Universitatea Tehnică ”Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Domeniul Calculatoare și Tehnologia Informației
Specializarea Calculatoare Tehnologia Informației

FIREBOY & WATERGIRL

Andrei Găina

1207A

Gameplay (rules): Pentru început, bine ai venit în lumea celor doi buni prieteni Fireboy și Watergirl. Initial Watergirl este pierduta într-o lume total necunoscuta. Pe parcurs aceasta își da seama ca pentru a-si regăsi bunul prieten Fireboy, trebuie sa izbutească în a rezolva mai multe provocari. Aceasta trebuie sa parcurga diferite trasee colectand diamantele albastre pentru a putea afla în ce loc este ascunsă cheia de la ușa care o duce cu un pas mai aproape de Fireboy. Însă aceasta trebuie să fie atenta deoarece drumul are obstacole iar dacă este neatenta și calca în lava se va evapora pierzand din cele 3 vieți care i-au fost oferite la început de drum. Neatentia costa iar dacă va pierde toate cele trei vieți v-a trebui sa o ia de la capăt.

Plot (gamestory): Jocul se desfășoară în doua lumi diferite. Prima lume este o lume cuprinsă de temperatuti negative, de un frig îngrozitor. Watergirl se afla într-o peștera de gheață. Aceasta trebuie să se grabeasca sa iasă din acea peștera deoarece exista riscul sa înghețe și ea. În continuare nimic nu este mai ușor ci din contra, provocarile sunt tot mai dificile. Aceasta nimereste la polul opus, într-o peștera cuprinsă de un foc aprig și de mult mai multe obstacole, unele chiar imprevizibile. Aici exista riscul ca Watergirl să se evapore iar pe măsura ce colecteaza diamantele aceasta pestere este inundată de lava.

Characters:

1. Fireboy este personajul secundar la început. Acesta este pierdut în neant așteptând să fie salvat de prietana lui, Watergirl.

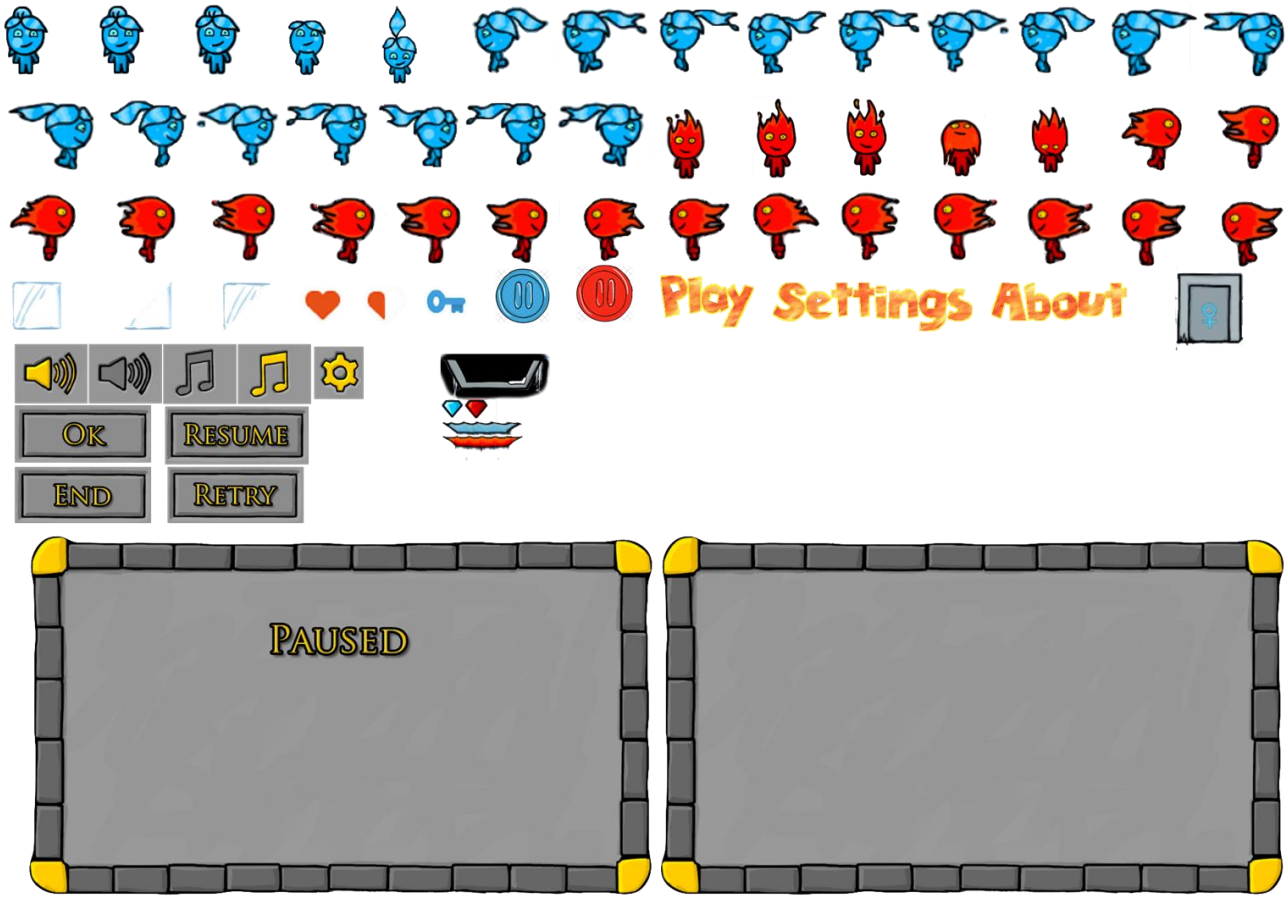
2. Watergirl este personajul principal. Aceasta face toată munca pentru a-si regăsi cel mai bun prieten. Deși aceasta este o fata, cu atenție reușește sa treacă peste provocari fizice dificile pentru cei mai mulți dintre băieți.

