Titlu proiect: The Blue Wizard Autor: Bardasu Ioan Alexandru Grupa: 1211B

***Poveste jocului:***

În lumea magică a regatului Urdonia, un rău înfiorător a căzut peste pământuri.

Vrăjitorul Albastru, un maestru al artei magice, se trezește în mijlocul acestui haos când fiica sa, prințesa Amara, este răpită de o forță întunecată. Determinat să-și recupereze fiica și să aducă lumină în întuneric, Vrăjitorul Albastru pornește într-o călătorie periculoasă și plină de aventuri.

În acest joc de acțiune și aventură, jucătorii preiau controlul asupra Vrăjitorului Albastru în misiunea sa epică de a înfrunta armatele de scheleți malefici și de a-și elibera fiica. Pe măsură ce avansează în joc, Vrăjitorul Albastru își descoperă puterile magice și învață noi vrăji pentru a distruge inamicii din calea lui.

Regatul Urdonia este plin de mistere, dar si pericole greu de infruntat, unde jucatorul va descoperi taramuri nemaivazute de ochiul uman pana acum. De la Padurea Mortilor, acesta va trece prin diferite lumi precum: Taramul Pinguinilor si Taramul Regelui Inflacarat. Pe parcursul calatoriei sale, acesta va strange experienta cu care v-a putea sa isi imbunatateasca abilitatile.

In final, numai dexteritatea si inteligenta jucatorului il vor ajuta pe Vrajitorul Albastru sa isi recupereze fiica mult iubita si sa restabileasca pacea in Regatul sau luminat.

***Regulile si scopul jocului:***

Jocul implica deplasarea personajului pe o harta in care se vor gasi diferite

obstacole, dar si inamici care vor incerca sa distruga personajul. Scopul jocului este ca jucatorul sa reuseasca sa omoare toti inamicii din nivel prin diferite mijloace pentru a putea trece la urmatorul nivel si pentru a primi puncte de experiente benefice pentru dezvoltarea de atacuri noi. Nivelul este finalizat in momentul in care nu mai exista niciun inamic in viata. Jocul se termina cand jucatorul reuseste sa distruga boss-ul de la ultimul nivel.

