = teren\_ocupat[jucător] + număr\_pietre[jucător]

De asemenea, calcularea scorului este evident necesară pentru desfășurarea unui joc; fără ea, nu am avea o stare de câștig. Se remarcă faptul că formula folosită în calcul scorului este:

## Fluxul de evenimente

### Fluxul de bază (Previzualizare & Scor)

**Previzualizare**

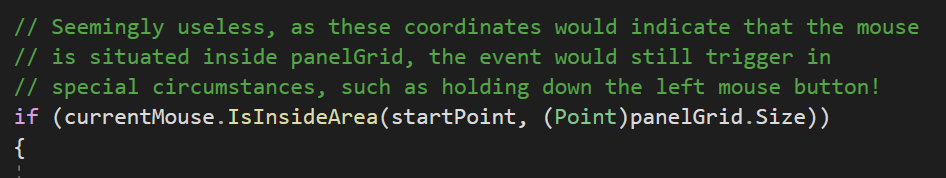
Când este tura unui jucător anume, evenimentul MouseMove, delegat obiectului panelGrid va fi, evident, impus când utilizatorul va muta cursorul pe obiect, dar și în cazul în care el va ține butonul de dreapta al mouse-ului apăsat și va plimba cursorul în afara obiectului, ceea ce este mai puțin clar și în niciun caz deductibil de jucător la prima vedere. Vom folosi 2 obiecte CellPos pentru a reține coordonatele la care se află cursorul acum (currentMousePose) și înainte de a îl muta (previousMousePos, declarat cu coordonate din interiorul tablei pentru a evita desenarea unei pietre în poziții eronate). Primul precizat va fi inițializat de fapt drept un CellPos\_ExtraPrecision, moștenitoarea lui CellPos descrisă pe scurt în secțiunea 1.1.2, aplicând polimorfismul deoarece:

1. Coordonatele matriciale ale lui previousMousePos sunt dependente de coordonatele matriceale ale lui currentMousePos, deci am dori ideal să le transferăm printr-o singură copiere; dacă am avea 2 obiecte CellPos\_ExtraPrecision, am reține o coordonată inutilă (având nevoie de coordonata în pixeli doar pentru poziția curentă în cazul descris în paragraful anterior, putem

evita modificarea poziției anterioare dacă nu ne aflăm pe tablă); dacă ambele obiecte ar fi CellPos pure, nu avem coordonatele în pixeli, ar fi nevoie să tratăm membrii clasei CellPos într-un scop în care nu au fost definite și ar trebui să executăm niște calcule pentru coordonatele matriciale de metoda apelantă, ceea ce nu e necesar dacă ascundem operațiile în spatele lui CellPos\_ExtraPrecision

1. Din moment ce coordonatele în pixeli sunt necesare doar pentru cazul particular descris, rezultă că, odată ce au fost pasate, nu este nevoie să mai avem acces la ele, deci o putem ascunde prin polimorfism

Pentru a evita erori cauzate de această interacțiune, se va verifica dacă cursorul se află la o poziție la care, dacă ar fi desenată piatra, ar ieși din spațiul în care pot fi puse pietrele (un dreptunghi).



^^^Parametrii: colțurile din stânga-sus, respectiv dreapta-jos a dreptunghiului^^^

În caz contrar, jucătorul are cursorul în afara tablei, deci se va șterge previzualizarea de pe margine în cazul în care a fost executată (adică dacă celula anterioară a fost goală) și coordonatele matriciale va prelua o valoare din interiorul tablei de joc.

În cazul în care este cursorul în aria precizată, se va verifica dacă coordonatele matriciale anterioare și curente sunt distincte. Dacă sunt, se vor pune se va șterge previzualizarea poziției anterioare dacă a fost executată (exact ca în cazul din paragraful anterior se face verificarea), se copiază coordonatele matriciale de la poziția actuală la cea anterioară, apoi se previzualizează punerea piesei dacă la poziția actuală se poate pune piesa.

Pentru a realiza previzualizarea propriu-zisă, dacă poziția matricială e goală, vom crea un obiect Go\_String care se va constitui din configurația tablei în acea tură, care va număra câte pietre ar captura cu mutarea dată, pietrele care fac parte din String (Go) și libertățile acestuia (operații realizate pe o copie a listei de obiecte Go\_String care menține de fapt configurația din tura anterioară după mutare, anume Go\_Strings). Dacă are mai mult de zero libertăți sau ar putea captura măcar o instanță Go\_String ca urmare a mutării, previzualizarea este validă, iar fereastra meciului va apărea desenată piatra.