Mechanics:

1. **Taste:**
   - pentru deplasare-> *săgeți sus, dreapta,stânga;*
   - pentru colectare -> *tastaspace.*
2. **Game points:***In funcție de lumea în care se afla, Watergirl va fi recompensata pentru diamantele adunate dar și pentru găsirea cheii ce o ajuta sa deschidă ușa care ascunde o noua lume plină de mister.*
3. **User Interaction:** Watergirl *trebuie* să evite craterele pline cu lava pentru a nu se evapora astfel pierzand din șanse pentru a trece nivelul cât mai repede. Aceasta trebuie sa meargă în dreptul diamantelor pentru a le colecta. Cu ajutorul tastei space aceasta va lua diamantele. Numărul diamantelor colectate este afisat în partea de sus a jocului. Odată ce colecteaza toate diamantele, cheia va apărea într-un loc oarecare. După ce colecteaza și cheia , va putea trece mai departe, într-o noua lume.

Win/Lose: Watergirl va catiga doar atunci când va reuși să îndeplinească cu bine toate provocarile și îl va găsi pe bunul ei prieten Fireboy. Daca provocarile sunt prea dificile și aceasta pierde toate cele trei vieți v-a trebui sa o ia de la capăt din păcate.

Spritesheets:



- **Watergirl**
- **Fireboy**
- **Items, butoane, dale, ferestre ajutatoare**

Background images:

MENU IMAGE

LEVEL I IMAGE

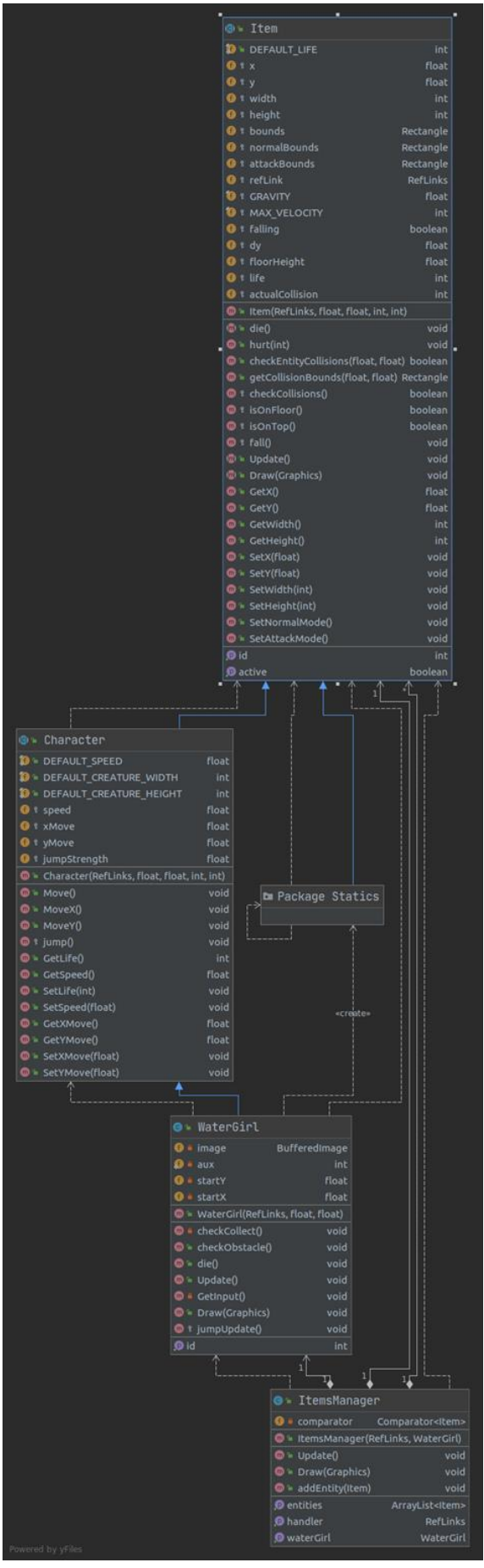
LEVEL II IMAGE

Componentele principale ale arhitecturii şi diagramele claselor:

Items Package

Items.Item (Implementează noțiunea abstracta de entitate)

Items.Character (Definește noțiunea abstractă de caracter activ în joc)
Items.Watergirl (Implementează personajul principal al jocului)
Items.ItemsManager (Manager de entități prezente în joc)
Items.Statics.StaticEntity(Defineste notiunea abstracta de entitate statica)
Items.Statics.BlueDiamond (Definește noțiunea abstractă de diamant alb.)
Items.Statics.BlueDoor (Definește noțiunea abstractă de ușa alb.)
Items.Statics.BlueWater (Definește noțiunea abstractă de apa alb.)
Items.Statics.Key(Implementează noțiunea de cheie pentru urm. lvl.)
Items.Statics.RedWater (Definește noțiunea abstractă de apa roșie)



Clasa abstractă *Item este* clasă de bază pentru clasa Character și StaticEntity, *astfel* acestea mostenesc atributele și metodele acesteia. Clasa WaterGirl extinde clasa Character definind astfel personajul principal cu acțiunile și puterile sale.

Clasa ItemsManager defineste managerul de entiati. Aceasta conține obiecte de tip personaj principal dar și obiecte statice, lucrând cu ele.

Metode semnificative:

Items.Item

Metode Public:

- **Item (Handler handler, float x, float y,int width, int height)**
  *Constructor de inițializare al clasei **Item**.*
- **void hurt(int damage)**
  Metodă ce scade din viața atunci când exista un oarecare damage intre itemi.
- **boolean checkEntityCollisions(float xOffset, float yOffset)**
  Metoda verifică coliziunile dintre player și entitatile din joc.

Metode Protected

- **boolean checkCollisions ()**
  Metoda verifică coliziunile pentru a împiedica suprapunerea cu dalele solide.
- ***boolean isOnFloor()***
  Metoda verifica dacă jucătorul se afla pe ceva solid (pamant/gheata).
- ***boolean isOnTop()***
  Metoda care verifica dacă jucătorul lovește ceva cu capul.
- ***void fall()***
  Metoda care implementeaza forța gravitationala.