- Deplasare

W

A

S

D

* Atac

SpaceBar

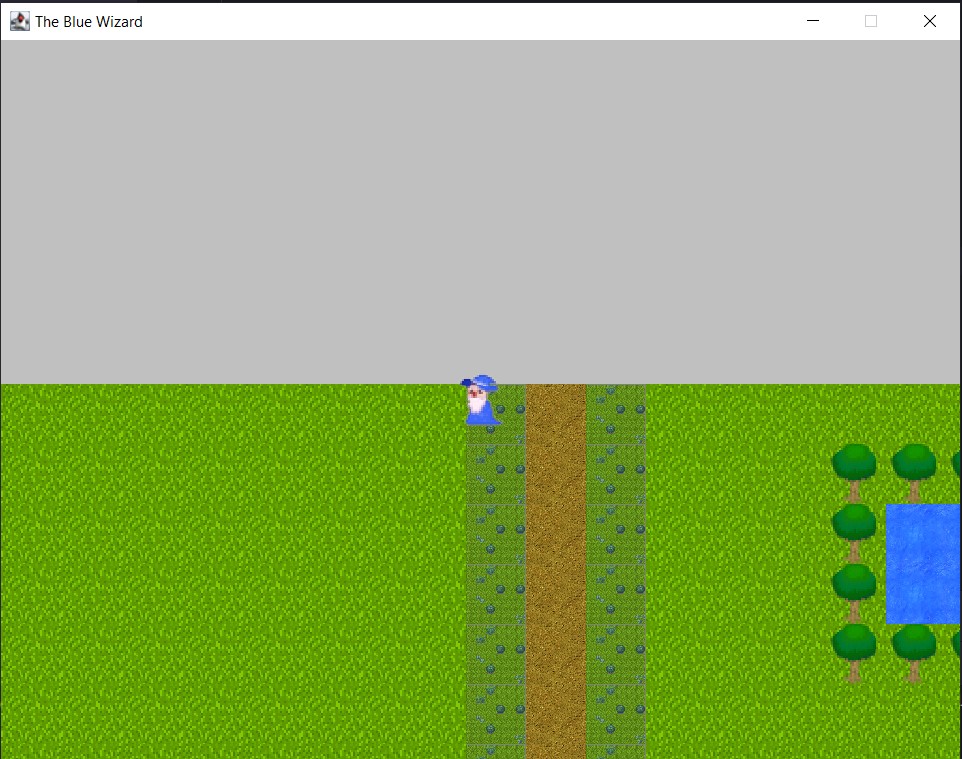
* Mareste viteza personajului

Shift

***Proiectarea nivelelor:***

Jocul va avea 3 nivele. Primul nivel se va afla intr-un peisaj de padure plina de verdeata. Aici personajul principal se va intalni cu primul set de inamici: Soldatii din regatul Intunecat. Aici, personajul va trece pe langa Raul Tamaduirii, printre copacii si iarba deasa, care il va ascunde pe vrajitor de inamicii rai. Al doilea nivel va avea loc in Taramul Pinguinilor, unde inghetul si zapada il impiedica din a isi indeplini misiunea. Chiar daca acesta trece de aceste obstacole ale naturii, el se va intalnii cu Pinguinii Roiali, care doresc sa isi distruga orice intrus din taramul lor. Ultimul nivel va fi in Taramul Regelui Inflacarat, unde personajul va da de ultimul set de inamici: Soldatii

Focului. In final, dupa ce a eliminat toti inamicii, acesta va trebui sa infrunte marele Sef al Soldatilor Focului, Regele Inflacarat.



In aceasta imagine este personajul care se afla pe mapa primului nivel. Aceasta este o schita a proiectului. Ulterioare modificari pot aparea. Copacii, apa, o parte din iarba si drumul vor fi TILE-uri, iar restul ierbii va fi fundalul jocului. Personajele se vor regasi peste tot pe harta, iar toate entitatile vor avea coliziuni cu diferitele obiecte precum: pietrele si copacii.

GameSprite-uri ale jocului:

***Personaj principal:***



***Atacul personajului:***



***Tipul de inamici:***

***Cartoon of a skeleton with a helmet and swords

Description automatically generated***

***GUI:***

***Diagrama UML:***

A black and white image of a computer screen

Description automatically generated with medium confidence

***Poze din joc:***

***A screenshot of a video game

Description automatically generated***

***A screenshot of a video game

Description automatically generated***

***A screenshot of a video game

Description automatically generated***

***A screenshot of a video game

Description automatically generated***

***Explicarea claselor si metodelor:***

Clasa Entity:

***Entity este o clasă de bază pentru toate entitățile din joc (cum ar fi jucătorul, inamicii, obiectele, etc.). Aceasta conține toate proprietățile și metodele comune care ar putea fi necesare pentru gestionarea entităților.***

***Proprietăți***

***GamePanel gp: Referință la obiectul GamePanel. Este folosit pentru a accesa diferite componente ale jocului, cum ar fi verificatorul de coliziuni (CollisionChecker), UI-ul, etc.***

***int worldX, worldY: Coordonatele entității în lumea jocului.***

***int speed: Viteza de mișcare a entității.***

***BufferedImage left1, left2, right1, right2: Imagini pentru diferite stări de animație ale entității. Acestea sunt folosite pentru a reda animațiile de mișcare în diferite direcții.***

***Rectangle solidArea: Zona de coliziune a entității. Este un dreptunghi folosit pentru a determina coliziunile cu alte obiecte sau entități.***

***int solidAreaDefaultX, solidAreaDefaultY: Pozițiile implicite ale zonei de coliziune.***

***boolean collisionOn: Indică dacă entitatea este într-o coliziune.***

***int actionLockCounter: Contor pentru blocarea acțiunilor, util pentru sincronizarea anumitor acțiuni.***

***String dialogues[]: Un array de dialoguri pe care entitatea le poate spune.***

***int dialogueIndex: Indexul curent al dialogului.***

***int maxLife: Viața maximă a entității.***

***int life: Viața curentă a entității.***

***int spriteCounter: Contor pentru schimbarea sprite-urilor (pentru animații).***

***int spriteNum: Numărul curent al sprite-ului.***

***Metode***

***Entity(GamePanel gp): Constructorul clasei Entity. Inițializează referința la GamePanel.***

***void setAction(): O metodă abstractă destinată să fie suprascrisă în clasele derivate. Este folosită pentru a seta acțiunile entității.***

