A black background with red lines

Description automatically generated

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE**

**TEMA 1**

**JOC LABIRINT IEPURAȘ**

Disciplina : Proiectare software

Student : Pojar Andrei-Gabriel

An : III

Grupa : 30233

Profesor coordonator : Anca Iordan

**CUPRINS**

1. Introducere

* Enunțul problemei
* Motivație

1. Instrumente utilizate
2. Justificarea limbajului de programare ales
3. Descrierea diagramelor UML
4. Descrierea aplicației
5. Testare
6. Concluzii
7. **Introducere**

* **Enunțul problemei**

Obiectiv: Obiectivul acestei teme este familiarizarea cu șablonul architectural Model-View-Presenter. Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

Cerințe:

În faza de analiză se va realiza diagrama cazurilor de utilizare. În faza de proiectare se va realiza diagrama de clase respectând arhitectura MVP și principiile SOLID, dar și diagrama entitate-relație corespunzătoare bazei de date.

În faza de implementare se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:

➢ proiectarea dată de diagrama de clase;

➢ unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

În faza de testare se vor implementa teste unitate (proiect de testare) corespunzătoare operațiilor de creare a bazei de date, de realizare a conexiunii la baza de date, de creare a tabelelor și de interogare a tabelelor bazei de date.

Finalizarea temei va consta în predarea unui director ce va cuprinde:

➢ Un fișier cu diagramele UML realizate;

➢ Baza de date;

➢ Aplicația soft;

➢ Documentația (minim 10 pagini) - un fișier care cuprinde: ▪ numele studentului, grupa; ▪ enunțul problemei; ▪ instrumente utilizate; ▪ justificarea limbajului de programare ales; ▪ descrierea diagramelor UML; ▪ descrierea aplicației.

Dezvoltați o aplicație pentru următorul joc: se un labirint de dimensiune nxn căsuţe în care pot să existe capcane pentru animale sălbatice şi o “comoară” de morcovi. Într-un colţ al labirintului se află un iepuraș flămând. Ajutaţi-l pe iepurașsă ajungă la “comoara” de morcovi astfel încât să nu fie prins într-una din capcane, ştiind că el se poate deplasa prin sărituri pe diagonală în căsuțele care au un vârf comun cu căsuța curentă. Jocul va fi dezvoltat pe 3 niveluri (n=4,5,6). După ce un jucător a obținut soluția, se va afișa numărul de deplasări utilizat de către acesta în soluția furnizată. Jucătorii vor putea să se autentifice și să-și vizualizeze punctajul. Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖ Operații CRUD pentru informațiile legate de jucătorii care s-au autentificat;

❖ Vizualizarea listei tuturor jucătorilor autentificați.

* **Motivație**

Aplicația dezvoltată aduce beneficii multiple prin furnizarea unei experiențe distractive și educative, oferind utilizatorilor un joc de labirint cu trei nivele de dificultate, îmbunătățind abilitățile cognitive și promovând gândirea strategică și rezolvarea de probleme. Integrarea unui sistem de autentificare și punctaj stimulează competiția și motivarea utilizatorilor, în timp ce funcționalitățile pentru administratori facilitează gestionarea eficientă a utilizatorilor și a datelor asociate.

1. **Instrumente utilizate**

Instrumentele utilizate pentru proiectarea, implementarea si testarea acestei aplicatii software, sunt:

* IntelliJ IDEA este utilizat ca mediu principal de dezvoltare pentru codul sursă al aplicației. Dezvoltat de compania JetBrains, care oferă suport pentru dezvoltarea de aplicații software în mai multe limbaje de programare, inclusiv Java, Kotlin, Scala, Groovy, JavaScript, TypeScript, HTML, CSS și altele. IntelliJ IDEA vine cu o serie de funcționalități avansate, cum ar fi completarea automată a codului, refactorizarea inteligentă, depanarea integrată, controlul versiunilor și multe altele, care facilitează procesul de dezvoltare și îmbunătățesc productivitatea dezvoltatorilor.
* MySQL utilizat ca sistem de management al bazelor de date (SGBD). MySQL oferă suport pentru Structured Query Language (SQL), facilitând interacțiunea cu baza de date prin intermediul unui set standardizat de comenzi și operații. Am mai folosit MySQL pentru crearea tabelelor, operatii CRUD pe datele tabelelor, tabele relationale.
* StarUML care permite dezvoltatorilor să creeze diagrame UML într-un mod vizual și intuitiv. Am folosit StarUML pentru modelarea si reprezentarea diagramei cazurilor de utilizare, diagramei de clase respectiv a diagramei entity-relationship corespunzatoare bazei de date.