Items.Character

Metode Public:

- **Character (Handler handler, float x, float y,int width, int height)**
  *Constructor de inițializare al clasei **Character**.*
- **void Move()**
  Metodă ce apeleaza mai multe metode care țin de mișcarea caracterului.
- **void MoveX()**
  Metoda modifică coordonata x decremetând-o atunci când creatura se deplasează la stânga,incremetând-o atunci când creatura se deplasează la dreapta.
- **void MoveY()**
  Metoda modifică coordonata y decremetând-o atunci când creatura se deplasează în sus. incremetând-o atunci când creatura se deplasează în jos.
-

Metode Protected

- **void jump()**
  Metoda aceasta realizeaza saltul caracterului.

Items.WaterGirl

Metode Public:

- **WaterGirl(Handler handler, floatx,floaty,int width, intheight)**
  *Constructor de inițializare al clasei **WaterGirl**.*
- **void checkObstacle()**
  Metoda verifca coliziunea caracterului cu obstacole care pot provoca damage.
- **void die()**
  Prin aceasta funcție se realizeaza diferite operatii atunci când caracterul moare.

Metode Protected

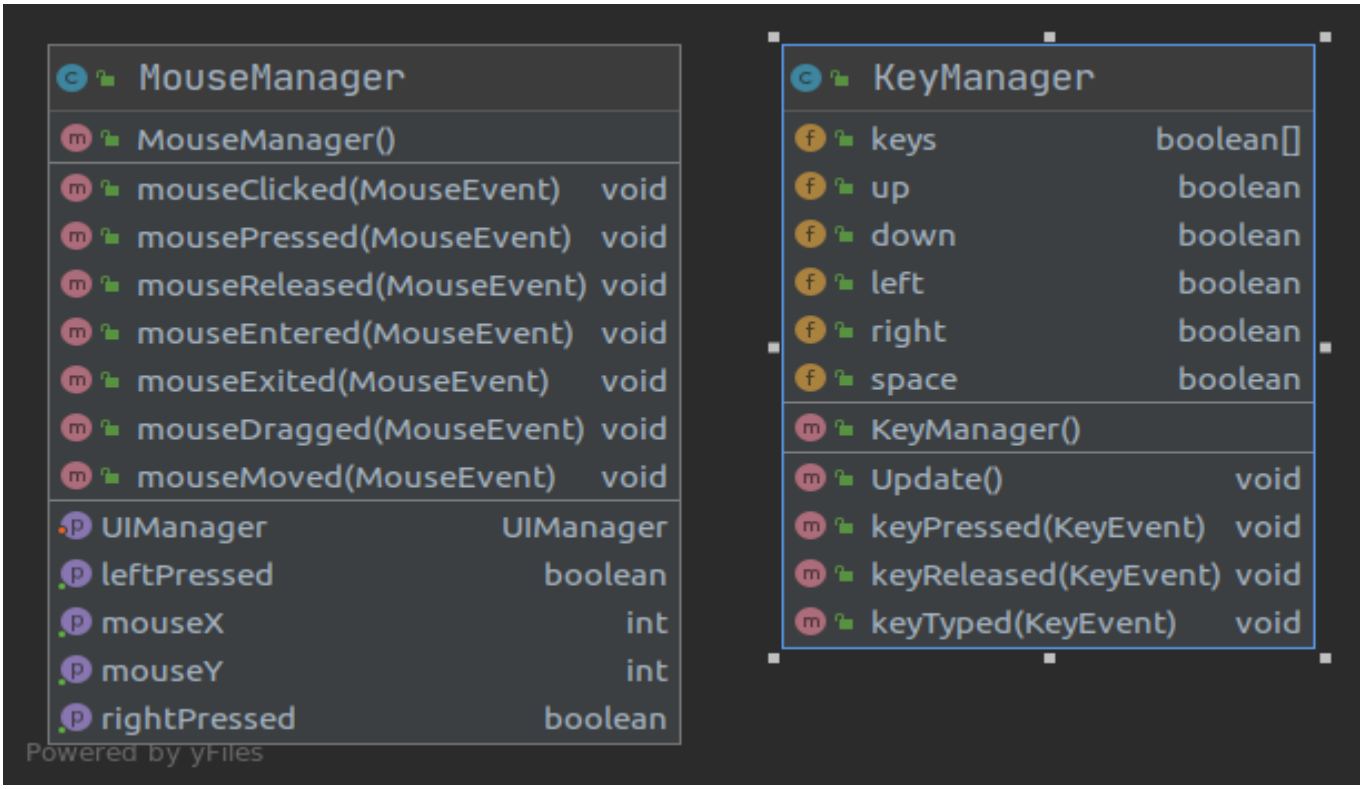
- **void jumpUpdate()**
  Funcție de update a sariturii caracterului.

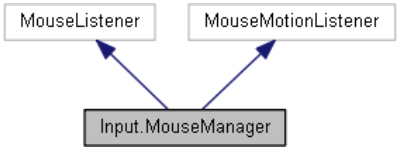
Metode Private

- **void checkCollect()**
  Metodă ce ajuta la colectarea diamantelor, cheilor și trecerea la nivelul urmator.
- **void GetInput()**
  Metoda verifica dacă a fost apasata o tasta din cele stabilite pentru controlul eroului.

Input Package:

Input.KeyManager (Gestionează intrarea (input-ul) de tastatură)
Input.MouseManager(Gestionează intrarea (input-ul) de la mouse)





Metode Public

KeyManager ()

Constructorul clasei.

void update ()

Acutalizează apăsarea tastelor.

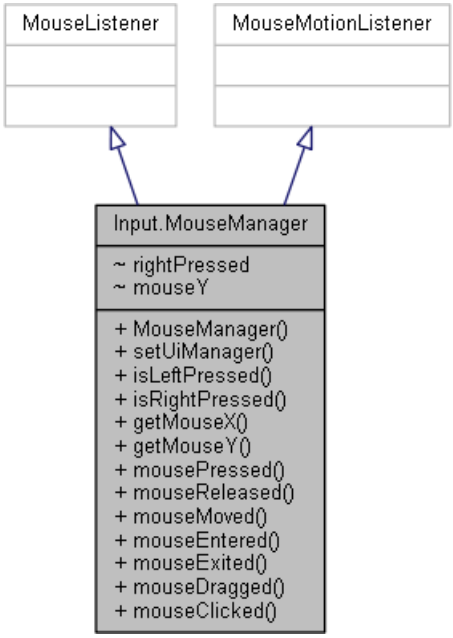
void keyPressed (KeyEvent keyEvent)

Funcție ce va fi apelată atunci când un un eveniment de tastă apăsată este generat.

void keyReleased (KeyEvent keyEvent)

Functie ce va fi apelată atunci când un un eveniment de tastă eliberată este generat.

