

**Claurențiu - Aventura
Raimond-Eduard
Butnaru
1207B**

Povestea jocului:

Odată, într-un regat îndepărtat, exista un tânăr pe nume Claurențiu, care trăia într-un sat micuț. El era un băiat plin de viață și aventuri, iar cea mai mare dorință a lui era să devină un erou celebru.

Intr-o zi, când mergea spre casa lui, a găsit un obiect strălucitor în mijlocul drumului. Era un pepene stralucitor, care, după ce a fost atins de Claurențiu, l-a teleportat într-o lume magică.

În această lume, el a descoperit un regat fermecat, care era locuit de personaje surprinzătoare. În mijlocul regatului, era un castel mare și misterios, în care locuia o prințesă frumoasă numită Aurora, care a fost răpită de un vrăjitor rău. Claurențiu a primit misiunea de a o salva pe prințesă și a început căutările pentru a o găsi.

Pe parcursul aventurii sale, Claurențiu a întâlnit diferite creaturi fantastice și a trecut prin diverse peisaje periculoase, cum ar fi păduri întunecate, munți înzăpeziți și lacuri adânci. În timpul aventurii sale, el a colectat bănuți și puteri magice, care i-au fost de mare ajutor în lupta cu vrăjitorul rău și armata sa de monștri.

Prezentare joc:

Campania jocului constă în aventura lui Claurențiu care încearcă să ajungă la capătul fiecărui nivel, căutând prințesa, evident, pe parcursul drumului său vor apărea obstacole și inamici, el trebuie să depășească aceste obstacole și să învingă acești inamici pentru a ajunge la obiectul lui suprem și pentru a înfrânge vrăjitorul malefic.

Reguli joc:

Jocul implică deplasarea personajului de la stânga la dreapta ecranului, depășind obstacolele sărind peste acestea, obstacolele mai înalte sau gropile mai

lungi necesită un salt mai puternic. Inamicii pot fi atacați prin acțiunea de atac al lui Clăușențiu, diferiții inamici pot încasa de la cel puțin o lovitură până la cât este nevoie ca un inamic să fie considerat doborât.

Jucătorul pierde atunci când „moare”, moartea apare atunci când Clăușențiu pierde toate punctele de viață sau când acesta cade în afara hărții - adică în așa zisele gropi.

Personajele jocului:

Clăușențiu - protagonistul și jucătorul personaj. El este eroul acestei povești. Un tânăr athletic care poate sări foarte sus. Cunoscut și pentru inteligența sa, el este capabil de depăși strategic inamicii.



Goblinii portocalii - inamici de baza, pot fi loviți direct însă dacă aceștia intră în contact cu personajul principal, acesta va pierde puncte de viață. Stilul lor de mișcare este, în mod normal, de la dreapta la stanga, întorcându-se dacă în drumul lor apare vreun zid sau obstacol.



Păianjenii uriași - acești inamici sunt un soi de păianjeni de care apar să fie aproape de înălțimea protagonistului. Aceștia au țesut pânze pe care atârână și încurcă progresul protagonistului.



Demonii - Demonii sunt cea mai puternică și dezvoltată categorie de inamici, aceștia sunt foarte teritoriali, iar când un om se apropie de teritoriul lor, aceștia devin

mai furioși și încep să se miște mult mai rapid decât ar face în condiții normale



Obiecte de colectat - singurul obiect pe care îl poate colecta Clăușențiu în aventura sa, sunt inimile, care, odată colectate, acestea îl pot vindeca pe Clăușențiu, dacă acesta a fost rănit în prealabil, sau îi pot crește scorul dacă acesta nu are nevoie de vindecare.



Tabla de joc :

- Terenul care menține jucătorul și inamicii.
- Blocuri plutitoare, care permit jucătorului să treacă peste gropi foarte mari și greu de sărit peste ele
- Pe hartă se mai pot vedea și pânze de păianjen care susțin păianjenii

Mecanica jocului :

Jocul constă în aventura lui Clăușențiu spre castel, acesta trebuie să lupte cu inamicii și să depășească obstacolele.

Controale:

- Tastele „Săgeată stânga” și „Săgeată dreapta” vor fi folosite pentru a deplasa caracterul, pe orizontala, fie spre stânga, fie spre dreapta
- Tasta W va permite jucătorului să sară.
- Tasta R - face ca protagonistul să atace în proximitatea sa, inamicii slabi pot fi eliminați aproape instant.

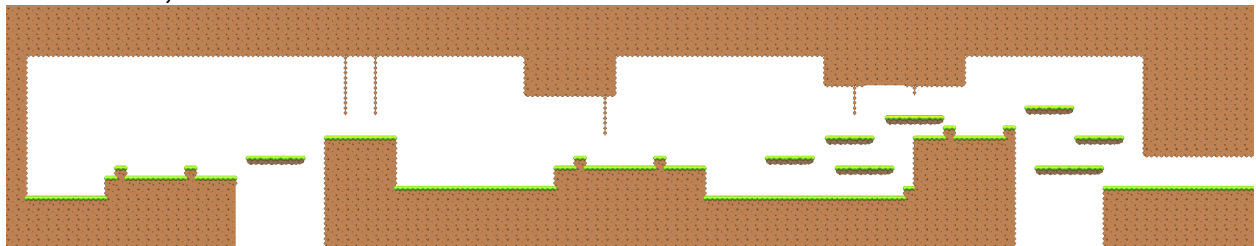
Interacțiuni:

- Atingerea directă a inamicilor rezultă daune asupra barei de viață a jucătorului, daunele pot varia în funcție de gravitatea atacului primit.
- Atacarea inamicilor prin diferitele metode de atac rezultă rănirea lor, urmată mai apoi de eliminarea lor.
- Atingerea inimilor va crește nivelul barei de viață a jucătorului până la maxim, orice inimă colectată peste nivelul maxim al barei de viață va contribui la scor.

Game sprite



Harta inițială a nivelului 3:



Nivelul 1:

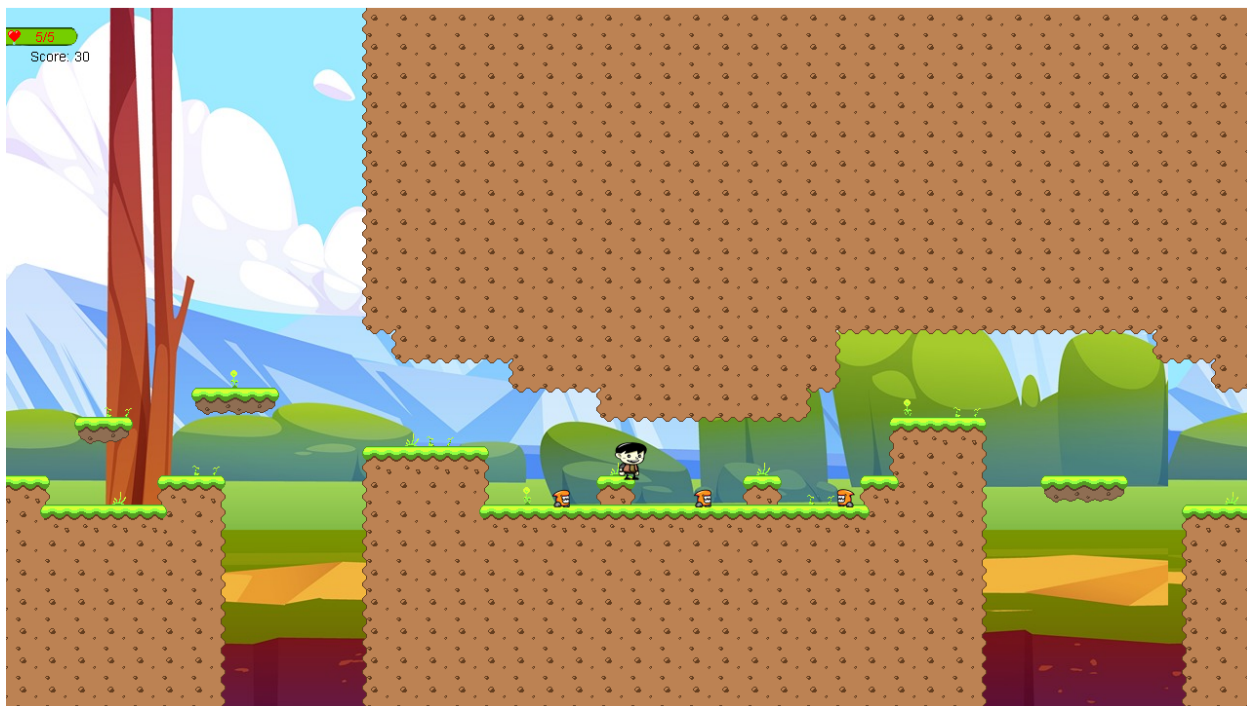
Primul nivel are loc într-o pădure care duce spre castel, acest nivel este de dificultate ușoară, deoarece are ca scop obișnuirea jucătorului cu mecanicile jocului, deplasarea, saltul, atacuri și interacțiunile cu inamicii.

Deplasarea se face de la stânga la dreapta, iar inamicii tind să se deplaseze spre jucător.

Nivelul se încheie când Clăușu ajunge la capătul cărării, continuând drumul spre castel.

Screenshot-uri din diferite ipostaze ale nivelului 1:





Nivelul 2:

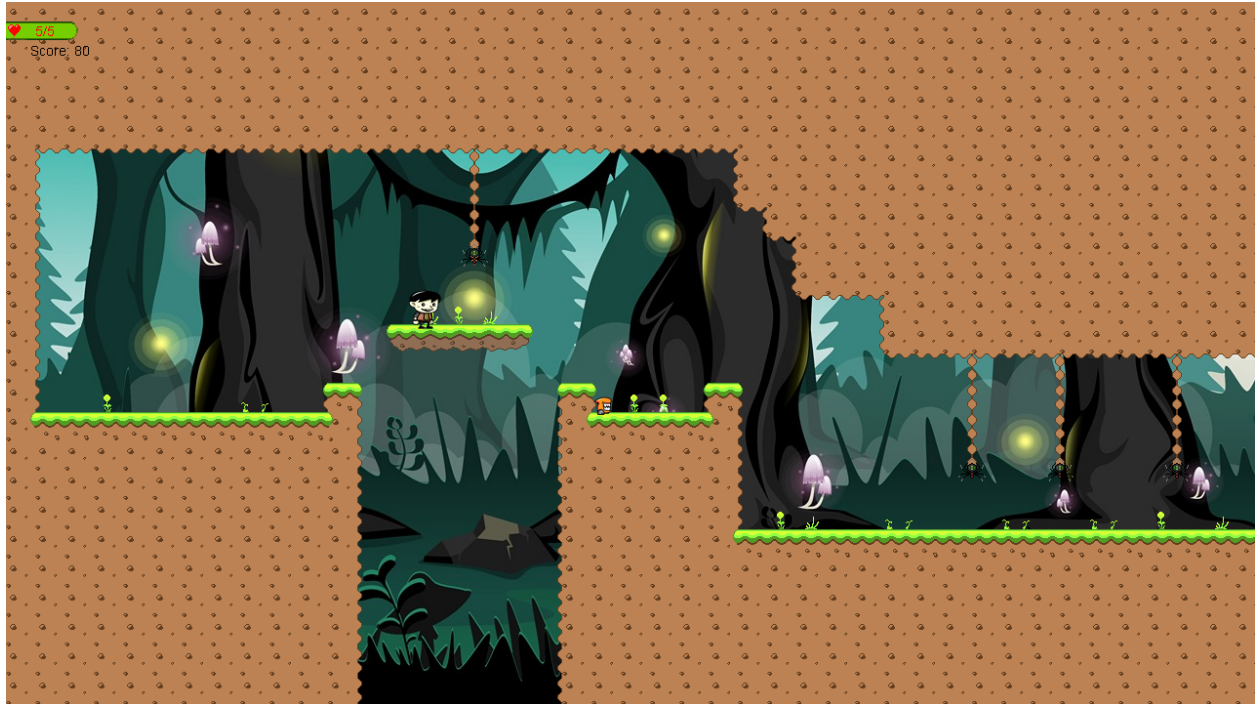
Al doilea nivel are loc noaptea, în aceeași pădure, unde apar capcane și inamicii devin mai agresivi, aceștia apar în număr mai mare față de primul nivel.