***void speak(): Metoda care face entitatea să vorbească. Setează dialogul curent în UI și incrementează indexul dialogului.***

***void update():***

***Apelează setAction() pentru a seta acțiunile curente.***

***Resetează collisionOn la false.***

***Verifică coliziunile cu tile-urile, obiectele și jucătorul folosind CollisionChecker.***

***Dacă nu există coliziuni, actualizează poziția entității în funcție de direcție și viteză.***

***Gestionează animațiile prin incrementarea spriteCounter și alternarea între spriteNum 1 și 2 la fiecare 16 cadre.***

***void draw(Graphics2D g2):***

***Desenează entitatea pe ecran.***

***Calculează poziția pe ecran a entității în funcție de poziția sa în lumea jocului și de poziția jucătorului.***

***Verifică dacă entitatea este în raza de desenare a jucătorului și desenează sprite-ul corespunzător direcției și numărului sprite-ului.***

***Clasa NPC\_Dragon***

***NPC\_Dragon este o clasă derivată din Entity, creată pentru a reprezenta un NPC de tip dragon în joc. Aceasta include funcționalități pentru a seta imagini, dialoguri și acțiuni specifice dragonului.***

***Constructorul***

***Constructorul: Initializează dragonul setând direcția inițială în jos, viteza la 1 și apelând metodele pentru a încărca imaginile și dialogurile dragonului.***

***Metode***

***getImage***

***Încarcă imaginile necesare pentru animarea dragonului atunci când acesta se mișcă în diferite direcții (stânga și dreapta).***

***setDialogue***

***Configurează dialogurile pe care dragonul le va rosti atunci când interacționează cu jucătorul. În acest caz, dragonul oferă instrucțiuni și sfaturi pentru jucător.***

***setAction***

***Controlează comportamentul dragonului, schimbând direcția acestuia la intervale regulate de timp folosind un mecanism aleator. Acest lucru face ca dragonul să se miște în mod aleatoriu în jurul hărții.***

***speak***

***Suprapune metoda speak din clasa Entity pentru a permite dragonului să rostească dialogurile configurate. Aceasta este o metodă de comportament care poate fi extinsă pentru dialoguri specifice dragonului.***

***Clasa Player***

***Variabile de instanță:***

***GamePanel gp: Referință către panoul de joc în care se află jucătorul.***

***KeyHandler keyH: Un obiect care gestionează intrările de la tastatură pentru jucător.***

***int hasCoins: Numărul de monede pe care jucătorul le-a colectat.***

***int defaultSpeed: Viteza implicită a jucătorului.***

***final int screenX, screenY: Coordonatele ecranului pe care se află jucătorul.***

***Constructor:***

***Primește referințe către GamePanel și KeyHandler.***

***Inițializează variabilele și apelurile metodei pentru setarea imaginilor și valorilor implicite ale jucătorului.***

***Metoda setDefaultValues():***

***Inițializează valorile implicite ale jucătorului, cum ar fi poziția inițială, viteza și direcția.***

***Metoda getPlayerImage():***

***Încarcă imaginile pentru jucător din fișierele specifice.***

***Metoda update():***

***Actualizează poziția jucătorului pe baza intrărilor de la tastatură și a coliziunilor.***

***Verifică dacă jucătorul interacționează cu alte obiecte sau NPC-uri.***

***Metoda interactNPC(int i):***

***Gestionează interacțiunea cu NPC-urile, inclusiv trecerea la starea de dialog dacă este cazul.***

***Metoda pickUpObject(int i):***

***Gestionează colectarea obiectelor și efectele acestora, cum ar fi creșterea vieții sau vitezei jucătorului.***

***Metoda draw(Graphics2D g2):***

***Desenează jucătorul pe ecran, selectând imaginea corespunzătoare în funcție de direcția și animația acestuia.***

***Aceste metode lucrează împreună pentru a controla comportamentul și aspectul jucătorului în joc.***

***Clasa AssetSetter este responsabilă pentru inițializarea și setarea obiectelor (obiecte, NPC-uri și monștri) în funcție de nivelul actual al jocului.***