Clasa citește dacă a fost apăsată o tastă, stabilește ce tastă a fost acționată și setează corespunzător un flag. În program trebuie să se țină cont de flagul aferent tastei de interes. Dacă flagul respectiv este true înseamnă că tasta respectivă a fost apăsată și false nu a fost apăsată.



Metode Public

MouseManager ()

Constructorul clasei.

void setUiManager (UIManager uiManager)

Setează obiectul de tip UIManager.

boolean isLeftPressed ()

Returnează flag-ul de click stânga.

boolean isRightPressed ()

Returnează flag-ul de click dreapta.

int getMouseX ()

Returnează poziția x a cursorului.

int getMouseY ()

Returnează poziția y a cursorului.

void mousePressed (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse apăsat este generat.

void mouseReleased (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse eliberat este generat.

void mouseMoved (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse este mutat.

void mouseEntered (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse introdus.

void mouseExited (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse își termină execuția.

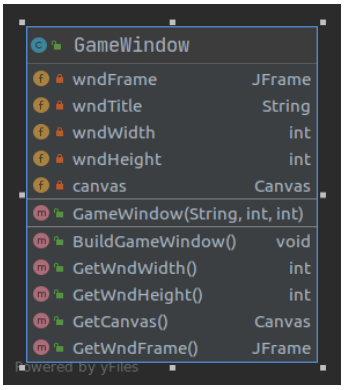
void mouseDragged (MouseEvent mouseEvent)

Funcție ce va fi apelată atunci când un un eveniment de mouse este apăsat și mutat.

void mouseClicked (MouseEvent mouseEvent

Funcție ce va fi apelată atunci când un un eveniment de mouse este apăsat.

Clasele MouseManager și KeyManager reprezintă metodele de interacțiune a utilizatorului cu jocul prin selectarea butoanelor, deplasarea jucătorului și colectarea bonusurilor.

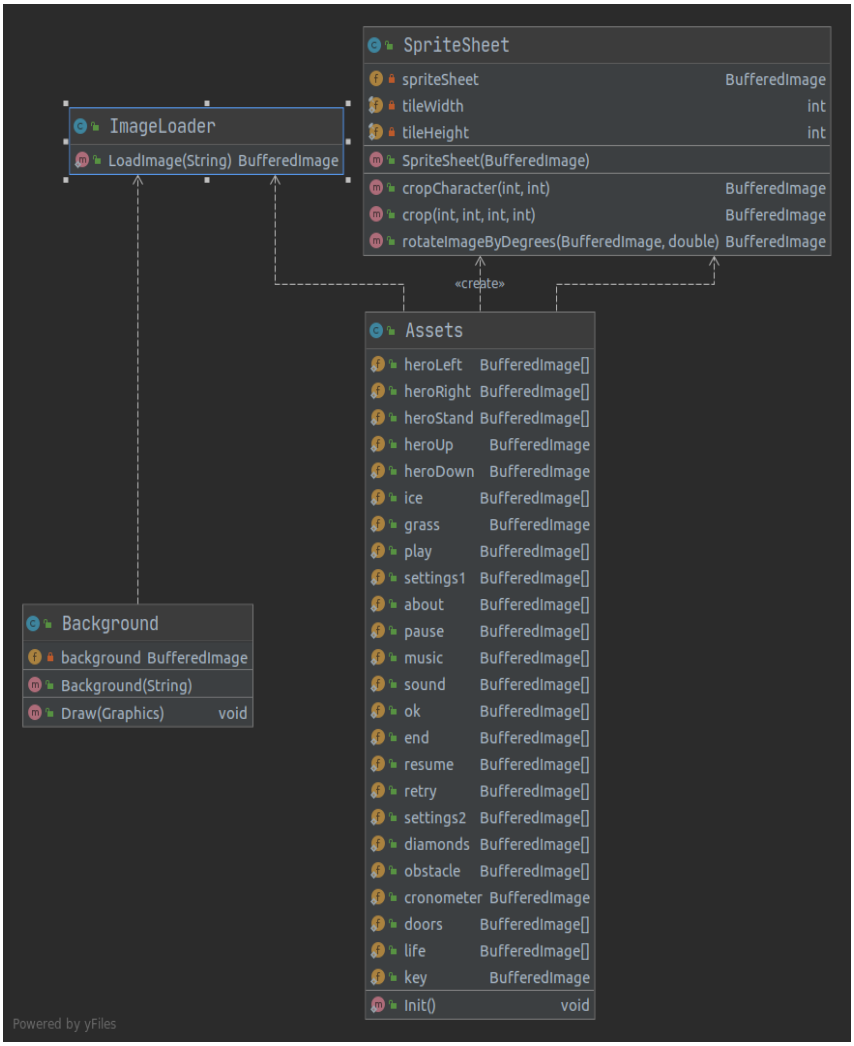


Package GameWindow:

GameWindow.GameWindow(Implementează noțiunea de fereastră a jocului)

Package Graphics:

Graphics.Assets (Clasa incarcă fiecare element grafic necesar jocului)
Graphics.Background (Clasa ce contine backgroundul jocului)
Graphics.ImageLoader (Clasa ce conține o metodă statică pentru încarcarea unei imagini în memorie)
Graphics.SpriteSheet (Clasa reține o referință către o imagine formată din dale (sprite sheet))