1. **Justificarea limbajului de programare ales**

Am ales ca limbaj de programare Java pentru implementarea acestui joc din urmatoarele motive:

* Fiabilitate și performanță: Java este un limbaj robust și fiabil, ceea ce îl face potrivit pentru dezvoltarea de aplicații critice, cum ar fi jocuri și aplicații interactive. De asemenea, Java beneficiază de un sistem de gestionare a memoriei eficient și de tehnici de optimizare care îmbunătățesc performanța aplicațiilor.
* Portabilitate: Java este cunoscut pentru portabilitatea sa, ceea ce înseamnă că aplicațiile dezvoltate în Java pot fi rulate pe diverse platforme și sisteme de operare fără modificări semnificative.
* Versatilitate: Java este un limbaj versatil, potrivit pentru o varietate largă de aplicații, inclusiv dezvoltarea de jocuri, aplicații desktop, aplicații web și multe altele.
* Siguranța: Java este un limbaj securizat și robust, cu un sistem de securitate puternic integrat. Acest lucru este crucial pentru aplicațiile care gestionează date sensibile sau care rulează în medii critice unde fiabilitatea este esențială.

Pentru partea de interfață grafică a acestui joc am ales Java Swing din următoarele motive:

* Flexibilitate și personalizare: Java Swing oferă o gamă largă de componente grafice și un set flexibil de funcționalități de personalizare. Aceasta vă permite să creați interfețe utilizator complexe și atractive, adaptate nevoilor și cerințelor specifice ale aplicației dvs.
* Compatibilitate cu Java: Fiind parte a platformei Java standard, Java Swing se integrează perfect cu alte tehnologii Java, cum ar fi JDBC pentru accesul la baza de date MySQL și altele. Acest lucru facilitează dezvoltarea și integrarea întregii aplicații.
* Ecosistem larg de biblioteci și framework-uri: Datorită popularității și istoricului său lung, Java Swing are un ecosistem bogat de biblioteci și framework-uri care îmbunătățesc funcționalitățile sale și permit dezvoltatorilor să extindă și să îmbunătățească aplicațiile lor cu ușurință.

1. **Descrierea diagramelor UML**

* **Diagrama de use case**

O diagramă de cazuri de utilizare (Use Case Diagram) este o diagramă în UML (Unified Modeling Language) care descrie interacțiunile între actori (utilizatori externi sau sisteme) și un sistem software în procesul de modelare a cerințelor.

**A diagram of a person's relationship

Description automatically generated**

* **Diagrama de clase**

Diagrama de clase este un tip de diagramă în cadrul limbajului de modelare UML (Unified Modeling Language) care prezintă structura statică a unui sistem software. Această diagramă descrie clasele din sistem, atributele și metodele acestora, precum și relațiile între clase.

A close-up of a computer diagram

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

* **Diagrama entitate-relație corespunzătoare bazei de date**

Diagrama de entitate-relație (ER) este un instrument de modelare a datelor care este utilizat pentru a reprezenta structura datelor într-o bază de date. Această diagramă se concentrează pe entități (obiecte sau lucruri distincte despre care se dorește să se păstreze informații) și relațiile dintre aceste entități.

**A screenshot of a computer

Description automatically generated**

1. **Descrierea aplicației**

Această aplicație reprezintă un joc de labirint în care jucătorii trebuie să ajute un iepuraș flămând să găsească "comoara" de morcovi fără a fi prins în capcanele pentru animale sălbatice. Jocul este disponibil în trei niveluri de dificultate, iar după ce un jucător completează un nivel, se va afișa numărul de deplasări effectuate.

**Funcționalități principale:**

* Înregistrare: Utilizatorii vor putea să se autentifice pentru a accesa jocul și funcționalitățile adiționale.