***Metoda setObject()***

***Inițializează și plasează obiectele în funcție de nivelul curent al jocului (gp.level).***

***Pentru fiecare nivel, se setează pozițiile și tipurile obiectelor, cum ar fi bocanci, inimi, monede și comoară.***

***Pozițiile sunt stabilite în funcție de coordonatele hărții (multiplicate cu dimensiunea unui tile).***

***Metoda setNPC()***

***Inițializează și plasează NPC-urile în funcție de nivelul curent al jocului (gp.level).***

***În acest caz, este un singur NPC, un dragon, plasat la o anumită poziție pe hartă.***

***Metoda setMonster(int lvl)***

***Inițializează și plasează monștrii în funcție de nivelul curent al jocului (lvl).***

***În funcție de nivel, se creează și se plasează diferiți monștri, cum ar fi scheletele, pinguinii sau minotaurii, la poziții prestabilite.***

***Aceste metode asigură setarea inițială a tuturor entităților din joc înainte ca acesta să înceapă și sunt esențiale pentru crearea și gestionarea mediului de joc.***

***Clasa CollisionChecker este responsabilă pentru verificarea coliziunilor dintre entități în joc, cum ar fi jucătorul, obiectele și monștrii, cu harta și între ele.***

***Metoda checkTile(Entity entity)***

***Această metodă verifică coliziunile între entități și părțile solide ale hărții, reprezentate de dalele hărții.***

***Se calculează coordonatele colțurilor entității în raport cu harta și se determină numărul dalei din harta care corespunde fiecărui colț.***

***Se verifică dacă dalele respective au coliziune (collision == true) și se setează un indicator de coliziune pentru entitatea dată, în funcție de direcția sa de mișcare.***

***Metoda checkObject(Entity entity, boolean player)***

***Această metodă verifică coliziunile între entitatea dată și obiectele din joc, cum ar fi monede, inimi, bocanci sau comoară.***

***Se iterează prin toate obiectele din joc și se verifică dacă entitatea se intersectează cu obiectul respectiv.***

***Dacă entitatea este jucătorul (player == true), se returnează indexul obiectului cu care s-a produs coliziunea.***

***Metoda checkEntity(Entity entity, Entity[] target)***

***Această metodă verifică coliziunile între entitatea dată și alte entități din joc, cum ar fi monștrii sau NPC-urile.***

***Similar cu checkObject, se iterează prin toate entitățile dintr-un vector dat și se verifică dacă entitatea dată se intersectează cu entitatea respectivă.***

***Dacă se găsește o coliziune, se returnează indexul entității cu care s-a produs coliziunea.***

***Metoda checkPlayer(Entity entity)***

***Această metodă verifică coliziunile între entitatea dată și jucător.***

***Similar cu celelalte metode, se calculează coliziunile și se verifică dacă entitatea se intersectează cu jucătorul.***

***Această metodă este folosită pentru verificarea coliziunilor între monștri și jucător.***

***În esență, clasa CollisionChecker facilitează detectarea și gestionarea coliziunilor dintre diverse entități din joc și elementele de mediu, contribuind la corectitudinea și realitatea experienței de joc.***

***Clasa DataBaseManager***

***Această clasă DatabaseManager este responsabilă pentru gestionarea interacțiunilor cu baza de date SQLite asociată jocului tău. Iată o scurtă explicație a fiecărei metode:***

***Metoda connect()***

***Este utilizată pentru a stabili conexiunea către baza de date SQLite.***

***Se încarcă driverul JDBC pentru SQLite (org.sqlite.JDBC).***

***Se specifică URL-ul bazei de date (jdbc:sqlite:ProiectPAOO/src/game.db).***

***Dacă conexiunea este realizată cu succes, se afișează un mesaj de confirmare.***

***Metoda createTableIfNotExists()***

***Verifică dacă tabela player există în baza de date. Dacă nu există, o creează.***

***Structura tabelului include un câmp id care este cheia primară și se auto-incrementează, și un câmp level care reține nivelul jucătorului.***