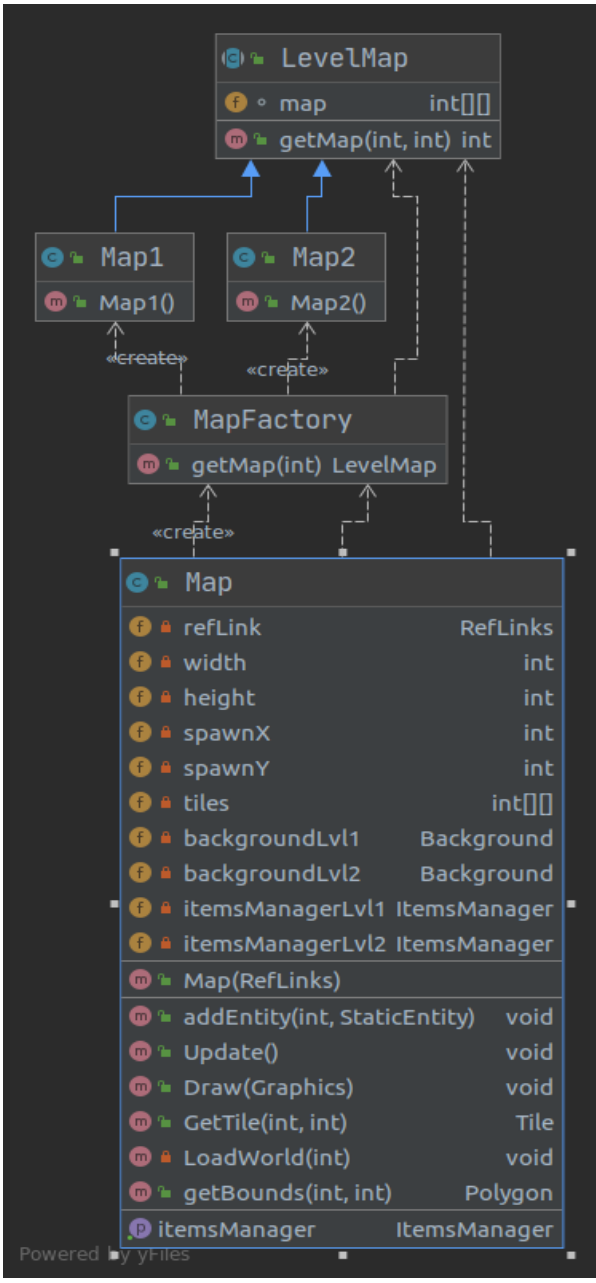


Metoda crop() returnează o dală de dimensiuni fixe (o subimagine) din sprite sheet de la adresa (x * lățimeDală, y * înălțimeDală).
Funcția init() inițializează referințele către elementele grafice utilizate.
Această funcție poate fi rescrisă astfel încât elementele grafice încărcate/utilizate să fie parametrizate. Din acest motiv referințele nu sunt finale.
Clasa ImageLoader oferă posibilitatea citirii imaginilor pentru a fi randate sau folosite, clasa SpriteSheet încarcă imaginile și are metoda de decupare pentru dale.

.

Package Maps:

Maps.LevelMap(Implementează noțiunea abstracta de entitate)
Maps.Map1 (Definește noțiunea abstractă de caracter activ în joc)
Maps.Map2 (Implementează personajul principal al jocului)
Maps.MapFactory (Manager de entități prezente în joc)
Maps.Map(Defineste notiunea abstracta de entitate statica)



Clasa abstractă *LevelMap este* clasă de bază pentru clasa Map1 și Map2, *astfel* acestea mostenesc atributele și metodele acesteia. Clasa MapFactory reprezintă fabrica de harti pentru fiecare nivel în parte.

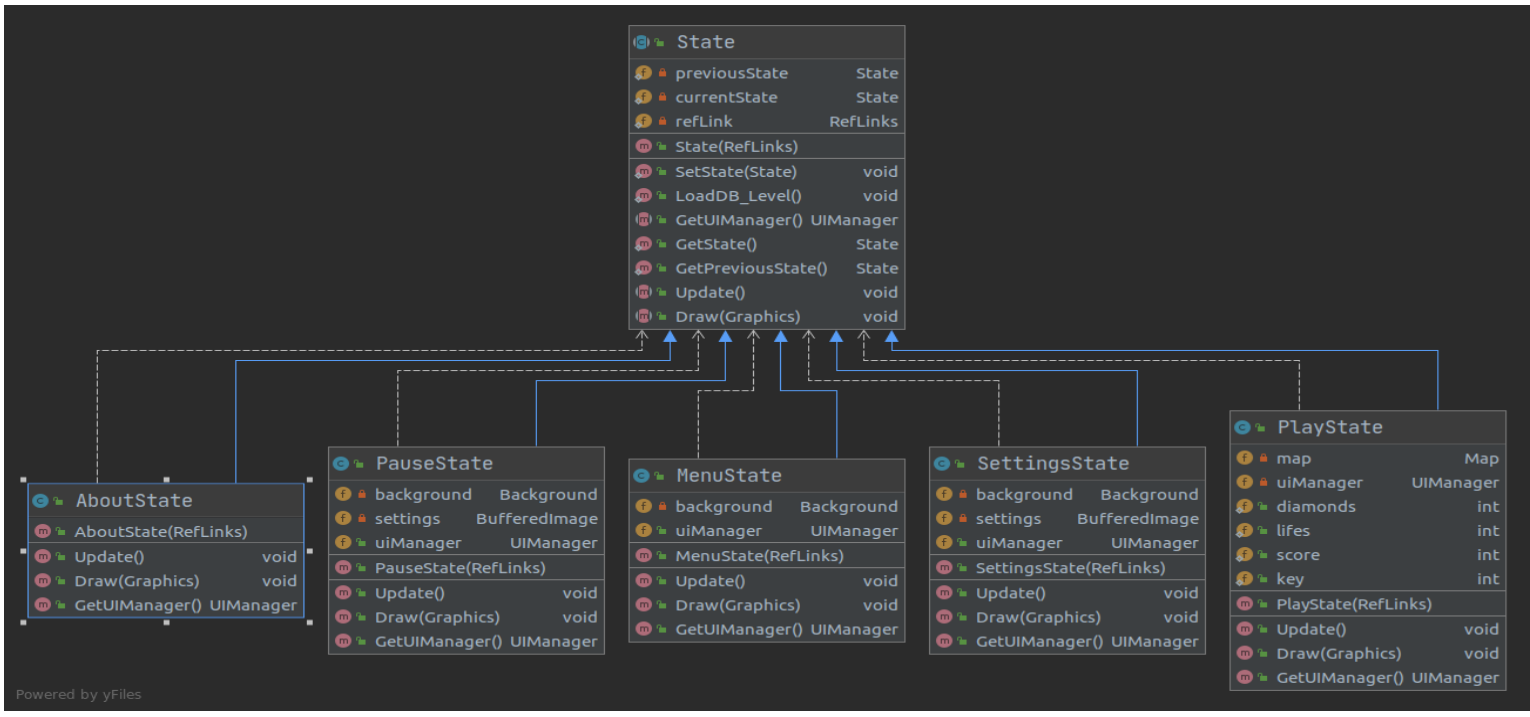
Clasa Map implementeaza notiunea de harta a jocului(cuprinde și entitatile, nu doar dalele).