Există două tipuri de utilizatori: jucători obișnuiți și administratori.

* Joc Labirint: Se va afișa labirintul cu dimensiunile respective pentru fiecare nivel.

Iepurașul va fi plasat într-un colț al labirintului, iar comoara de morcovi va fi amplasată în alt loc.

Capcanele pentru animale sălbatice vor fi plasate aleatoriu în labirint. Jucătorul va putea să-și deplaseze iepurașul folosind sărituri pe diagonale în căsuțele adiacente. Scopul este de a ajunge cu iepurașul la comoara de morcovi evitând capcanele. Nivele: Există trei nivele de dificultate diferite, fiecare cu dimensiuni proprii de labirint. Nivelurile sunt de dimensiunile n=4, n=5 și n=6.

* Calculul numărului de deplasări:

După ce jucătorul ajunge la comoara de morcovi, numărul total de deplasări efectuate este afișat.

* Gestionarea Utilizatorilor:

Utilizatorii autentificați pot vizualiza propriul punctaj. Administratorii au acces suplimentar pentru a gestiona informațiile despre jucători, inclusiv operații CRUD și vizualizarea listei de jucători autentificați.

* Interfață Grafică:

Interfața grafică va fi intuitivă și atrăgătoare, oferind o experiență plăcută utilizatorilor.

Se va afișa grafic labirintul și mișcările iepurașului în timp real pe ecran.

Baze de Date:

* Informațiile despre utilizatori, inclusiv numele, parolele și punctajele, vor fi stocate într-o bază de date.

**Arhitectura MVP (Model-View-Presenter)**

* Model: Reține datele și logica de afaceri. Include clasele pentru definirea structurii de date și operațiile pe acestea, precum și repository-ul pentru a accesa și manipula datele din baza de date.
* View: Reprezintă interfața utilizatorului. Prin intermediul View-ului, utilizatorul interacționează cu aplicația. View-ul primește date de la Presenter și le afișează în mod corespunzător utilizatorului. De asemenea, primește evenimente de la utilizator și le trimite către Presenter pentru a fi procesate.
* Presenter: Conține logica de prezentare și acționează ca un intermediar între Model și View. Se ocupă de cererile utilizatorului primite prin View, procesează aceste cereri și interacționează cu Model-ul pentru a obține sau a actualiza datele necesare. Presenter-ul actualizează apoi View-ul cu informațiile actualizate.

**Pachetul Model**

* Clasa DatabaseConnection-subpachetul Repo:
* Este responsabilă de gestionarea conexiunii la baza de date.
* Oferă metode pentru a facilita operațiile de comunicare și manipulare a datelor din baza de date.
* Clasa UserRepo-subpachetul Repo:
* Responsabilă pentru gestionarea operațiunilor de bază legate de utilizatori în ceea ce privește interacțiunea cu baza de date.
* Aceasta include adăugarea, actualizarea, ștergerea și obținerea informațiilor despre utilizatori, precum și preluarea listelor de utilizatori.
* Clasa MazeGame:
* Reprezintă grila labirintului și include metode pentru inițializarea labirintului și manipularea pozițiilor cheie, cum ar fi morcovul, iepurașul și capcanel(plasarea obiectelor cheie și a capcanelor).
* Clasa Point:
* Reprezintă un punct într-un sistem de coordonate bidimensional și este utilizată pentru a reprezenta poziții în cadrul grilei de labirint și pentru a gestiona mișcarea iepurașului și a altor obiecte.
* Clasa User:
* Reprezintă un utilizator al aplicației și conține informații despre acesta, cum ar fi numele legal, numele de utilizator, parola, punctajul și rolul său (administrator sau nu).

**Pachetul Presenter**

* AdministratorPresenter:
* Este responsabilă pentru gestionarea interacțiunilor dintre interfața utilizatorului administrator și modelul de date.
* Aceasta include operații precum crearea, actualizarea și ștergerea utilizatorilor, precum și afișarea tuturor utilizatorilor.
* HomePagePresenter
* Această clasă acționează ca un intermediar între interfața utilizatorului de acasă și alte vederi ale aplicației, gestionând operațiile de autentificare a utilizatorului și facilitând navigarea între diferitele funcționalități ale aplicației.
* MazeGamePresenter
* Acționează ca un intermediar între interfața jocului labirint și modelul de date, gestionând logica jocului și actualizând informațiile utilizatorului în funcție de progresul și rezultatul jocului.
* Se gestionează mișcarea iepurașului și se verifică dacă e s-a câștigat sau nu jocul și afișează mesaje corespunzătoare.