***Dacă tabela este creată cu succes, se afișează un mesaj de confirmare.***

***Metoda savePlayerState(int level)***

***Salvează starea jucătorului în baza de date.***

***Folosește o instrucțiune SQL de tip INSERT pentru a adăuga un nou rând în tabela player cu nivelul jucătorului dat ca argument.***

***Nu afișează mesaje în caz de succes sau eșec.***

***Metoda loadPlayerState()***

***Încarcă starea jucătorului din baza de date.***

***Folosește o instrucțiune SQL de tip SELECT pentru a obține ultimul rând din tabela player, ordonat după id în ordine descrescătoare și limitat la un singur rând.***

***Creează un obiect PlayerState cu nivelul jucătorului încărcat din baza de date și îl returnează.***

***În caz de eșec, afișează mesajul de eroare.***

***Această clasă este concepută pentru a facilita salvarea și încărcarea stării jucătorului în/din baza de date, permițând astfel persistența datelor jocului chiar și între sesiunile de joc.***

***Clasa GamePanel***

***Această clasă GamePanel este o componentă crucială a jocului tău și este responsabilă pentru gestionarea afișării, actualizării și interacțiunii cu utilizatorul. Iată o explicație a membrilor și metodelor sale:***

***Membri:***

***Variabile pentru setările ecranului: Aceste variabile definesc dimensiunile și alte aspecte ale ecranului de joc, cum ar fi dimensiunea tile-urilor și dimensiunea ecranului.***

***Variabile pentru setările lumii: Aceste variabile definesc dimensiunile și limitele lumii de joc.***

***FPS: Variabila pentru gestionarea cadrelor pe secundă (Frames Per Second).***

***Obiecte pentru gestionarea sistemului: tileM, keyH, music, se, cChecker, aSetter și ui sunt obiecte pentru gestionarea hărții, tastelor, sunetului, verificării coliziunilor, configurării activelor și interfeței utilizatorului.***

***Thread-ul de joc: gameThread este un fir de execuție care gestionează bucla principală de actualizare și desenare a jocului.***

***Variabile de stare a jocului: Aceste variabile rețin starea curentă a jocului, cum ar fi starea de titlu, de joc, de pauză, de dialog și de finalizare.***

***Altele: Există variabile pentru nivel, numărul maxim de inimi și obiecte pentru jucător, NPC-uri și monștri.***

***Metode:***

***GamePanel(): Constructorul clasei. Inițializează dimensiunile și aspectele ecranului și configurează interacțiunea cu tastatura.***

***setupGame(): Inițializează jocul, încărcând harta și configurând obiectele.***

***saveGame(): Salvează starea jocului în baza de date folosind DatabaseManager.***

***startGameThread(): Pornirea firului de execuție pentru actualizarea și desenarea jocului.***

***run(): Metoda de execuție principală a firului de execuție. Actualizează și desenează jocul într-o buclă continuă.***

***update(): Actualizează starea jocului în funcție de starea curentă.***

***paintComponent(Graphics g): Desenează componentele jocului pe ecran în funcție de starea curentă a jocului.***

***playMusic(int i): Redă o melodie specifică folosind obiectul music.***

***stopMusic(): Oprește redarea muzicii.***

***playSE(int i): Redă un efect de sunet specific folosind obiectul se.***

***Această clasă centralizează toate aspectele jocului și facilitează interacțiunea între diferitele sale componente pentru a oferi o experiență de joc captivantă și plăcută.***

***Clasa KeyHandler***

***Clasa KeyHandler este responsabilă pentru gestionarea evenimentelor legate de tastatură în cadrul jocului tău. Ea implementează interfața KeyListener și oferă metode pentru a trata evenimentele de apăsare și eliberare a tastelor.***