Metode semnificative:

Maps.Map

Metode Public:

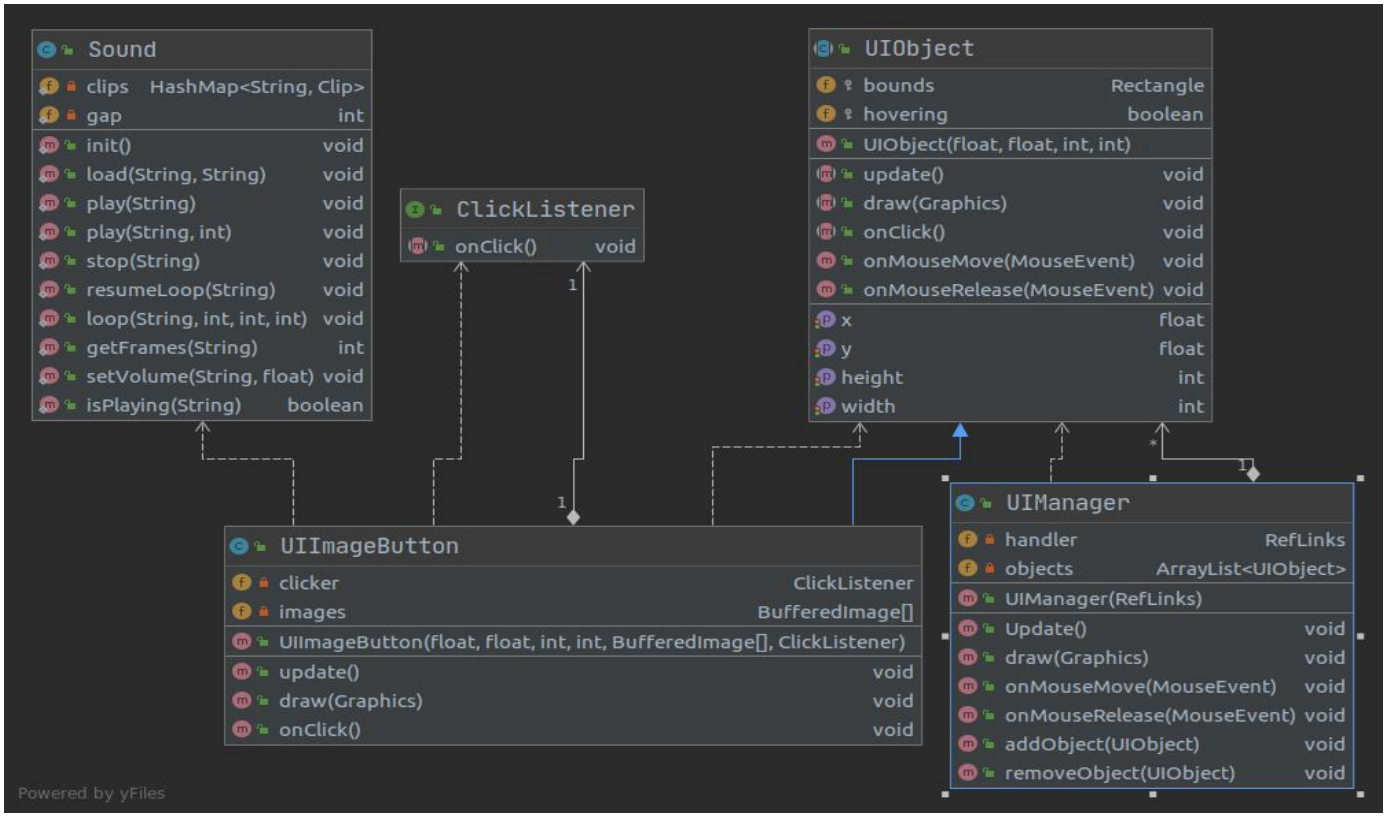
- **Map(Reflinks reflink)**
  *Constructor de inițializare al clasei **Map**.*
- **void Update()**
  Metodă ce actualizeaza harta în funcție de evenimente (un copac a fost taiat).
- **void Draw(Graphics g)**
  Metoda de desenare a hartii.
- **void GetTile(int x, int y)**
  Întoarce o referinta către dala aferenta codului din matrice de dale.
- **void LoadWord(int level)**
  Metoda de încărcare a hartii jocului.
- **Polygon getBounds(int x, int y)**
  Metoda ce returneaza poligonul de coliziune a dalelor.



Package States:

States.State (Clasa abstacta ce defineste notiunea de stare în joc)
States.AboutState (Implementează starea de credits)
States.MenuState (Implementează noțiunea de meniu pentru joc)
States.SettingsState (Implementează meniul de setari pentru joc)
States.PlayState (Implementează starea de play)
States.PauseState (Implementează starea de pauza)

Un joc odată ce este lansat în execuție nu trebuie "să arunce jucătorul direct în luptă", este nevoie de un meniu care sa conțină opțiuni: New Game, Load Game, Settings, About etc. Toate aceste opțiuni nu sunt altceva decât stări ale programului (jocului) ce trebuiesc încărcate și afișate în funcție de starea curenta. Clasa *State definește* cele 5 stări ale jocului: Play*State, MenuState, SettingsState, AboutState și PauseState*. Aceasta este o clasă abstractă ale cărei metode vor fi implementate de clasele ce o extind. Fiecare din acestea conțin în constructorul de inițializare butoanele specifice stării actuale. Astfel, în starea MenuState vom găsi trei butoane: Start - pentru a începe jocul, Settings - pentru setari și About - pentru credits. In PlayState vom găsi doar un buton de Pauza pentru a ne întoarce la meniul de pauza, iar în SettingsState vom găsi doua butoane pentru muzica și sunet.