Sunt mai multe gropi în acest nivel, stimulând jucătorul să fie mai atent la ce face.

Totodată, în acest nivel apar și păianjenii.

Acest nivel se încheie atunci când Claurențiu ajunge la capătul cărării întunecate.

Screenshot-uri cu diferite ipostaze din nivelul 2:



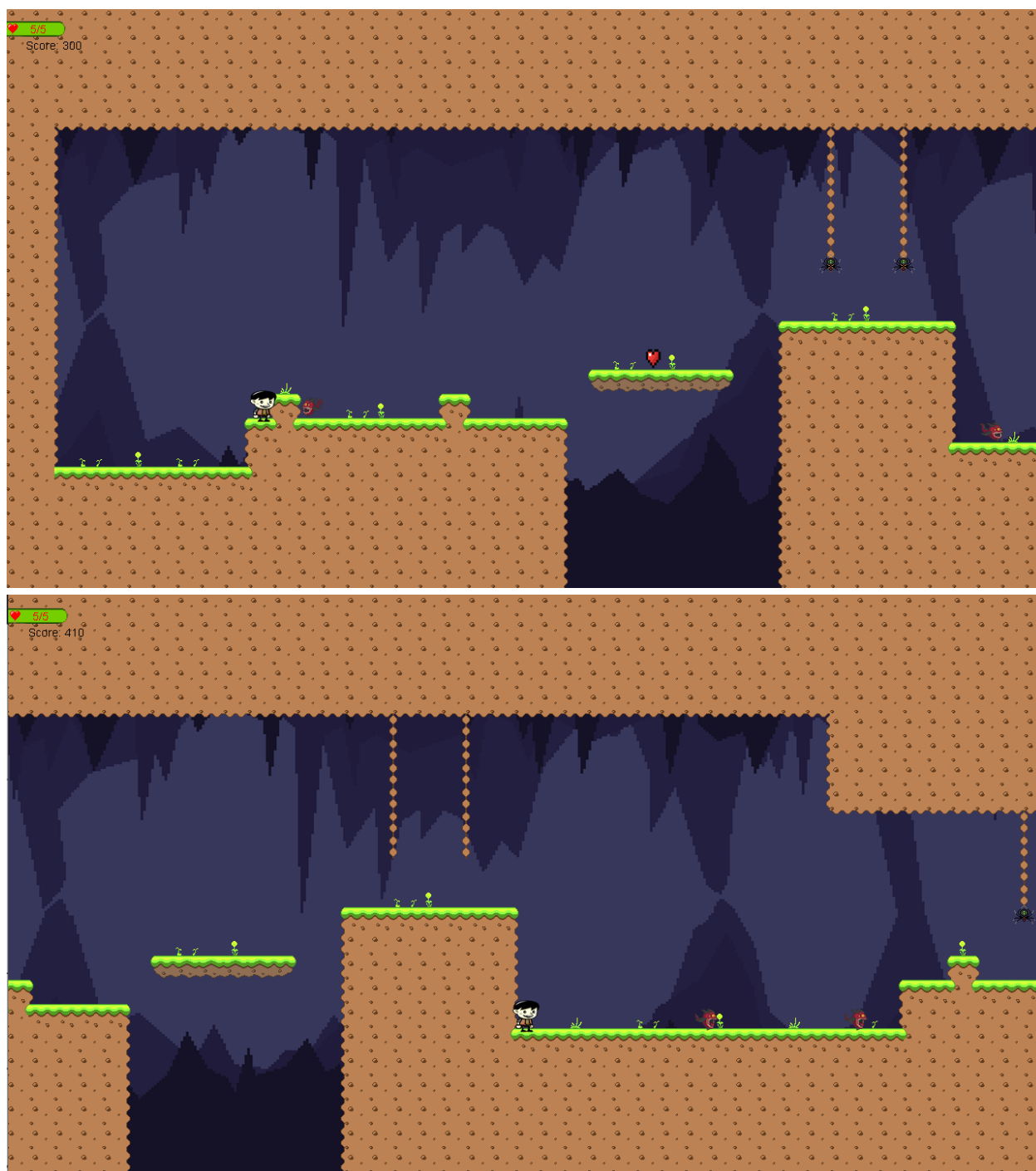
Nivelul 3:

În acest nivel, Clăușențiu se află într-o peșteră, habitatul natural al demonilor, acești demoni devin foarte agresivi când Clăușențiu se apropie de aceștia.

Toți inamicii de până acum se afla și ei în acest nivel, doar că sunt și ei mai agresivi.

Nivelul se încheie la capătul hărții.

Screenshot-uri cu diferite ipostaze ale nivelului 3:



Demonii surprinși în stadiul „enraged”

Descriere meniu:

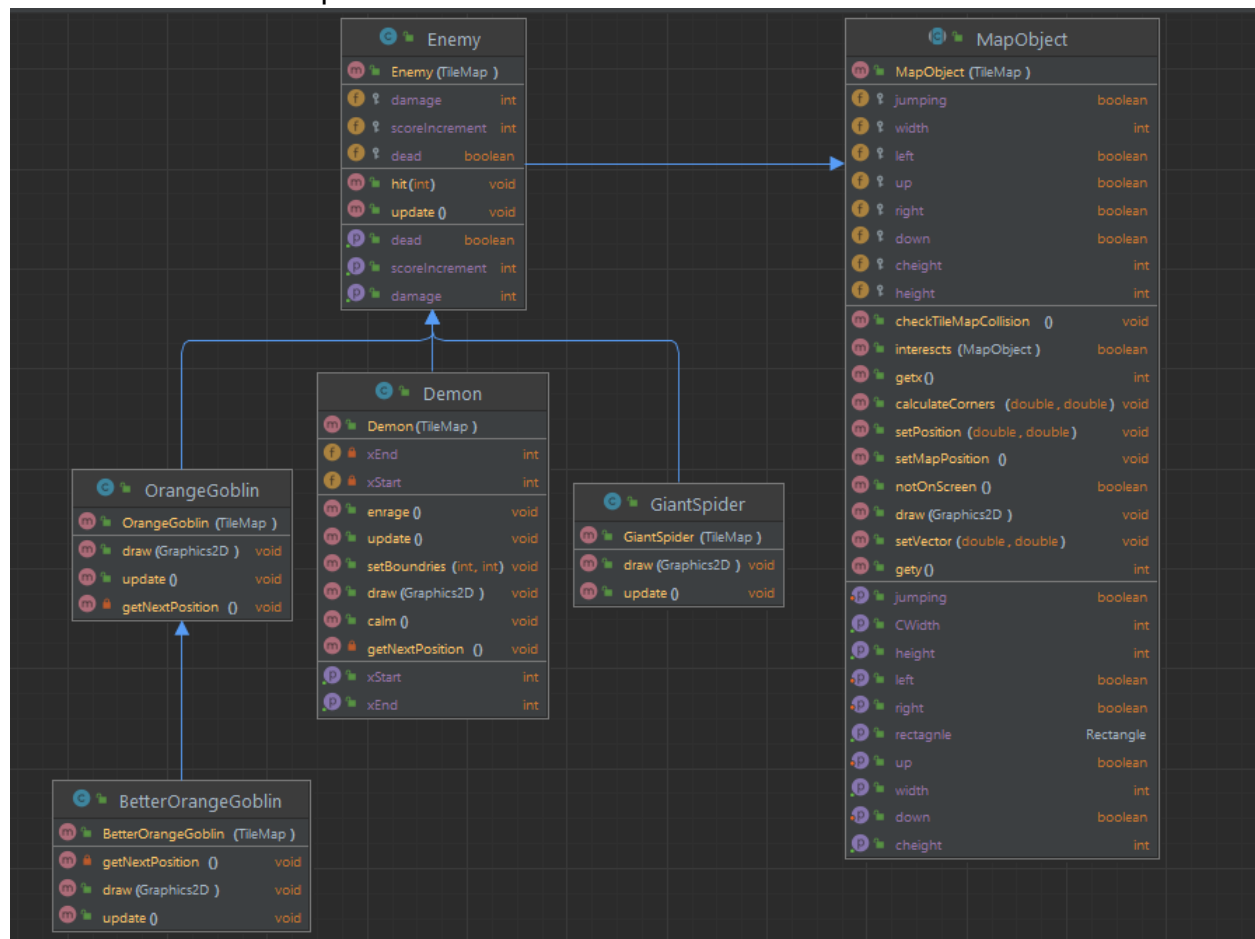
Meniul cuprinde următoarele opțiuni:

- New Game - începe jocul de la nivelul 1.
- Load Game - începe jocul de la începutul ultimului nivel din care am ieșit
- Quit - închide jocul

Diagrama de clase:

Pachetul „Enemies”:

Mențiune importantă: Clasele „Enemy” și „MapObject” nu fac parte din acest pachet însă clasele din acest pachet sunt derivate de la acestea.



Clasa „MapObject” are toate atributele de baza declarate ca și protected, atribute pe care le moștenesc atât inamicii cât și jucătorul. Tot în clasa „MapObject” se descrie metoda care se ocupă de coliziunile generale.

Pentru ca o coliziune să aibă loc, sunt calculate distanțele colțurilor la fiecare obiect în parte (metoda `calculateCorners`) ca mai apoi să se verifice coliziunea între două patrulatere, intersecția dintre patrulatere este definită o dată în metoda „`intersects`”, care verifică intersecția cu un alt obiect de tip „MapObject”, și în „`checkTileMapCollision`”, care verifică coliziunile cu harta.

Clasa „Enemy” moștenește toate atributele și metodele din „MapObject”, adaugând valoarea „`damage`”, verificarea „`dead`” (se folosește un boolean ca să verifice

moartea inamicului) și metoda „hit”, care scade viața inamicilor când sunt loviți de un atac.

Clasa „GiantSpider” are la bază toate atributele menționate anterior, aceste obiecte sunt inamici statici, au coliziune doar cu jucătorul.