***Membri:***

***Variabile pentru starea tastelor: Aceste variabile, cum ar fi upPressed, downPressed, leftPressed, rightPressed și shiftPressed, indică dacă o anumită tastă este apăsată într-un moment dat. Ele sunt utilizate pentru a controla mișcarea și alte acțiuni ale jucătorului în joc.***

***Obiectul gp: Acesta este un obiect de tip GamePanel care este asociat cu acest KeyHandler. Este folosit pentru a accesa starea și metodele din cadrul GamePanel.***

***Metode:***

***keyPressed(KeyEvent e): Această metodă este apelată atunci când o tastă este apăsată. În funcție de starea jocului (gp.gameState), aceasta interpretează acțiunea tastelor și răspunde corespunzător. De exemplu, în timpul stării de titlu, direcția tastelor se schimbă între opțiunile meniului, în timp ce în timpul jocului, apăsarea tastelor W, A, S, D determină mișcarea jucătorului și apăsarea tastei SHIFT poate modifica viteza acestuia. De asemenea, tastă P este folosită pentru a trece între stările de joc și de pauză, iar tastă Y este folosită pentru a salva jocul în starea de pauză.***

***keyReleased(KeyEvent e): Această metodă este apelată atunci când o tastă este eliberată. Ea actualizează variabilele care țin evidența stării tastelor apăsate.***

***keyTyped(KeyEvent e): Această metodă este apelată atunci când o tastă este tastată. În implementarea actuală, această metodă nu este utilizată.***

***Această clasă joacă un rol esențial în interacțiunea dintre utilizator și joc, permițând jucătorului să controleze acțiunile din lumea jocului utilizând tastatura. Ea conectează interacțiunea utilizatorului cu logica de joc din cadrul GamePanel, permitând astfel jocului să răspundă la acțiunile jucătorului într-un mod dinamic și interactiv.***

***Main***

***Clasa Main este punctul de intrare în aplicația ta și este responsabilă pentru crearea ferestrei principale a jocului și inițierea acestuia.***

***Metoda main(String[] args):***

***Aceasta este metoda principală care este invocată când se rulează programul.***

***Se creează un obiect de tip JFrame pentru a reprezenta fereastra jocului.***

***Se setează comportamentul de închidere a ferestrei pentru a opri aplicația când fereastra este închisă.***

***Se setează faptul că fereastra nu poate fi redimensionată.***

***Se stabilește titlul ferestrei ca "The Blue Wizard".***

***Se creează un obiect de tip GamePanel, care este panoul principal al jocului.***

***Panoul jocului este adăugat la fereastra principală.***

***Se folosește metoda pack() pentru a asigura că dimensiunile ferestrei se ajustează automat la conținutul său.***

***Fereastra este plasată în centrul ecranului folosind metoda setLocationRelativeTo(null).***

***Fereastra este făcută vizibilă apelând metoda setVisible(true).***

***Este demarat firul de execuție al jocului prin apelul metodei startGameThread() a obiectului gamePanel, astfel încât logica jocului să înceapă să ruleze.***

***Clasa PlayerState***

***Clasa PlayerState reprezintă starea jucătorului în ceea ce privește nivelul acestuia în joc. Această clasă are un singur membru de date:***

***level: Un număr întreg care indică nivelul la care se află jucătorul în joc.***

***Clasa PlayerState are un constructor care primește nivelul și îl setează pe membrul level. Acest constructor este folosit pentru a crea obiecte PlayerState cu nivelul specificat.***

***Constructorul:***

***public PlayerState(int level): Acesta este constructorul clasei PlayerState. Primește ca parametru nivelul jucătorului și îl atribuie membrului level al obiectului curent. Acest constructor este utilizat pentru a inițializa obiecte PlayerState cu nivelul specificat.***

***Clasa Sound***

***Clasa Sound este responsabilă pentru gestionarea sunetelor în joc. Aceasta folosește biblioteca Java Sound API pentru a încărca și reda fișiere audio. Iată o explicație a membrilor și metodelor clasei:***

***Membri de date:***

***Clip clip: Reprezintă un clip audio, utilizat pentru redarea sunetului.***

***URL[] soundURL: Un vector care conține URL-urile fișierelor audio.***

***Constructor:***

***public Sound(): Constructorul clasei inițializează vectorul soundURL cu resursele fișierelor audio. Aceste resurse sunt localizate în directorul /sound al proiectului.***