**Pachetul View**

* AdminPageView-subpachetul graphicalInterfaces
* Reprezintă interfața grafică pentru pagina de administrare a aplicației.
* Ea permite administratorului să creeze, să actualizeze și să șteargă utilizatori, precum și să vizualizeze lista de utilizatori.
* HomePageView- subpachetul graphicalInterfaces
* Reprezintă interfața grafică pentru pagina principală a aplicației.
* Ea permite utilizatorilor să se autentifice și să acceseze cele 3 nivele de joc ale jocului.
* MazeGameView:
* Ea afișează labirintul și permite utilizatorului să interacționeze cu acesta pentru a muta personajul (iepurele) în labirint.
* Include butoane pentru joc, vizualizare scoruri, gestionare utilizatori (pentru administratori) și deconectare.
* AdminInterface-subpachetul interf:
* Definește metodele necesare pentru interacțiunea cu interfața de utilizator pentru funcționalitățile administrative. Aceste metode permit obținerea datelor introduse de utilizator și efectuarea operațiilor CRUD (Create, Read, Update, Delete) pe utilizatori în cadrul sistemului.
* HomePageInterface:
* Definește metodele necesare pentru interacțiunea cu interfața de utilizator pentru pagina de pornire a aplicației. Aceste metode permit obținerea datelor introduse de utilizator și afișarea mesajelor în interfață.
* MazeGameInterface:
* Definește o singură metodă necesară pentru interacțiunea cu interfața de utilizator a jocului de labirint. Această metodă permite afișarea de mesaje în interfața utilizatorului.

Cateva poze din aplicație:

Aceasta este pagina principală:

A screenshot of a computer

Description automatically generated

Pagina de Admin care apare când te loghezi ca administrator:

A screenshot of a computer

Description automatically generated

Pagina de joc:

A collage of a cat and carrots

Description automatically generated

1. **Testare**

Am implementat teste unitate (proiect de testare) corespunzătoare operațiilor de creare a bazei de date, de realizare a conexiunii la baza de date, de creare a tabelelor și de interogare a tabelelor bazei de date.

DatabaseConnectionTest verifică dacă poate fi deschisă o conexiune la baza de date, dacă putem obține o conexiune validă, verifică dacă conexiunea este închisă corect.

UserRepoTest verifică dacă se poate adăuga cu succes în baza de date un nou player, dacă putem face update pe el, dacă îl putem șterge. Se mai testează dacă se afișează toți utilizatorii în tabel corespunzător.

1. **Concluzii**

Aplicația dezvoltată în Java, cu arhitectura MVP și respectând principiile SOLID, reprezintă o soluție robustă și eficientă pentru jocul de labirint propus. Această abordare aduce numeroase beneficii, printre care:

* Mentenanță Ușoară: Prin împărțirea logică a componentelor în modele, prezentatoare și vizualizări, modificările și actualizările pot fi realizate fără a afecta alte părți ale aplicației. Astfel, se asigură o mentenanță mai ușoară și mai rapidă.
* Extensibilitate: Arhitectura MVP permite extinderea facilă a aplicației prin adăugarea de noi funcționalități sau îmbunătățiri. Clasele sunt împărțite în mod clar, ceea ce face mai ușoară gestionarea și extinderea codului.
* Reutilizabilitate: Prin utilizarea interfețelor și a claselor de bază abstracte conform principiilor SOLID, se promovează reutilizarea codului. Aceasta permite componentelor să fie adaptate și utilizate în mai multe contexte fără a rescrie codul.
* Testare Simplă: Datorită separării logicului de afaceri de interfeța grafică, este mai ușor să se testeze unitar și să se simuleze interacțiuni fără a fi necesară o interfață grafică completă. Aceasta duce la dezvoltarea unor teste mai eficiente și mai fiabile.