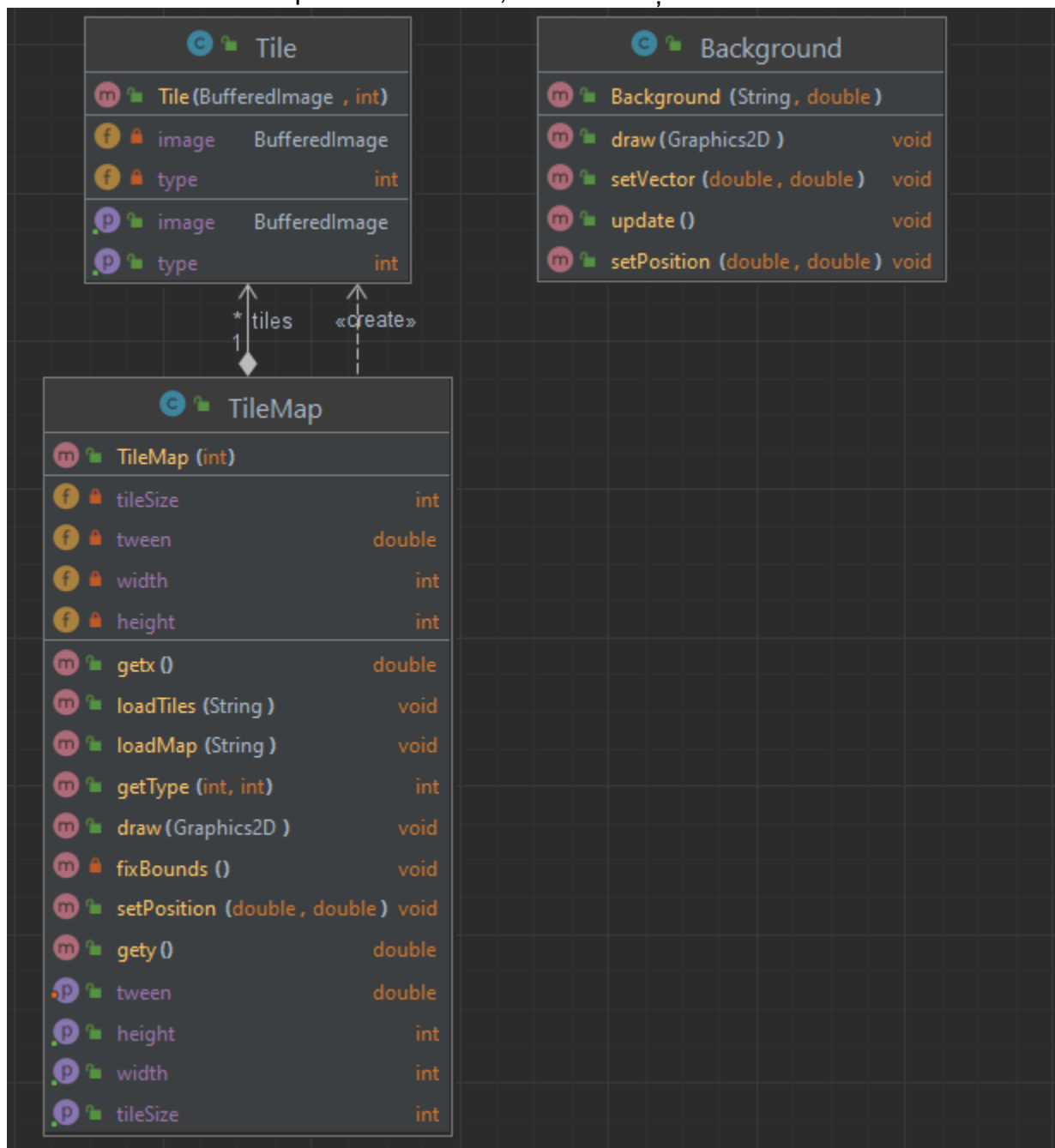
Clasele „OrangeGoblin” și „BetterOrangeGoblin” implementează o metodă de mișcare de la stanga la dreapta, care schimbă sensul de deplasare la lovirea unui perete. Diferența dintre „OrangeGoblin” și „BetterOrangeGoblin” este ca „BetterOrangeGoblin” este mai rapid și are nivelul de „damage” mult mai mare.

Clasa „Demon” este cea mai complexă clasă, pe lângă toate metodele menționate anterior, această clasă poate defini o zonă pe hartă, unde dacă intră jucătorul, intră în starea „enraged” și devine mai agresiv și rapid, iar când jucătorul părăsește zona, acesta se calmează. Demonii au cel mai mare nivel de „damage” din joc.

Pachetul „TileMap”:

În acest pachet sunt descrise metodele prin care se încarcă tile-urile, ulterior și harta jocului. Clasa „TileMap” moștenește clasa „Tile”, folosind mai departe tipul de tile-uri (Normal sau Blocked) și imaginile pe care clasa „TileMap” le folosește ca să încarce în mod vizual harta.

Clasa „Background” este folosită în toate nivelele cât și în stările jocului, deoarece acesta se ocupă cu încărcarea, desenarea și actualizarea fundalului.



Pachetul „Entity”:

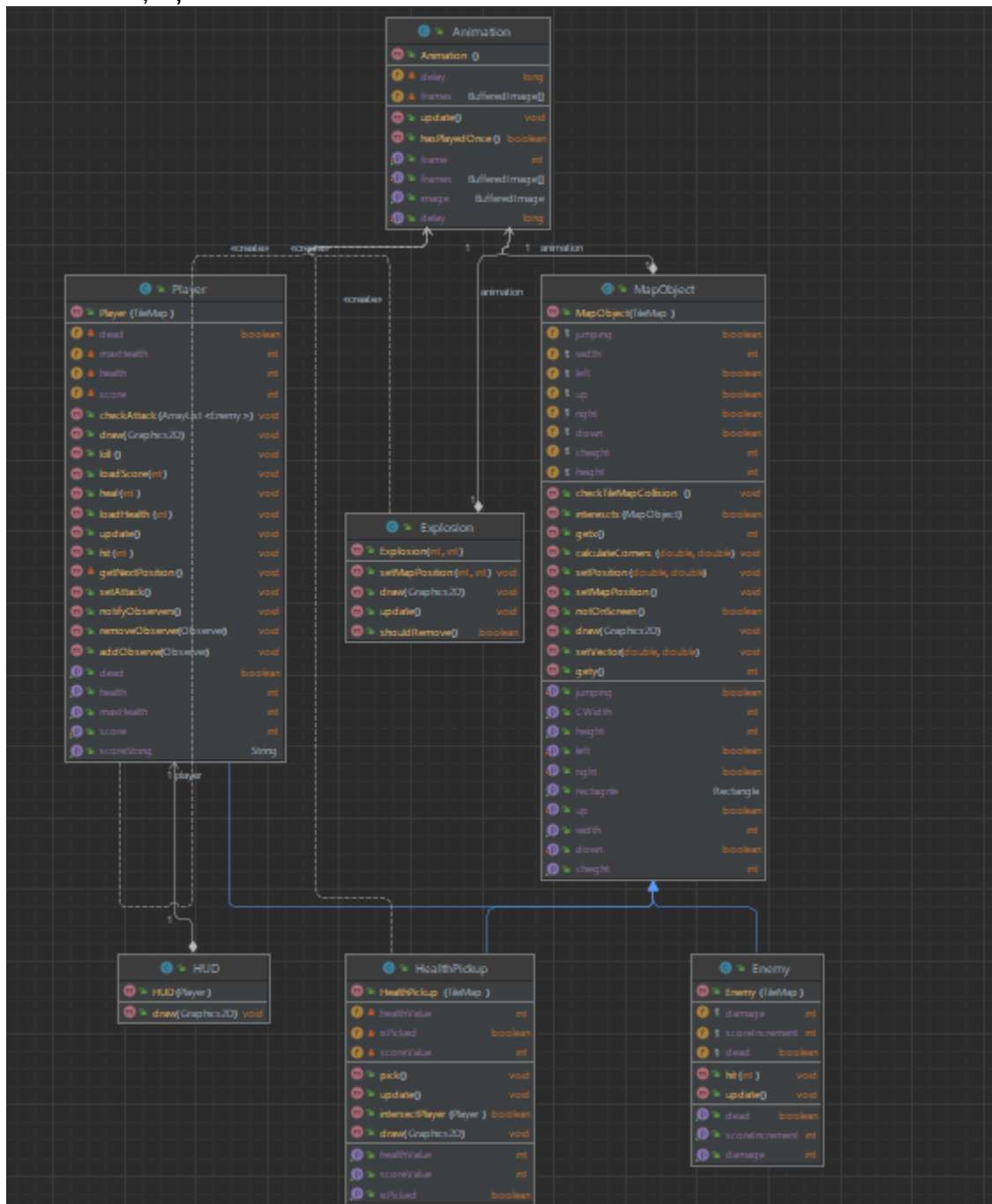
În acest pachet este definit jucătorul cu toate atributele sale (viață, score, damage, atac etc.), iar aici apare din nou clasa „MapObject”.

În acest pachet au fost definite și clasele „Animation”, „HUD”, „Explosion” și „HealthPickup”.

Clasa „Animation” se ocupă cu managerierea animațiilor formate din numere variabile de cadre pe secundă. Totodată are un semafor care verifică redarea unei animații, ceea ce este folosit, mai apoi, pentru terminarea atacului jucătorului.

Clasa „Explosion” se ocupă cu managerierea animațiilor de explozie a inamicilor, urmărind strict pozițiile lor pe hartă.

Clasa „HUD” se ocupă cu afișarea informațiilor despre jucător în timp real. Mai precis bara de viață și scorul.

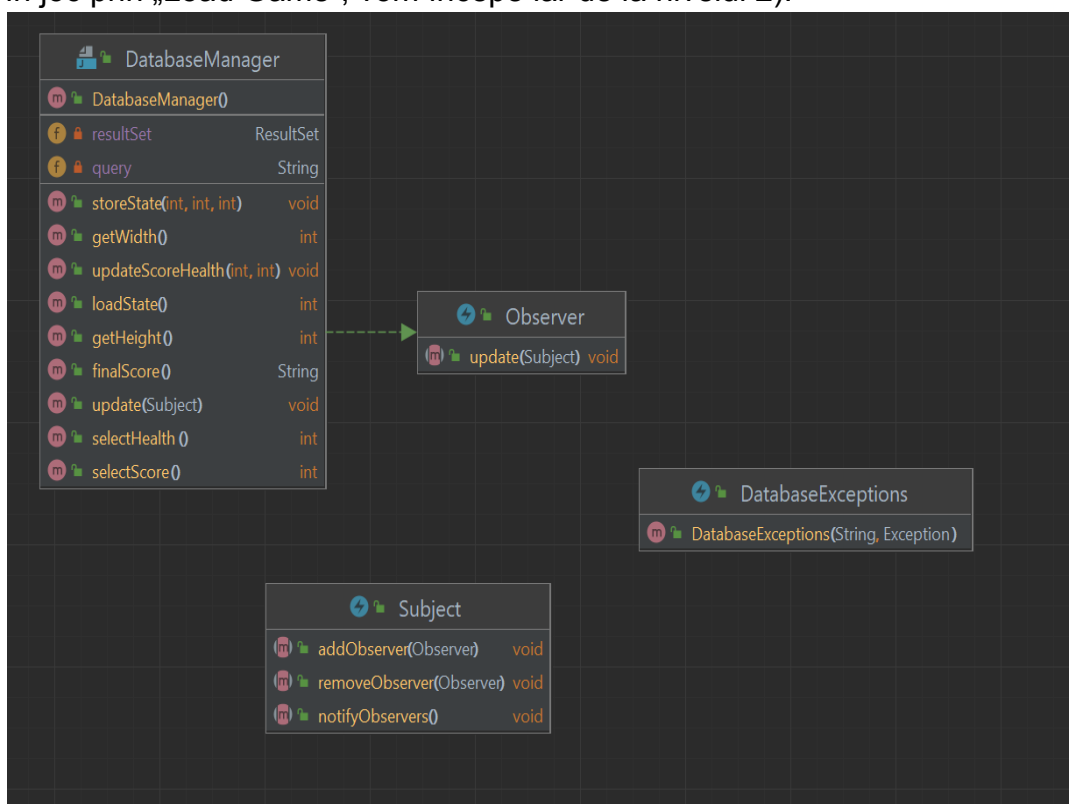


Pachetul „DatabaseManager”:

Pachetul conține clasele „DatabaseException”(Clasa de excepții), „DatabaseManager”, „Subject” și „Observer”; ultimele 2 vor fi prezentate la modelele de proiectare.