***Metode:***

***public void setFile(int i): Această metodă încarcă un fișier audio specificat de indexul i în clipul audio clip. Fișierul audio este obținut din vectorul soundURL.***

***public void play(): Metoda play() redă clipul audio.***

***public void loop(): Metoda loop() redă clipul audio în mod repetat (buclează redarea).***

***public void stop(): Metoda stop() oprește redarea clipului audio.***

***Această clasă facilitează manipularea și redarea sunetelor în joc, oferind metode simple pentru încărcarea, redarea, buclarea și oprirea sunetelor.***

***Clasa UI***

***Clasa UI este responsabilă pentru afișarea interfeței utilizatorului în joc, inclusiv ecranul de titlu, ecranul de pauză, ecranul de dialog și alte elemente de UI. Iată o explicație a membrilor și metodelor clasei:***

***Membri de date:***

***GamePanel gp: Referință către obiectul GamePanel asociat, utilizat pentru a accesa starea și datele jocului.***

***Graphics2D g2: Contextul grafic 2D folosit pentru desenarea elementelor UI.***

***BufferedImage heart\_full, heart\_3v4, heart\_half, heart\_1v4, heart\_blank: Imaginile pentru diferitele stări ale inimii.***

***Font arial\_40, arial\_70B: Fonturile utilizate pentru textul afișat.***

***BufferedImage HeartImage: Imaginea inimii.***

***BufferedImage CoinImage: Imaginea monedei.***

***boolean messageOn: Indicator pentru afișarea unui mesaj.***

***String message: Mesajul care urmează să fie afișat.***

***String currentDialogue: Textul dialogului curent.***

***int commandNum: Numărul comenzii selectate în ecranul de titlu.***

***static double playTime: Timpul total de joc.***

***DecimalFormat dFormat: Format pentru afișarea timpului de joc.***

***Constructor:***

***public UI(GamePanel gp): Constructorul inițializează membrii și încarcă imaginile necesare pentru inimă și monedă.***

***Metode:***

***public void showMessage(String text): Metoda afișează un mesaj pe ecran.***

***public void draw(Graphics2D g2): Metoda principală de desenare a UI-ului, determină ce ecran trebuie afișat și desenează elementele corespunzătoare.***

***private void drawLife(GamePanel gp): Metoda desenează starea de viață a jucătorului (numărul de inimi) pe ecran.***

***private void drawTitleScreen(): Metoda desenează ecranul de titlu al jocului.***

***public void drawPauseScreen(): Metoda desenează ecranul de pauză.***

***public void drawDialogueScreen(): Metoda desenează ecranul de dialog.***

***public void drawSubWindow(int x, int y, int width, int height): Metoda desenează un sub-fereastră utilizată pentru dialog sau alte scopuri.***

***public int getXforCenteredText(String text): Metoda calculează coordonata x pentru a centra textul pe ecran.***

***Această clasă gestionează afișarea elementelor de UI și interacțiunile cu utilizatorul în joc, folosind diverse metode pentru a desena ecranele și elementele UI în mod corespunzător.***

***Clasa MON\_Minotaur***

***Clasa MON\_Minotaur este o subclasă a clasei Entity și reprezintă un tip specific de monstru în joc. Iată o explicație a membrilor și metodelor clasei:***

***Constructor:***

***public MON\_Minotaur(GamePanel gp): Constructorul clasei inițializează caracteristicile specifice ale monstrului Minotaur, cum ar fi numele, direcția de mișcare, viteză, viața maximă și imaginea sa. De asemenea, setează zona solidă a monstrului.***

***Metode:***

***public void getImage(): Metoda încarcă imaginile necesare pentru animația de mers a Minotaurului (stânga și dreapta).***

***public void setAction(): Metoda determină acțiunile pe care Minotaurul le va efectua în timpul jocului. În acest caz, Minotaurul are un comportament aleatoriu, ale cărui direcții de mișcare sunt generate aleatoriu cu o probabilitate de 25% pentru fiecare direcție (sus, jos, stânga, dreapta).***

