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

RobWorld

Melinte Alexandru-Gicu

1207A

Gameplay (rules): Pentru început, bine ai venit în lumea RobWorld, o lume care este secționată de 2 cadre (cadrul nopții și cadrul zilei ploioase). Aici îl vei găsi pe RobiRo, un robot din viitor, ce dorește a deschide cele două lumi prin intermediul colectării cheilor prezente. Cheile se găsesc pe diferite cutii, în diferite zone, unele dintre ele situându-se deasupra unei ape pline cu pești devoratori de metal. RobiRo are posibilitatea de a sări de pe o cutie pe alta, pe o distanță avantajoasă (fiind robot, puterile lui sunt peste cele umane), și de se mișca stânga-dreapta colectând chei. RobiRo, a fost programat cu 3 vieți. Acesta pierde o viață în momentul în care atinge apa plina cu pești. Dacă RobiRo consumă cele 3 vieți acesta se dezactivează automat, iar jocul este pierdut.

Plot (game story): Jocul se desășoară în *RobWorld,* o lume realizată de *MagRo,* un inginer inteligent din anul 2450. În urma unei pierderi de lichid *RobPower,* lumea lui *MagRo* este secționată în două lumi diferite, care mai de care mai primejdioasă. Pentru a putea readuce stabilitatea, *MagRo* l-a creat pe *RobiRo,* un robot superior rasei umane, cu o capacitate de mișcare ieșită din comun. Acesta fost programat să colecteze toate cheile din fiecare lume, având 3 vieți pentru supraviețuire. Odată colectate, pasul final în restabilirea echilibrului este colectarea steagului lumii *RobWorld.*

Characters:

1. RobiRo este robotul lumii *RobWorld,* creat de *MagRo* pentru a readuce echilibrul. Acesta are capabilitatea de a colecta chei, de a sări la un nivel foarte înalt și sanse de 50% de a primi o putere care ii mărește viteza.

Mechanics:

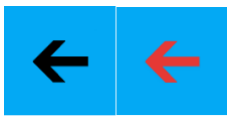
1. **Taste:**
   - pentru deplasare -> *săgeți sus, dreapta, stânga;*
   - pentru colectare -> *tasta space.*
2. **Game points:** *RobiRo* trebuie să colecteze cheile, fiecare cheie aducându-i un punctaj de 2500.
3. **User Interaction:** *RobiRo* trebuie să meargă în dreptul cheii pentru a o colecta, indiferent de poziția în care se află. Odată ajuns lângă cheie, trebuie apăsată *tasta space* pentru a adăuga cheia la numărul de chei contorizate în partea de sus a ecranului.

Win/Lose: RobiRo va câștiga odată cu readucerea echilibrului, adică cu colectarea tuturor cheilor prezente în cele două cadre. Dacă pierde și ultima viață, *RobiRo* se dezactivează, iar echilibrul nu mai poate fi restabilit, jocul încheindu-se.

Spritesheets:

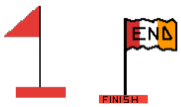
RobiRo + Buton de Start + Buton de Options + elementele lumii RobWorld

Butoane de back:



Butone de exit:

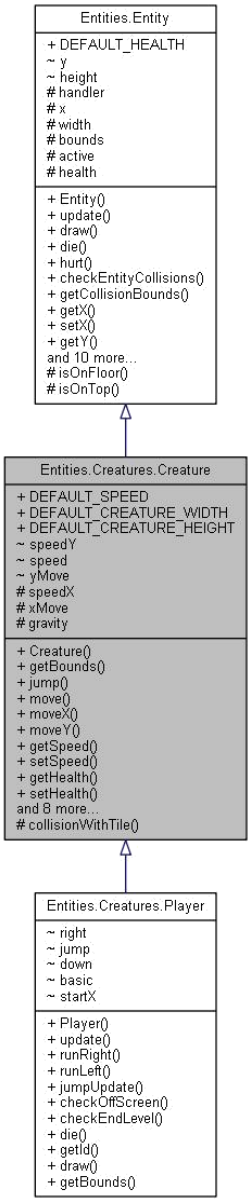
Flag-uri:



Background images:

Rain/clouds:

Componentele principale ale arhitecturii și diagramele claselor:



Entity Package

Entities.Creatures.Creature (Implementează noțiunea creatură prezentă în joc)
Entities.Statics.EndLevel (Definește noțiunea abstractă de steag)
Entities.Entity (Implementează noțiunea de entitate)
Entities.EntityManager (Manager de entități prezente în joc)
Entities.Statics.FinalFlag (Definește noțiunea abstractă de steag final)
Entities.Statics.KeyItem (Definește noțiunea abstractă de cheie)
Entities.Statics.Life (Definește noțiunea abstractă de viață)
Entities.Creatures.Player (Implementează noțiunea de jucător)
Entities.Statics.StaticEntity (Definește noțiunea abstractă de cheie/viață/steag din joc)

Clasa abstractă *Entity* este clasă de bază pentru clasa *Creature*, astfel aceasta mostenește atributele și metodele acesteia. Clasa *Player* extinde clasa *Creature* întrucât jucătorul este o creatură cu alte metode și atribute în plus fiind capabil să colecteze și să se miște. Clasa *StaticEntity* extinde clasa *Entity*, iar aceasta este extinsă de clasele *FinalFlag, KeyItem, Life* și *EndLevel*.