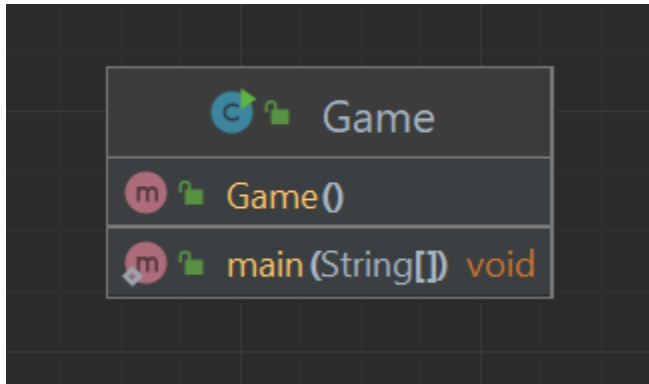
Clasa „DatabaseException” este folosită în toate clasele care folosesc o conexiune la baza de date, mai precis apare în cazul în care există posibilitatea unei excepții la conexiunea cu baza de date.

Clasa „DatabaseManager” realizează conexiunile către baza de date și operațiile care țin de baza de date. Mai precis, de aici se încarcă dimensiunile ecranului jocului. Tot din acest manager se face update cu datele jucătorului în timp real și se salvează starea în care se află jocul la momentul respectiv (adică dacă ne aflăm în nivelul 2 și dorim să părăsim jocul, apăsând tasta „ESC” se va reține nivelul 2, iar când intrăm în joc prin „Load Game”, vom începe iar de la nivelul 2).



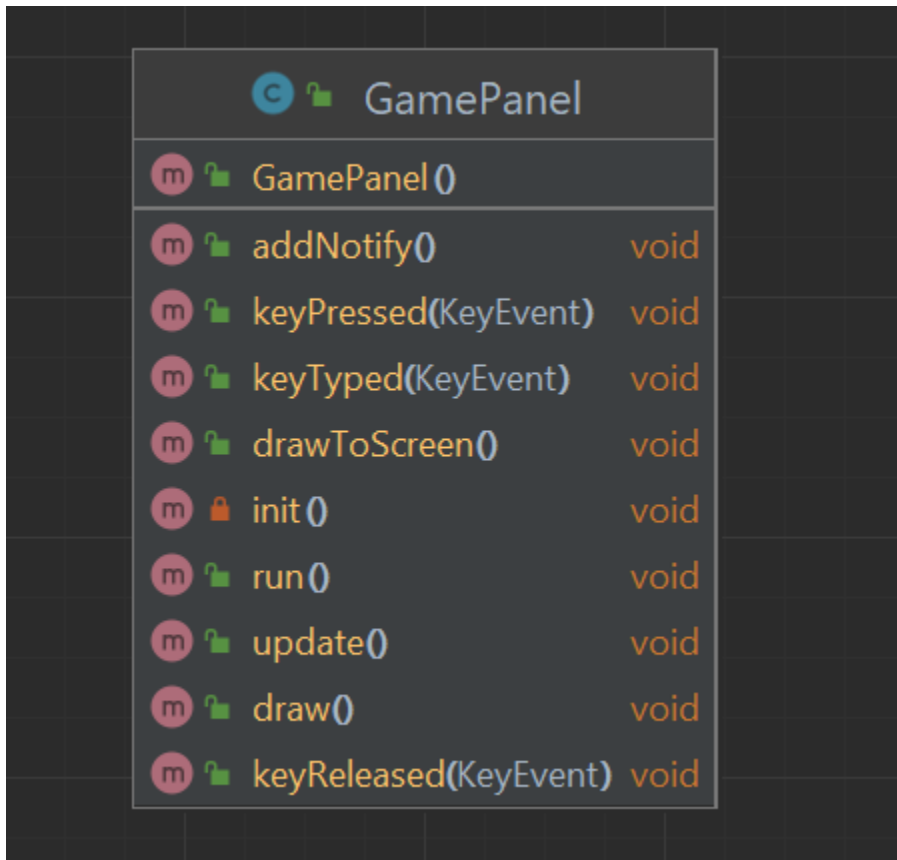
Diagramele împărțite pe clase:

-Clasa „Game”:



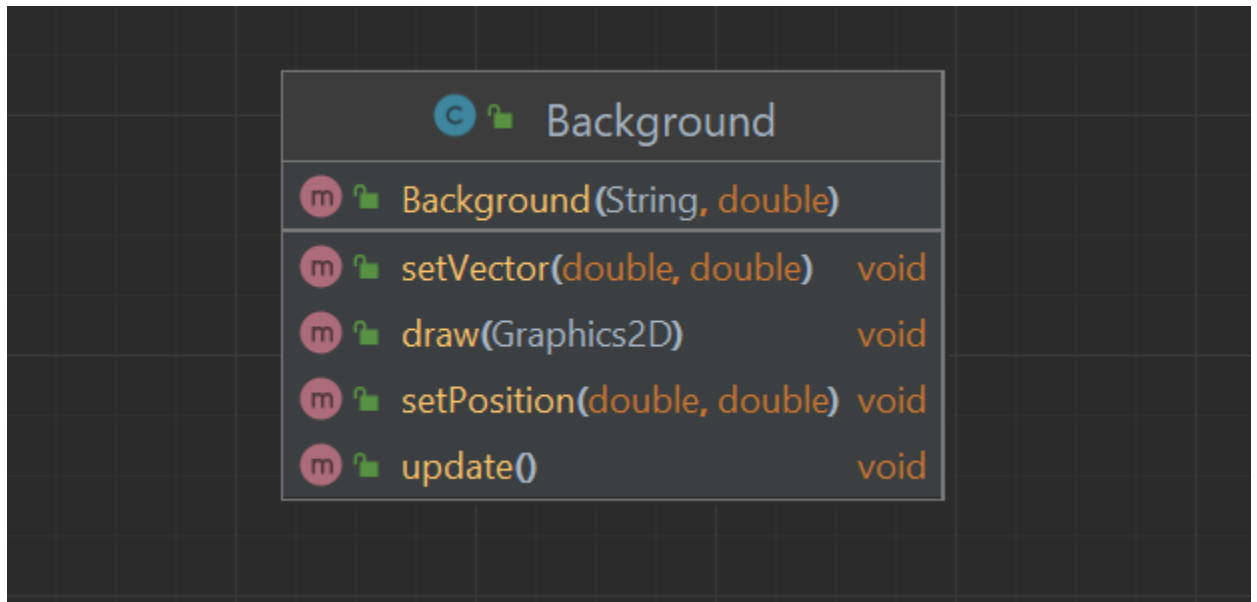
Aceasta clasa este doar „main-ul” jocului, de aici se pornește jocul.

-Clasa „GamePanel”:



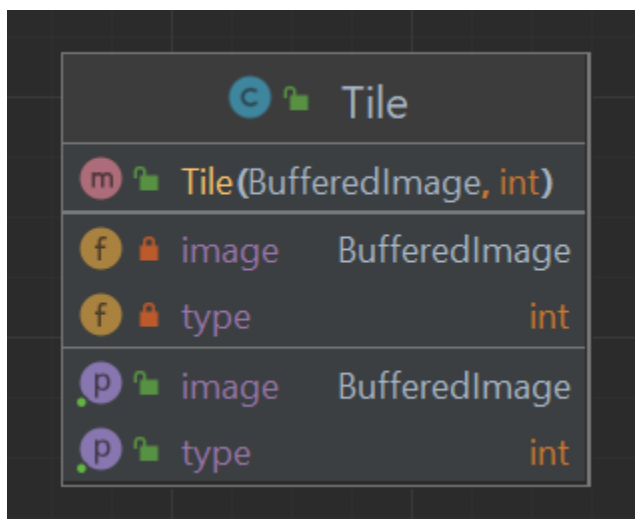
Clasa care se ocupă cu inițializarea, rularea și update-ul jocului.

-Clasa „Background”:



Clasa care încarcă fundalurile și le scalează la dimensiunile dorite, se ocupă și de reluarea fundalului în cazul în care acesta se sfârșete.

-Clasa „Tile”:



Se ocupă de atribuirea unui tip de tile-uri și totodată este o clasă abstractă.

-Clasa „TileMap”:

TileMap		
m	TileMap (int)	
f	height	int
f	tween	double
f	tileSize	int
f	width	int
m	getType (int, int)	int
m	draw(Graphics2D)	void
m	fixBounds ()	void
m	loadTiles (String)	void
m	getX ()	double
m	setPosition (double, double)	void
m	loadMap (String)	void
m	gety ()	double
p	width	int
p	height	int
p	tween	double
p	tileSize	int

Aici se seteaza, încarcă și desenează harta cu toate tile-urile și tipul lor

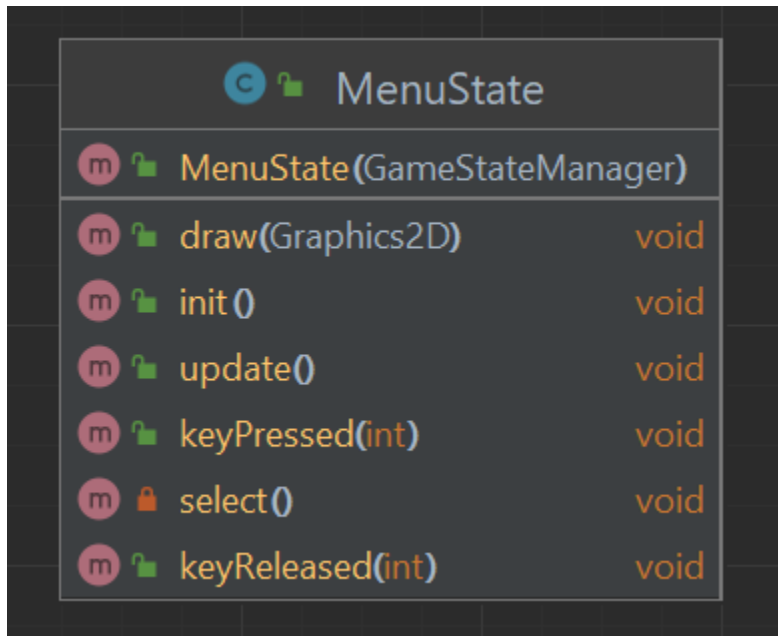
-Clasele „WinState” si „GameOverState”:

GameOverState		
m	GameOverState(GameStateManager)	
m	update()	void
m	keyPressed (int)	void
m	init()	void
m	keyReleased(int)	void
m	draw(Graphics2D)	void

WinState		
m	WinState(GameStateManager)	
m	draw(Graphics2D)	void
m	keyReleased(int)	void
m	update()	void
m	keyPressed (int)	void
m	init()	void

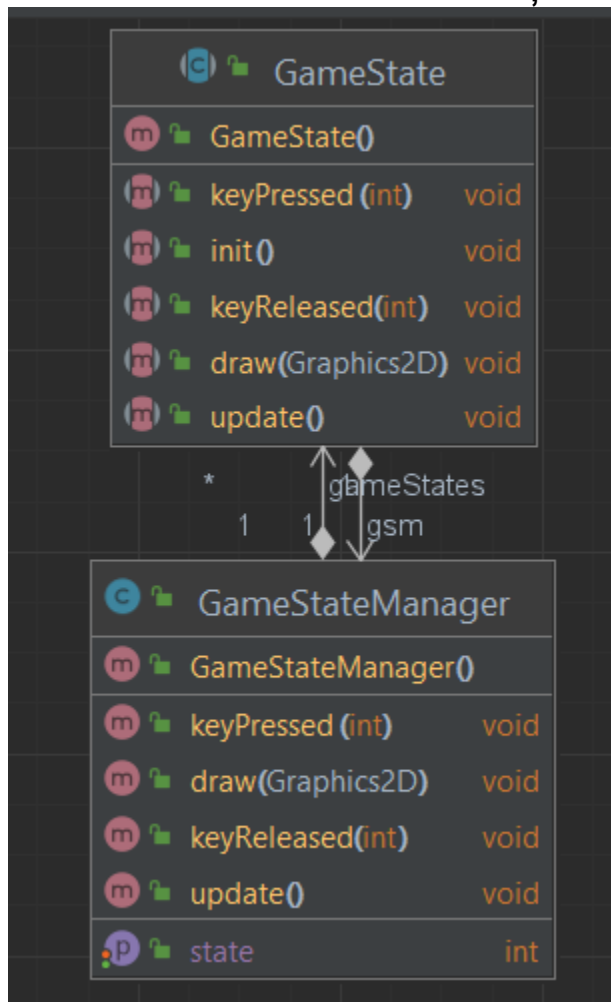
Aceste două clase sunt foarte similare deoarece afișează un mesaj care informează jucătorul de stadiul lui curent. Singura diferență este că în clasa „WinState”, se face conexiune la baza de date și se afișează scorul total.