Package UI:

UI.ClickListener (Implementează noțiunea de click în joc)
UI.UIImageButton (Implementează un obiect de tip UIImageButton
pentru "User Interface")
UI.UIObject (Implementează un obiect de tip UIObject
pentru "User Interface")
UI.UIManager (Implementează noțiunea manager de obiecte
pentru "User Interface")
UI.UIObject (Implementează noțiunea Obiect pentru "User Interface")
UI.Sound(Implementeaza notiunea de sunet pentru joc)

Metode semnificative:

UI.UIImageButton:

Metode Public

UIObject (float x, float y, int width, int height, BufferedImage image)

Constructor, inițializează un obiect-imagine pentru "User Interface".

void update ()

Actualizează starea obiectului pe "User Interface".

void draw (Graphics g)

Desenează obiectul pe "User Interface".

void onClick ()

Implementează noțiunea de click pe obiect.

UI.Sound:

Metode statice Public

void init()

Initializam un hashmap care v-a păstra toate melodiile.

void load(String s, String n)

Încarca melodia s în locația cu cheia n a hashmap-ului creat.

void play(String s, int i)

Pornește melodia s cu pasul de frame i.

void stop(String s)

Oprește melodia s.

void loop(String s, int frame, int start, int end)

Play la melodie într-o bucla.

void setVolume(String s, float f)

Modificam volumul melodiei.

Boolean isPlaying(String s)

Verifica dacă melodia este pornita.

UI.UIManager

Metode Public

UIManager (Handler handler)

*Constructor de inițializare al clasei**UIManager**.*

void update ()

*Actualizează obiectele de tip**UIObject**.*

void draw (Graphics g)

*Desenează obiectele de tip**UIObject**.*

ArrayList< UIObject > getObjects ()

Returnează vectorul de obiecte.

void setObjects (ArrayList< UIObject > objects)

Setează vectorul de obiecte.

void addObject (UIObject o)

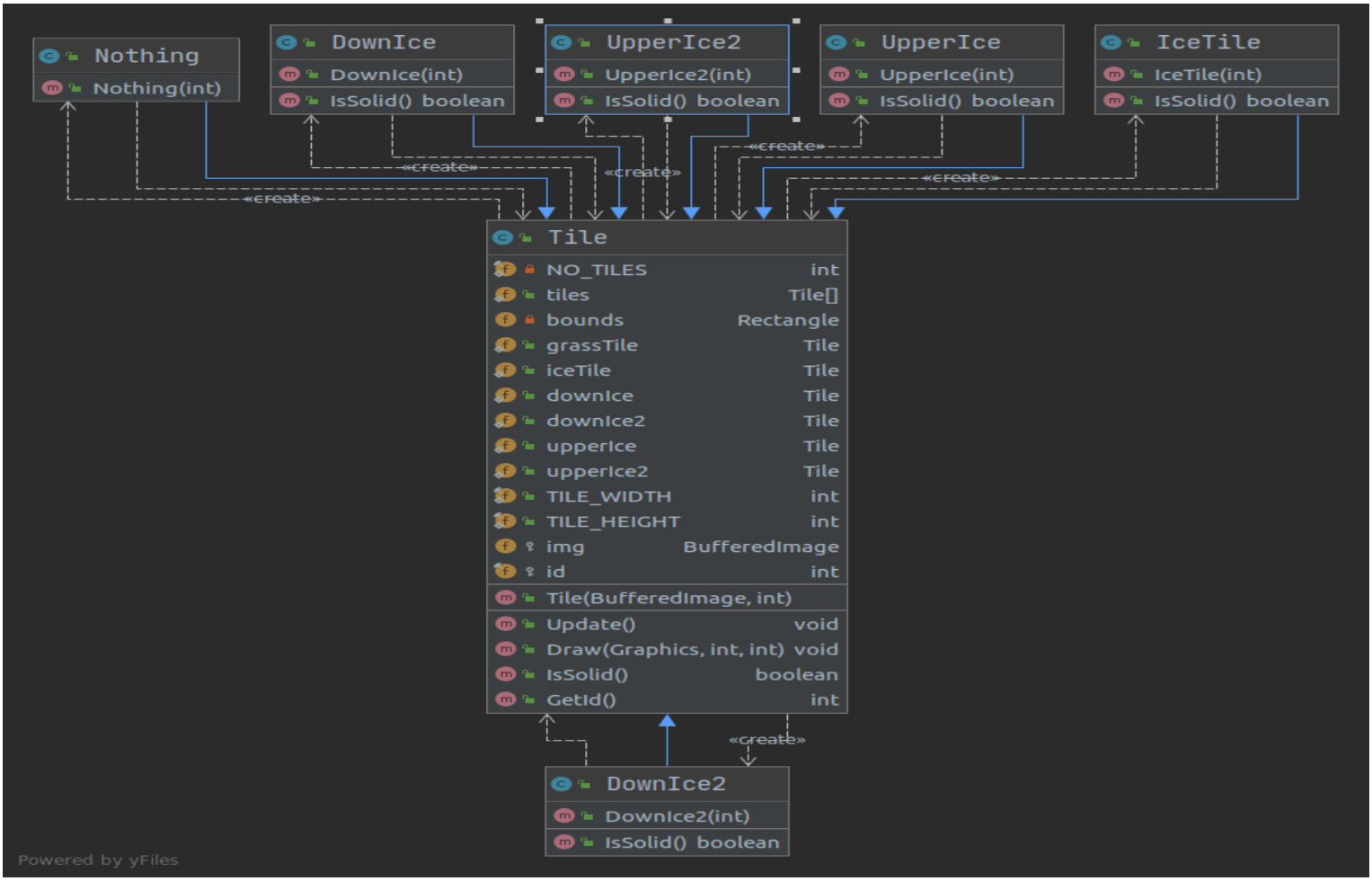
Adaugă un obiect.

void removeObject (UIObject o)

Elimină un obiect.