**Scor**

Odată ce obiectul Go\_String a fost gestionat, vom salva modificările aduse în această tură de la CopyOfGo\_Strings în Go\_Strings, după care vom extrage libertățile pietrei puse în această tură. Vom aplica pe matricea grilei de joc un Floodfill pentru fiecare libertate existentă și nemarcată, marcând fiecare celulă nevizitată (cellsJustExplored).

Dacă celula este goală, o vom pune într-o coadă pentru a o vizita ulterior (toBeExplored) și într-o altă coadă formată doar din pozițiile goale (exploredEmptySpaces). În schimb, dacă celula conține o piatră inamică, vom ține cont c-am văzut măcar o asemenea piatră.

După ce au fost vizitate toate celulele goale dintr-o arie (remarcăm faptul că vom parcurge doar celule goale), dacă am dat de măcar o piatră inamică, teritoriul din exploredEmptySpaces e neutru, deci îl vom șterge din HashSet-ul cu teritorii inamice (doar inamicul putând pierde teritorii pentru că au devenit neutre într-o tură anume, jucătorul actual nu își poate face propriile teritorii neutre). Altfel, teritoriului este al jucătorului curent, deci vom adăuga în HashSet-ul cu teritoriile jucătorului curent.

La final, incrementăm numărul de pietre al jucătorului curent, scădem din numărul pietrelor inamice numărul de pietre capturate de obiectul Go\_String recent creat, ștergem din teritoriile ambiilor jucători piatra pusă în această tură apoi folosim formulele scrise în secțiunea 2.1

Într-un final, afișăm scorurile obținute pe interfața grafică pentru ca jucătorii să le vadă.

### Pre-condiții

**Previzualizare**

Trebuie pus cursorul pe tabla de joc pe o celulă dată (sau scos cursorul de pe tablă sau chiar pus pe o piatră aflată pe tablă în cazul operației inverse, prevăzută tot de respectiva funcționalitate).

**Scor**

Trebuie pusă o piatră pe o poziție legală prin click-stânga.