-Clasa „MenuState”:



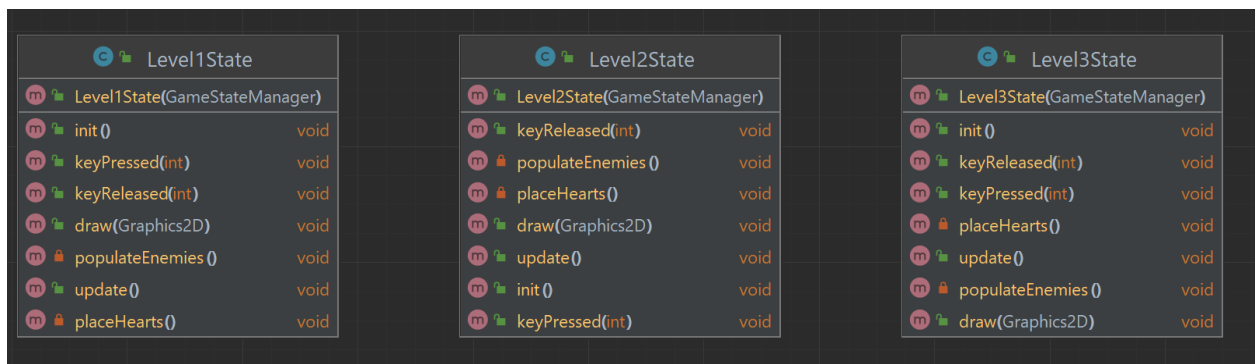
Similară cu cele două clase menționate anterior, această clasă doar afișează stadiul în care se află jucătorul. Ceea ce diferențiază această clasă de celelalte, este interactivitatea. Meniul colorează în timp real opțiunea pe care ne aflăm, iar selecția unei opțiuni trimite către alt state.

-Clasele „GameState” și „GameStateManager”:



Cele două clase dependente una de cealaltă, se ocupă cu managerierea state-urilor. Clasa „GameState” este clasă de bază pentru celelalte state-uri, iar acele state-uri sunt manageriate în clasa „GameStateManager”.

-Clasele „Level1State”, „Level2State” și „Level3State”:



Aceste 3 clase implementează metode similare una cu cealaltă, diferențele

principale sunt: față de nivelul 1, nivelul 2 trebuie să inițializeze din baza de date scorul și viața pentru nivelul 2, iar nivelul 3 trebuie să stocheze scorul final.

Alte diferențe sunt numărul instanțelor de tip „Enemy”, de la un nivel la altul sunt instanțiați tot mai mulți inamici.

-Clasa „MapObject”:

MapObject		
m	MapObject(TileMap)	
f	left	boolean
f	up	boolean
f	width	int
f	down	boolean
f	cheight	int
f	right	boolean
f	height	int
f	jumping	boolean
m	checkTileMapCollision ()	void
m	setMapPosition ()	void
m	draw(Graphics2D)	void
m	getx()	int
m	notOnScreen ()	boolean
m	setVector(double, double)	void
m	setPosition(double, double)	void
m	gety()	int
m	calculateCorners (double, double)	void
m	interescts (MapObject)	boolean
p	down	boolean
p	left	boolean
p	jumping	boolean
p	width	int
p	height	int
p	cheight	int
p	rectagnle	Rectangle
p	right	boolean
p	up	boolean
p	CWidth	int

Această clasă are implementată toate atributele care se regăsesc la entitățile din joc, tot aici sunt implementate coliziunile, deplasările și alte mecanici pe care le moștenesc toate entitățile.

Coliziunile sunt realizate prin verificarea unei intersecții dintre două patrulatere, aceste 2 patrulatere pot fi entități sau teren. La coliziunea cu terenul se verifică tipul de teren (blocat sau normal) ceea ce indică dacă e teren solid sau nu.

Coliziunile cu inamicii rezultă daune asupra barei de viață a jucătorului. Se aplică același principiu.

-Clasa „Player”:

Player		
Player(TileMap)		
f	health	int
f	maxHealth	int
f	dead	boolean
f	score	int
m	addObserver(Observer)	void
m	checkAttack (ArrayList <Enemy>)	void
m	notifyObservers()	void
m	hit(int)	void
m	heal(int)	void
m	loadHealth(int)	void
m	draw(Graphics2D)	void
m	setAttack()	void
m	loadScore(int)	void
m	update()	void
m	removeObserver(Observer)	void
m	kill()	void
m	getNextPosition()	void
p	health	int
p	dead	boolean

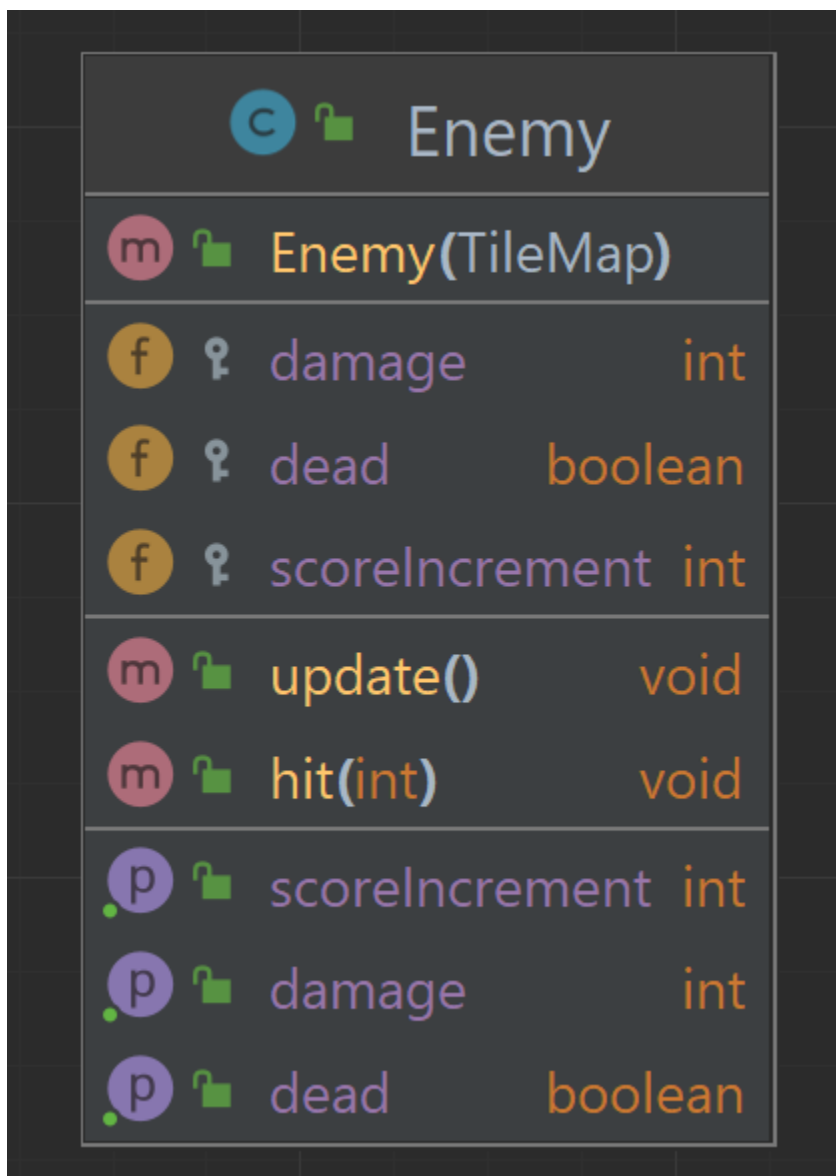
Această clasă moștenește clasa „MapObject”, pe lângă attributele anterioare, se aduc ca noutate metodele de încarcat viață și scor, notificarea observatorului de schimbările de score și viață, metoda de vindecare și cea de atac.

Coliziunea cu inamicii este cea standard, ce adaugă în schimb această clasă este metoda „checkAttack” care verifică fiecare inamic dacă atacă sau este atacat.

Se implementează și metodele „hit” și „heal” care sunt opuse una celeilalte.

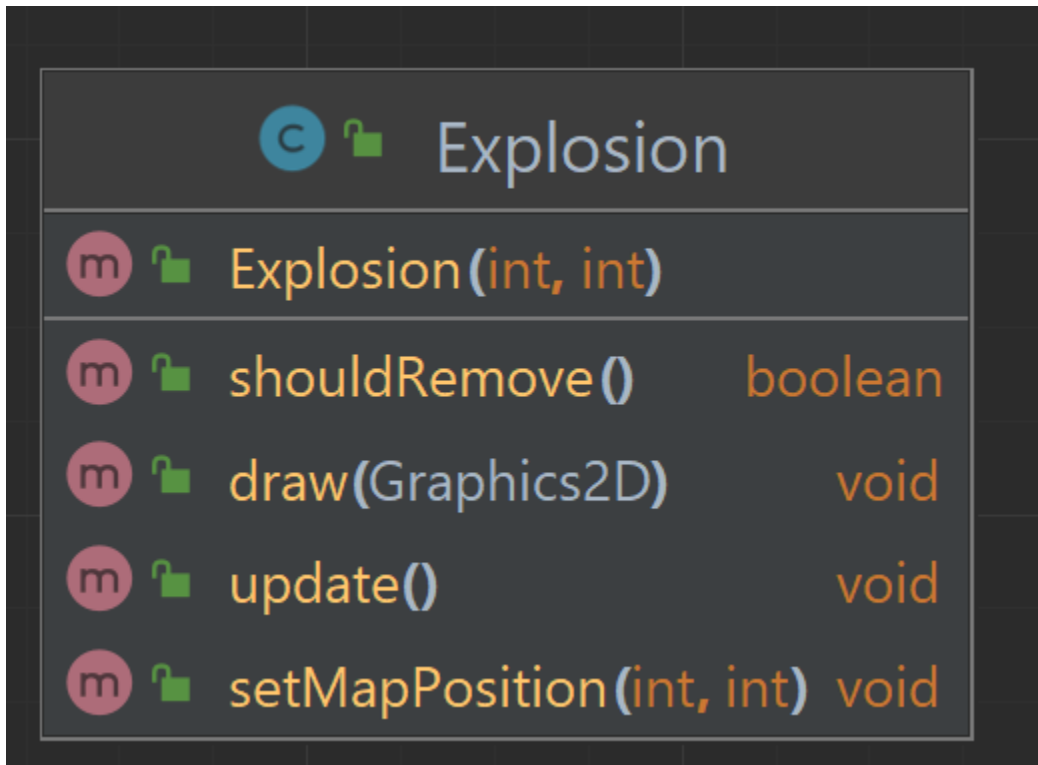
Metoda „kill” setează campul „dead” pe valoarea true ca jucătorul să fie eliminat când cade în abis.