Package Tiles:

Tiles.DownIce (Bucata de gheață)
Tiles.DownIce2 (Bucata de gheață)
Tiles.IceTile (Abstractizează noțiunea de dală de tip gheață)
Tiles.Nothing (Abstractizează noțiunea de dală goala)
Tiles.UpperIce (Bucata de gheață)
Tiles.UpperIce2(Bucata de gheață)



Metode Public

Tile (BufferedImage texture, int id)

Constructorul aferent clasei.

void update ()

Actualizează proprietățile dalei.

void draw (Graphics g, int x, int y)

Desenează în fereastră dala.

boolean isSolid ()

Returnează proprietatea de dală solidă (supusă coliziunilor) sau nu.

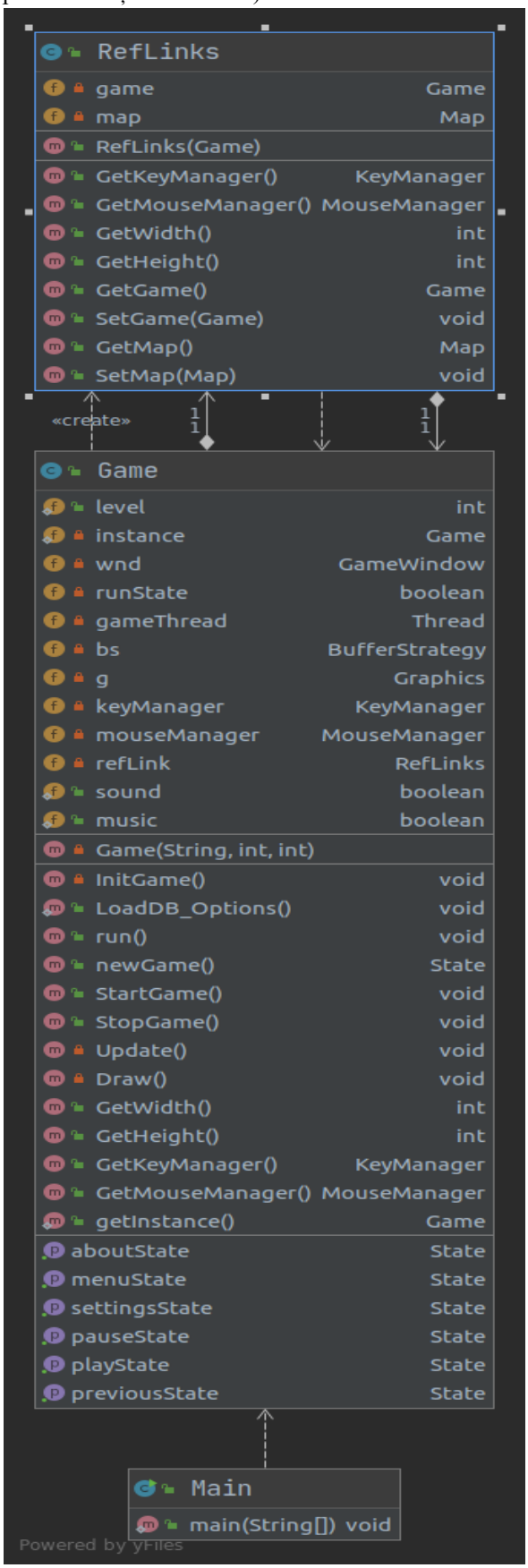
int getId ()

Returnează id-ul dalei

Clasa *Tile* este clasă de bază pentru clasele ce reprezintă dale. Acestea se identifică printr-un ID unic care servește utilizarea dalelor într-un/-o vector/matrice de dale.

Package MainGame:

MainGame.Game (Clasa principală a întregului proiect. Implementează Game - Loop (Update -> Draw))
MainGame.Handler (Clasa ce reține o serie de referințe ale unor elemente pentru a fi ușor accesibile)



Metode Statice Public

Game (String title, int width, int height)
*Constructor de inițializare al clasei **Game**.*

State newGame ()
Metoda returnează o nouă referință către joc.

void run ()
Funcția ce va rula în thread-ul creat.

KeyManager getKeyManager ()
Returnează obiectul care gestionează tastatura.

MouseManager getMouseManager ()
Returnează obiectul care gestionează mouse-ul.

synchronized void start ()
Crează și startează firul separat de execuție (thread).

synchronized void stop ()
Oprește execuția thread-ului.

int GetWidth ()
Returnează lățimea ferestrei.

int GetHeight ()
Returnează înălțimea ferestrei.

State getGameState ()
Returnează referința către joc.

void LoadDB_Options ()
Încarca setarile din baza de date.

Metode Statice Public

static Game getInstance ()
Metoda returnează un obiect de tip Game(SINGLETON).

Metode Private

void InitGame()
Metoda construiește fereastra jocului, initializeaza aseturile, listenerul de tastatura etc.

Interacțiunea dintre componentele sistemului:

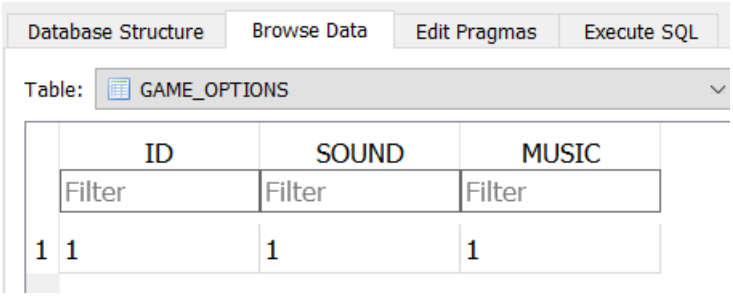
WaterGirl se poate deplasa pe gheață însă trebuie să se grabeasca pentru a nu ingheta și ea. Deasemenea ea trebuie să fie atenta la lava pentru a nu se evapora.

Diagramă UML:

Baze de date

Pentru conceperea acestui joc am folosit 2 baze de date.

1. Game_options: contine o tabela din care se extrag si se scriu setarile jocului in ceea ce priveste muzica si sunetul.



2. Game_level: contine o tabela din care se incarca si se salveaza nivelul la care s-a ramas.