***Această clasă definește comportamentul specific al Minotaurului în joc, inclusiv modul în care se mișcă și reacționează în diferite situații. Ea extinde funcționalitatea clasei de bază Entity, care gestionează entitățile din joc.***

***Clasa MON\_Penguin***

***Clasa MON\_Penguin este o subclasă a clasei Entity și reprezintă un alt tip de monstru în joc. Iată o explicație a membrilor și metodelor clasei:***

***Constructor:***

***public MON\_Penguin(GamePanel gp): Constructorul clasei inițializează caracteristicile specifice ale monstrului Penguin, cum ar fi numele, direcția de mișcare, viteză, viața maximă și imaginea sa. De asemenea, setează zona solidă a monstrului.***

***Metode:***

***public void getImage(): Metoda încarcă imaginile necesare pentru animația de mers a Penguinului (stânga și dreapta).***

***public void setAction(): Metoda determină acțiunile pe care Penguinul le va efectua în timpul jocului. În acest caz, Penguinul are un comportament aleatoriu, ale cărui direcții de mișcare sunt generate aleatoriu cu o probabilitate de 25% pentru fiecare direcție (sus, jos, stânga, dreapta).***

***Această clasă definește comportamentul specific al Penguinului în joc, inclusiv modul în care se mișcă și reacționează în diferite situații. Ea extinde funcționalitatea clasei de bază Entity, care gestionează entitățile din joc.***

***Clasa MON\_Skeleton***

***Clasa MON\_Skeleton este o subclasă a clasei Entity și reprezintă un tip de monstru specific în joc. Iată o explicație a membrilor și metodelor clasei:***

***Constructor:***

***public MON\_Skeleton(GamePanel gp): Constructorul clasei inițializează caracteristicile specifice ale monstrului Skeleton, cum ar fi numele, direcția de mișcare, viteză, viața maximă și imaginea sa. De asemenea, setează zona solidă a monstrului.***

***Metode:***

***public void getImage(): Metoda încarcă imaginile necesare pentru animația de mers a Skeletonului (stânga și dreapta).***

***public void setAction(): Metoda determină acțiunile pe care Skeletonul le va efectua în timpul jocului. În acest caz, Skeletonul are un comportament aleatoriu, ale cărui direcții de mișcare sunt generate aleatoriu cu o probabilitate de 25% pentru fiecare direcție (sus, jos, stânga, dreapta).***

***Această clasă definește comportamentul specific al Skeletonului în joc, inclusiv modul în care se mișcă și reacționează în diferite situații. Ea extinde funcționalitatea clasei de bază Entity, care gestionează entitățile din joc.***

***Clasa TileManager***

***Clasa TileManager este responsabilă pentru gestionarea și desenarea pieselor de teren din joc. Iată o explicație a membrilor și metodelor clasei:***

***Constructor:***

***public TileManager(GamePanel gp): Constructorul primește o referință către obiectul GamePanel și inițializează câteva variabile și apoi încarcă imaginea pentru fiecare tip de piesă de teren, în funcție de nivelul actual al jocului.***

***Metode:***

***public void getTileImage(): Această metodă încarcă imaginile necesare pentru fiecare tip de piesă de teren, în funcție de nivelul jocului. Imaginile sunt stocate într-un tablou de obiecte Tile, unde fiecare element reprezintă o imagine specifică.***

***public void loadMap(String filePath): Această metodă încarcă harta de joc dintr-un fișier text specificat prin filePath. Harta este stocată într-un tablou bidimensional de numere întregi, mapTileNum, care indică tipul de piesă de teren pentru fiecare poziție din hartă.***

***public void draw(Graphics2D g2): Această metodă desenează piesele de teren pe ecran, folosind informații despre poziția relativă a jucătorului față de harta de joc. Ea parcurge fiecare poziție din harta de joc și desenează imaginea corespunzătoare pentru fiecare tip de piesă de teren.***

***Clasa TileManager este esențială pentru afișarea și gestionarea aspectului vizual al lumii de joc și interacțiunile dintre jucător și mediul înconjurător.***