-Clasa „Enemy”:



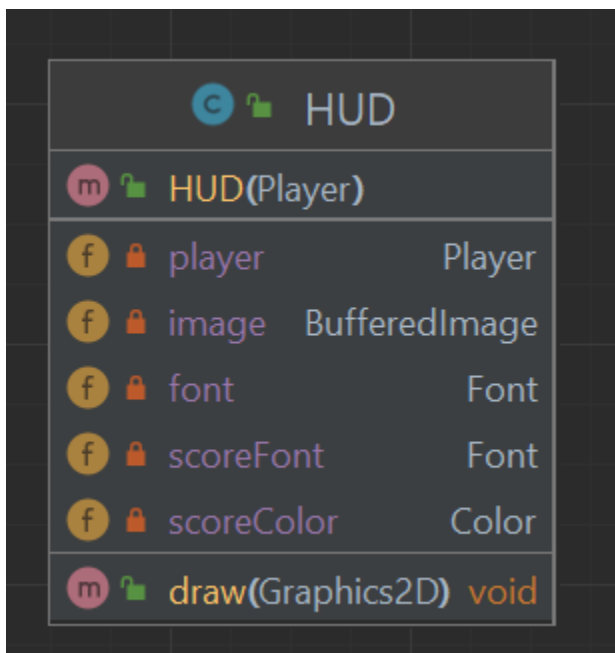
Această clasă este clasa de bază pentru toate tipurile de inamici care se regăsesc în joc.

-Clasa „Explosion”:



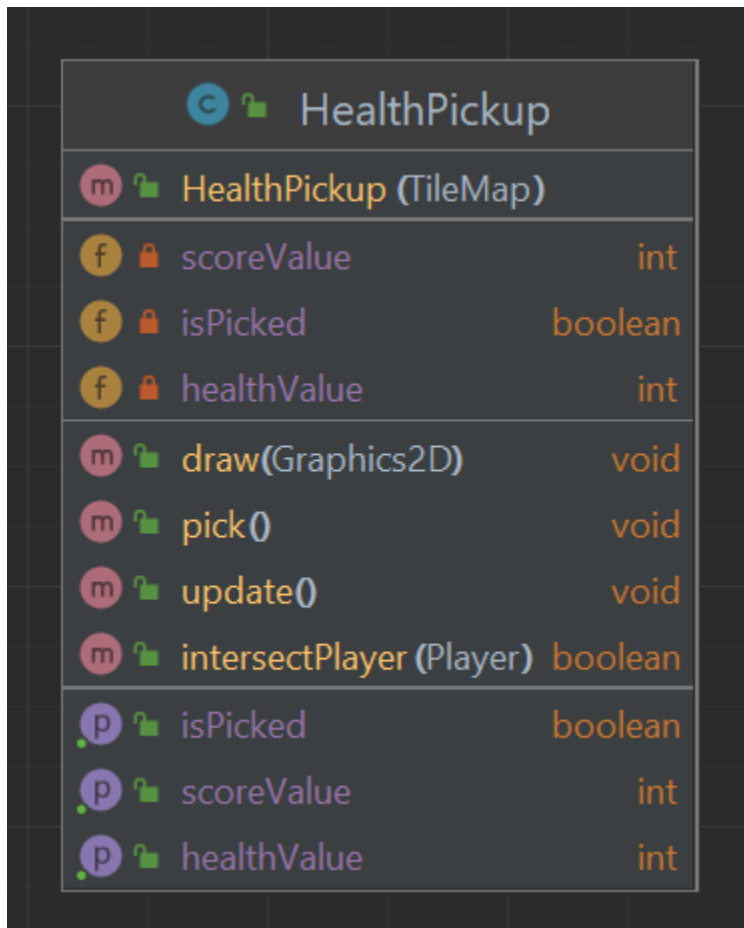
Această clasă se ocupă de managerierea exploziilor care apar în momentul în care un inamic moare. Practic se ocupă cu încărcarea animației, desenarea exploziei pe ecran când inamicul moare și ștergerea exploziei după ce se termină animația.

-Clasa „HUD”:



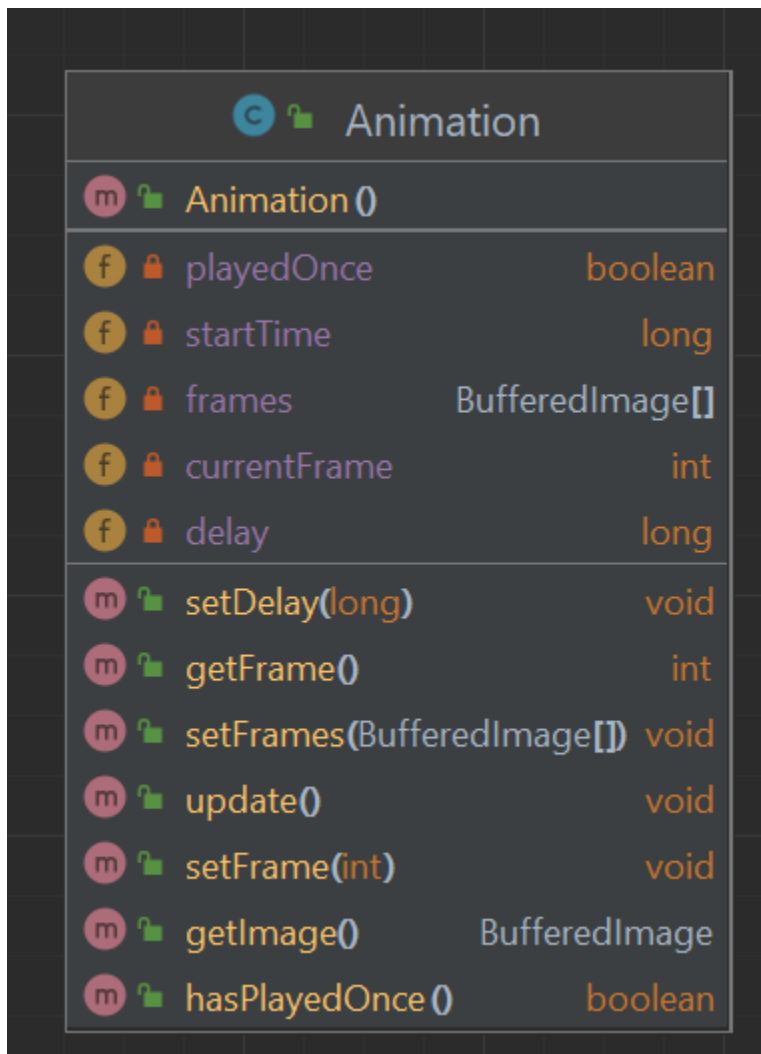
Această clasă se ocupă de actualizarea HUD-ului jucătorului, preia câtă viață mai are jucătorul și o afișează în timp real. Totodată afișează și scorul jucătorului.

-Clasa „HealthPickup”:



Această clasă face parte din categoria entități deoarece pentru a avea interacțiunea dorită cu jucătorul trebuie să aibă coliziune. Această clasă moștenește clasa „MapObject” și încarcă imaginea de inimioară. Când jucătorul intră în coliziune cu acest obiect, 2 acțiuni se pot petrece: fie crește viața cu 2 puncte, fără a depăși maximul sau dacă maximul este atins deja, va crește scorul cu 20 de puncte.

-Clasa „Animation”:

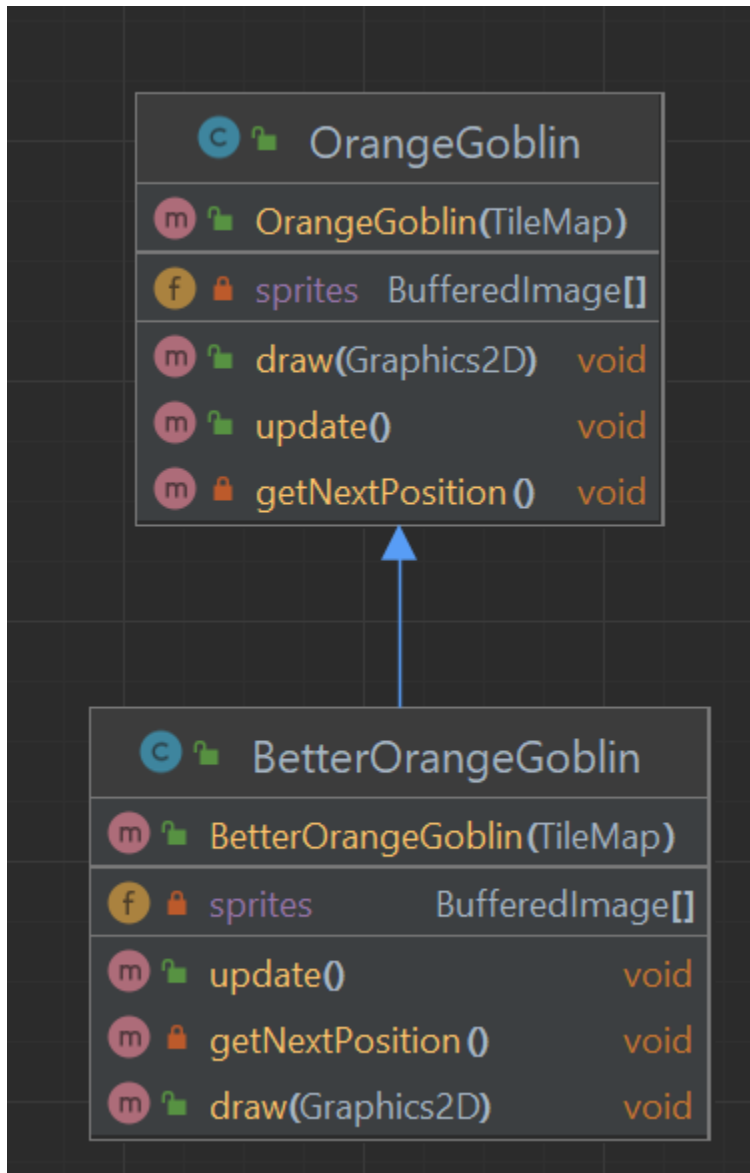


The image shows a screenshot of an IDE window displaying the class definition for 'Animation'. The class is shown with its name and a small icon. Below the class name, the methods and fields are listed. Each item is preceded by a circular icon: a pink circle with 'm' for methods and a yellow circle with 'f' for fields. Each item also has a small lock icon to its right. The methods and fields are listed in two columns, with the return type or data type on the right.

Animation	
m	Animation()
f	playedOnce boolean
f	startTime long
f	frames BufferedImage[]
f	currentFrame int
f	delay long
m	setDelay(long) void
m	getFrame() int
m	setFrames(BufferedImage[]) void
m	update() void
m	setFrame(int) void
m	getImage() BufferedImage
m	hasPlayedOnce() boolean

Clasa aceasta se ocupă cu managerierea animațiilor, mai precis sunt încărcăți vectori de imagini, care apoi sunt puși pe categorii. Iar prin metoda „setFrames” se setează ce animație va fi redată mai departe.