### Post-condiții

**Previzualizare**

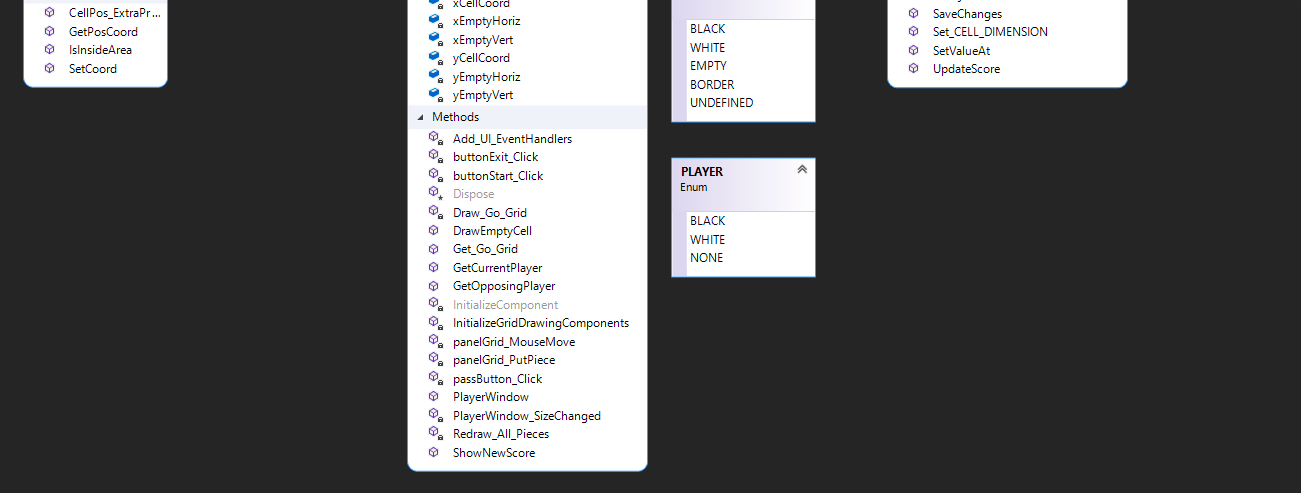
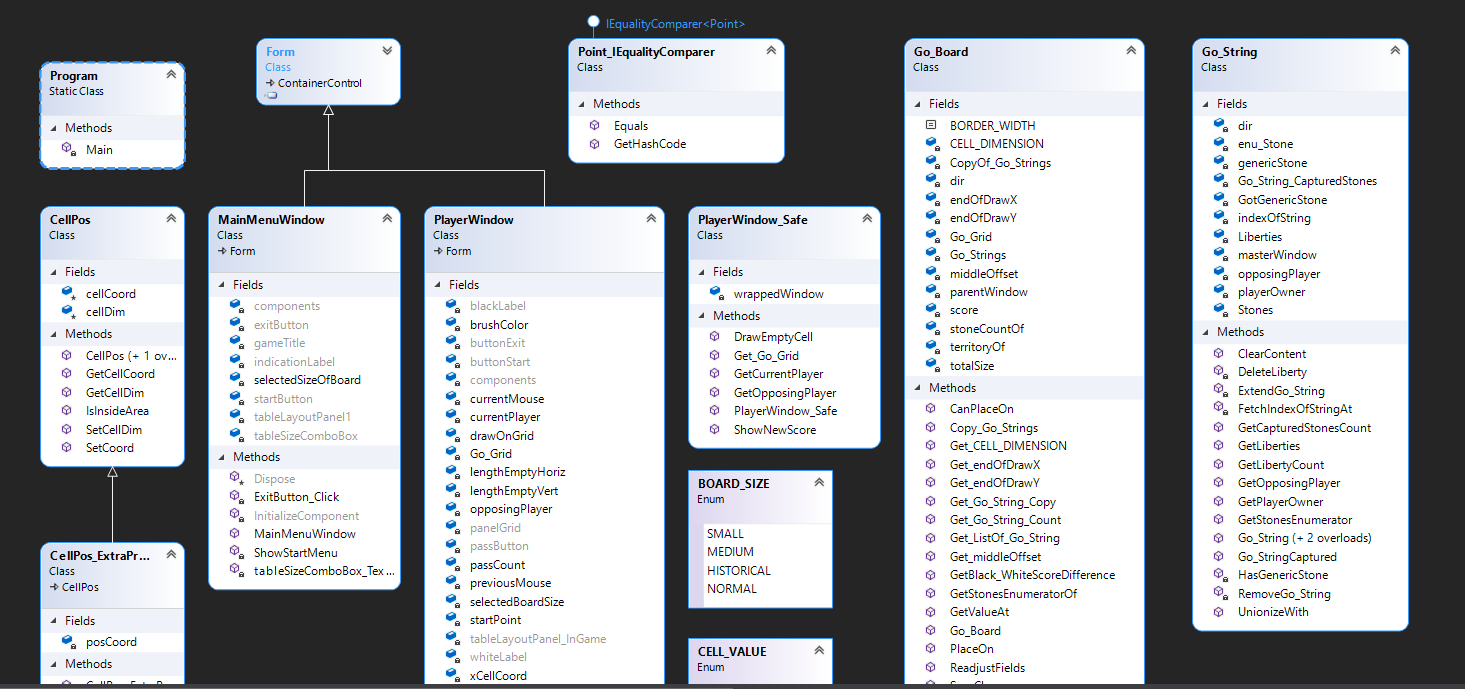
Se va desena piesa dacă mișcarea de pe celula dată este legală (și se va șterge dacă am scos cursorul de pe tablă de pe o celulă goală sau pur și simplu de la o celulă goală la una a cărui conținut nu e relevant pentru ștergere).

**Scor**

Ar trebui să apară pe interfața grafică noul scor.

# Implementare

## Diagrama de clase



# Bibliografie

* Programare orientata pe obiecte II 2024-2025 sem I – <https://classroom.google.com/u/0/c/NzE5Njc3MjU4NDM5>
* POO an 1 - 2023-2024 An 1 C+TI+ISM – <https://classroom.google.com/u/0/c/NjY1MzE3MDIwMTMw> (mai precis, arhiva cu materialele compilate de către domnul profesor Macarie Breazu)
* Stack Overflow - Where Developers Learn, Share, & Build Careers – <https://stackoverflow.com/>
* Use Sockets to send and receive data over TCP - .NET | Microsoft Learn – <https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/sockets/socket-services>
* Go Game – <https://www.mathsisfun.com/games/go.html> (inspirație pentru design în general)
* Kifu Snap – <https://www.crazy-sensei.com/?lang=en&location=kifu_snap> (inspirație pentru calculul scorului în cazuri particulare – precum când tabla are doar pietre de aceeași culoare)