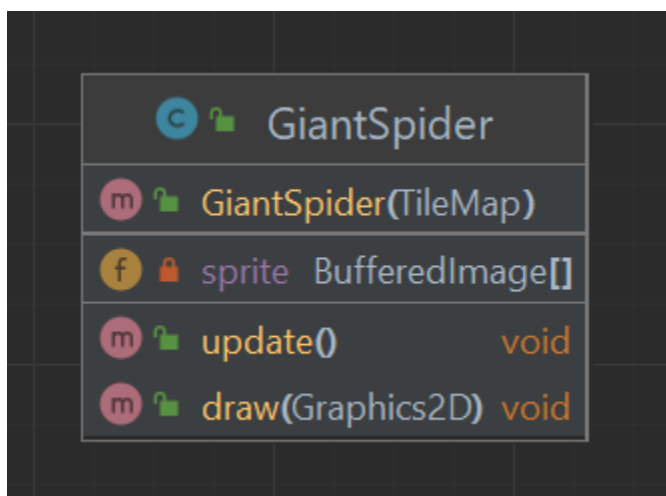
-Clasele „OrangeGoblin” și „BetterOrangeGoblin”:



Aceste două clase sunt similare între ele, „OrangeGoblin” este derivată din clasa „Enemy”, are implementată metoda de mișcare în `update`, funcția „`getNextPosition`” verifică unde se va afla următoarea sa poziție pentru a evita bug-urile vizuale care s-ar putea întâmpla când inamicul lovește un perete. La întâlnirea unui perete, goblinii își schimbă sensul de mers.

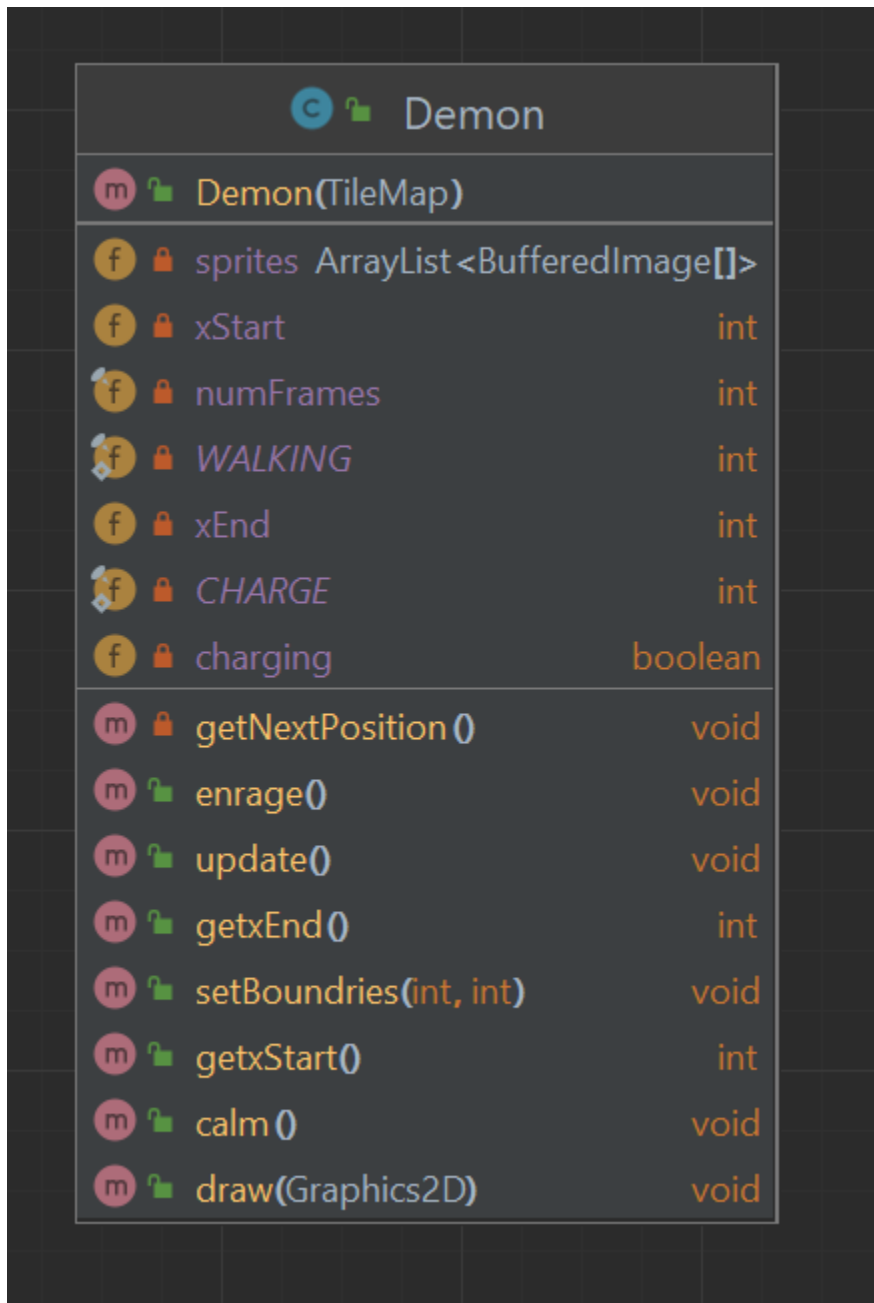
Față de „OrangeGoblin”, „BetterOrangeGoblin” se mișcă mult mai repede și mai agresiv, valorează mai multe puncte, dar și scade viață mult mai multă.

-Clasa „GiantSpider”:



Fiind un inamic care nu se mișcă are doar metodele moștenite.

-Clasa „Demon”:

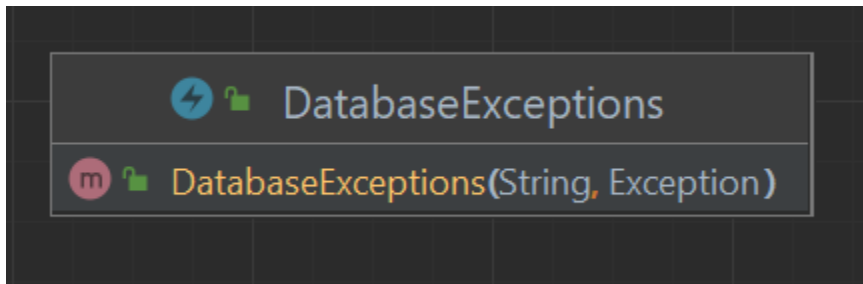


The screenshot shows the class structure of the 'Demon' class in a Java IDE. The class is named 'Demon' and has a constructor 'Demon(TileMap)'. It contains several fields and methods. The fields are: 'sprites' (ArrayList<BufferedImage[]>), 'xStart' (int), 'numFrames' (int), 'WALKING' (int), 'xEnd' (int), 'CHARGE' (int), and 'charging' (boolean). The methods are: 'getNextPosition()' (void), 'enrage()' (void), 'update()' (void), 'getxEnd()' (int), 'setBoundries(int, int)' (void), 'getxStart()' (int), 'calm()' (void), and 'draw(Graphics2D)' (void).

Icon	Field/Method	Type
m	Demon(TileMap)	
f	sprites	ArrayList<BufferedImage[]>
f	xStart	int
f	numFrames	int
f	WALKING	int
f	xEnd	int
f	CHARGE	int
f	charging	boolean
m	getNextPosition()	void
m	enrage()	void
m	update()	void
m	getxEnd()	int
m	setBoundries(int, int)	void
m	getxStart()	int
m	calm()	void
m	draw(Graphics2D)	void

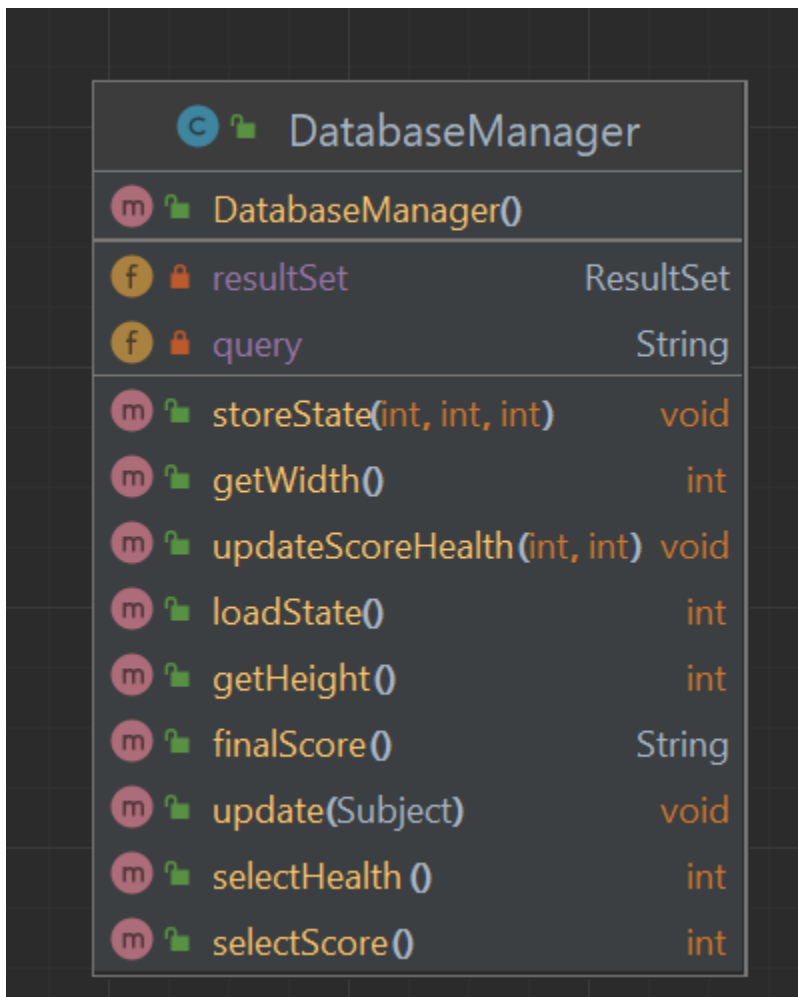
Cea mai complexă clasă de tip inamic, pe lângă metodele similare cu cele de la „OrangeGoblin”, aceasta clasă definește și o zonă mică pe hartă unde atunci când se apropie un jucător, crește agresivitatea demonilor, dar se schimbă și animația odată cu aceasta. Evident, când jucătorul părăsește această zonă, demonii se calmează.

-Clasa „DatabaseException”:



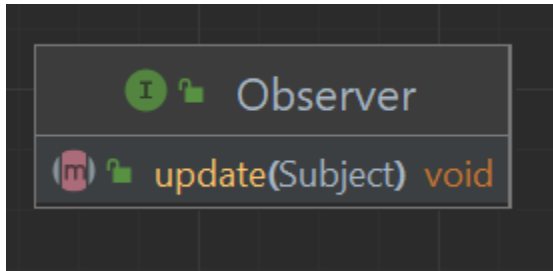
Clasa de excepție pentru a fi apelată de fiecare dată când apare o excepție la baza de date.

-Clasa „DatabaseManager”:



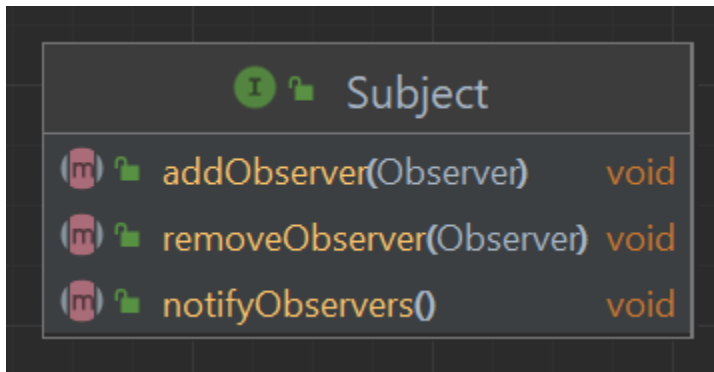
Aici sunt manageriate toate operațiile cu baza de date, campurile `query` și `resultSet` sunt folosite la toate citirile din baza de date, pe când la scrieri este folosit doar câmpul `query`. Totuși, deoarece scrierile în baza de date sunt lente, fiecare scriere se întâmplă asincron într-un thread separat de cel principal.

-Clasa „Observer”:



Se declară o singură metodă numită update care se regăsește în „DatabaseManager”

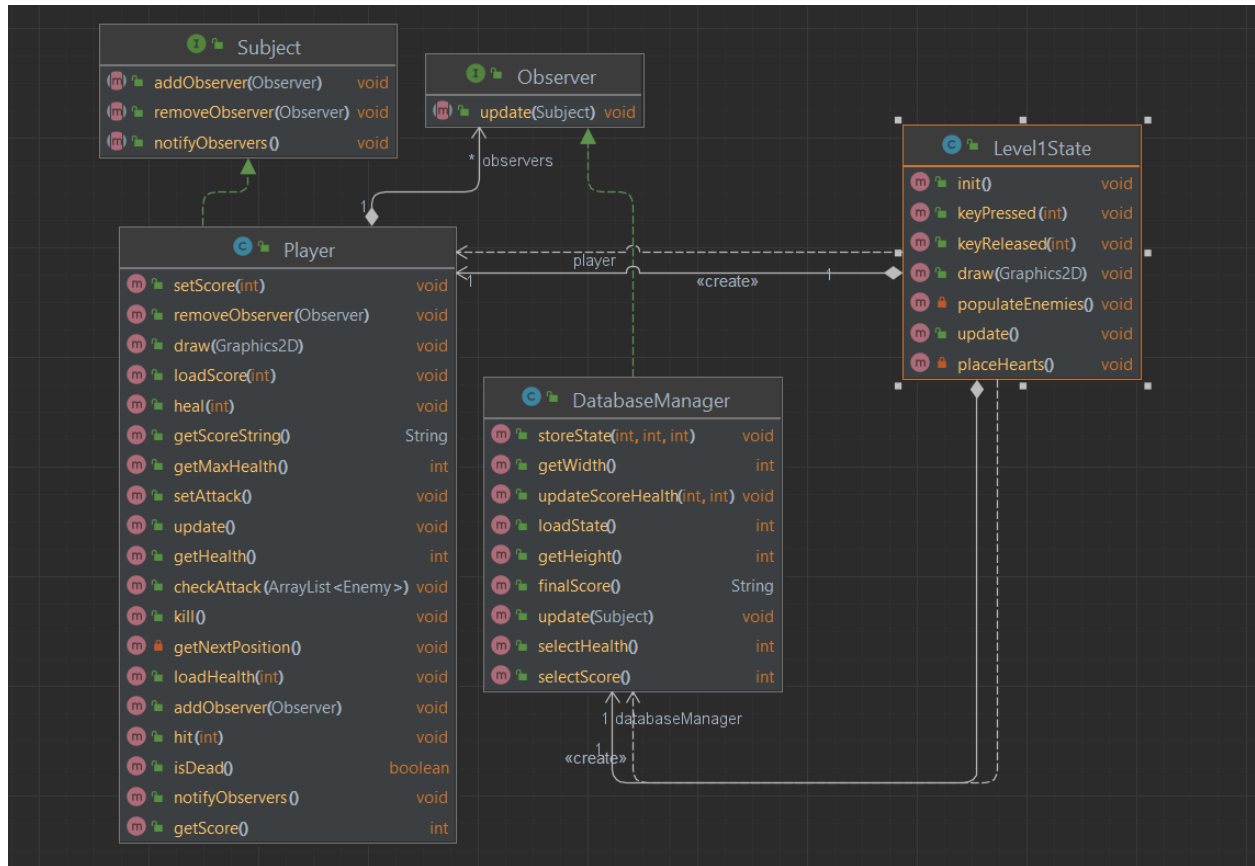
-Clasa „Subject”:



Definește toate metodele pe care pot fi aplicate pentru un observator, adauga, șterge și trimite o notificare la observatorii activi.

Modelele de proiectare utilizate:

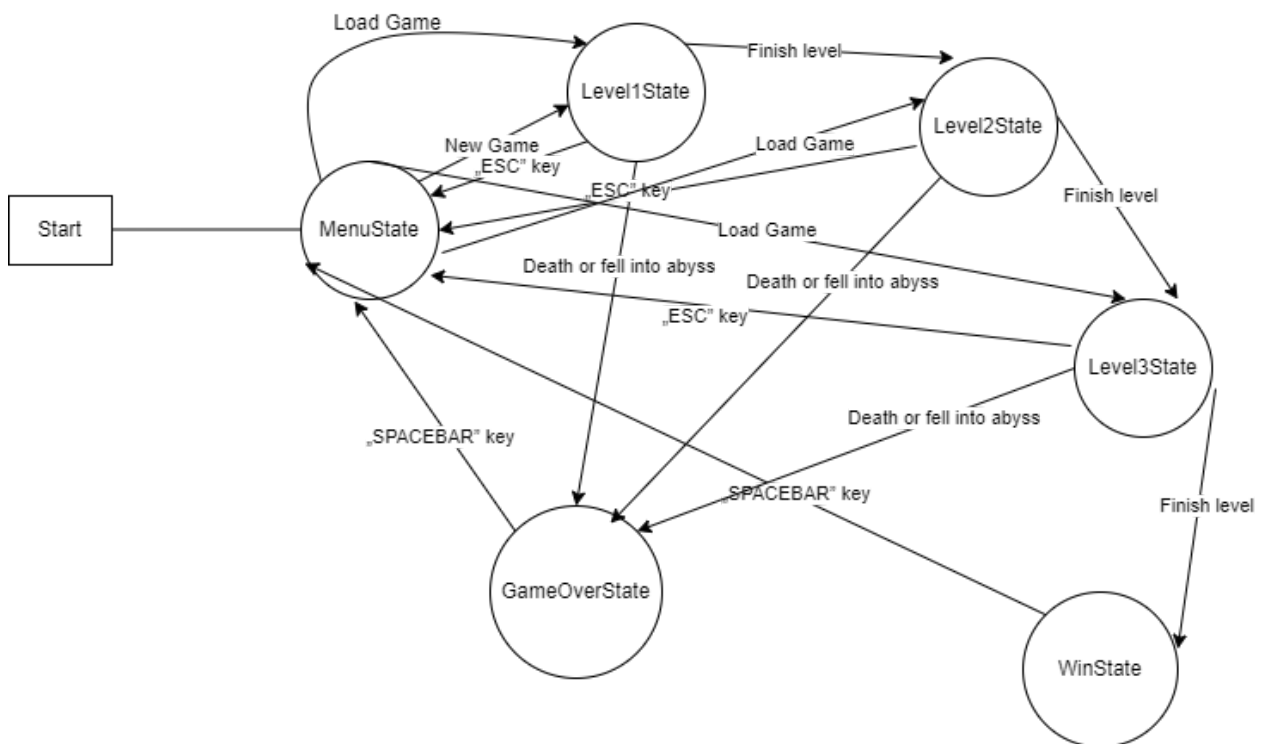
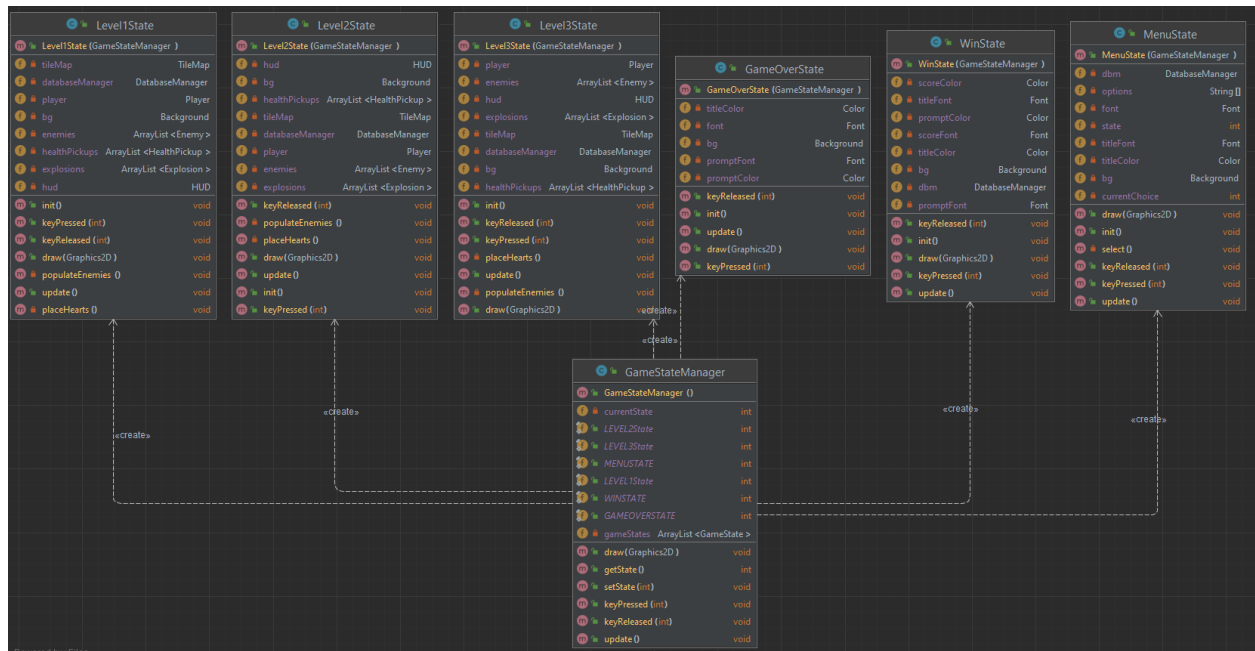
1) Observer pattern:



Am definit clasele „Subject” și „Observer” pentru a avea metodele intermediare. Mai precis add, remove și notify, respectiv update.

Metodele add, remove și notify sunt definite în clasa „Player”, iar metoda update este definită în clasa „DatabaseManager”, obiectele de tip „Player” și „DatabaseManager” sunt instanțiate în toate nivelele (în cazul curent, am adus doar diagrama nivelului 1 pentru referință), fiind adăugat ca observator un obiect de tip „DatabaseManager” prin metoda „addObserver”. Iar în fiecare nivel, la funcția de update, când jucătorul a o interacțiune care îi crește scorul sau scade/crește nivelul de viață, subiectul (Jucătorul) va notifica observatorul (adica „DatabaseManager”) de aceste schimbări, urmând mai apoi să facă actualizările necesare în mod asincron pentru a ușura operațiunile.

2) State pattern:



Descrierea amănunțită a acestui design pattern: Totul începe în MenuState, de acolo în funcție de opțiune selectată se poate accesa alt state al jocului.

Alegând „New Game” utilizatorul poate să înceapă jocul de la nivelul 1 (același lucru se poate întâmpla și dacă iese forțat și apasă „Load Game”). Câștigând nivelul 1, jocul trece la starea nivelului 2, unde deja lucrurile încep să se salveze. Apăsând tasta „ESC”, jucătorul este trimis direct la meniul principal, de unde poate da „Load Game”

pornind iarăși de la nivelul 2 al jocului.

Aceleași principii se aplică și pentru nivelul 3, însă, finalizarea nivelului 3 duce la câștigarea jocului, adică se ajunge în WinState, de acolo doar prin apăsarea tastei „SPACE” se ajunge înapoi la meniul principal.

Orice moarte pe parcursul celor 3 nivele trimite utilizatorul în „GameOverState”, un state care anunță utilizatorul că jocul s-a încheiat și aceștia au pierdut, fiind nevoiți să o ia de la capăt

Bibliografie:

Head First Java, 2nd Edition by Kathy Sierra and Bert Bates

Dive Into Design Patterns by Alexander Shvets

Thinking in Java by Bruce Eckel

[2d Game Asset Character](#)

[Opengameart.org](#)

<https://github.com/foreignguymike/legacyYTtutorials>

Cursuri + Laboratoare