Entities.Creatures.Creature

Metode Public

- **Creature (Handler handler, float x, float y, int width, int height)**
  *Constructor de inițializare al clasei **Creature**.*
- **abstract Rectangle getBounds ()**
  Metodă ce urmează a fi implementată.
- **void jump ()**
  Metoda verifică coliziunile pentru a împiedica străpungerea dalei.
- **void move ()**
  Metoda verifică coliziunile pentru a modifica cele doua coordonate, x și y.
- **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.

Atribute Statice Public

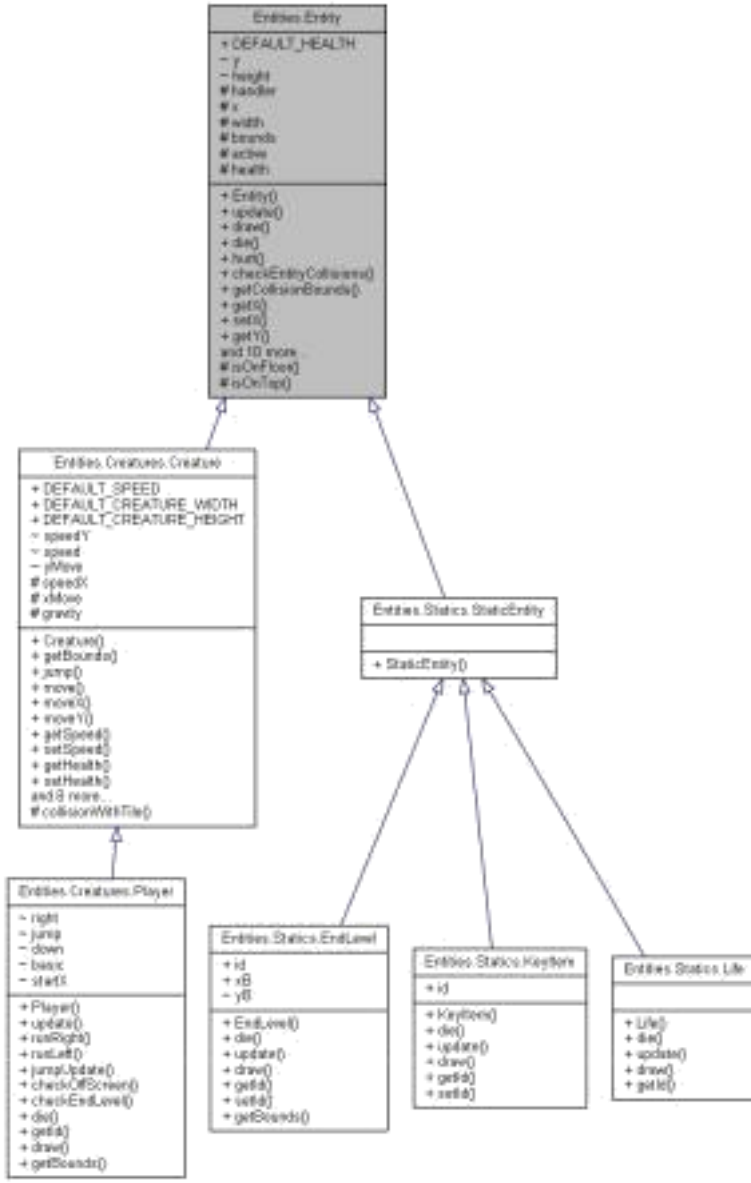
- **static final float DEFAULT_SPEED = 4.0f**
- **static final int DEFAULT_CREATURE_WIDTH = 64**
- **static final int DEFAULT_CREATURE_HEIGHT = 64**

Metode Protected

- **boolean collisionWithTile (int x, int y)**
  Metoda returnează valoarea de adevăr a metodei "isSolid" corespunzătoare dalei de la coordonatele x și y.

Atribute Protected

- **float speedX**
- **float xMove**
- **float gravity = 10f**



Metode Public

- **Entity (Handler handler, float x, float y, int width, int height)**

  *Constructor de inițializare al clasei **Entity**.*

- **abstract void die ()**

  Metoda va fi implementată de clasele ce o vor extinde.

- **void hurt (int tmp)**

  Metoda de distrugere element static.

- **boolean checkEntityCollisions (float xOffset, float yOffset)**

  Această metodă parcurge fiecare entitate și daca una dintre ele se intersectează cu o altă entitate, aceasta returnează 1, altfel 0.

- **Rectangle getCollisionBounds (float xOffset, float yOffset)**

  Metoda de creare a unui dreptunghi de coliziune.

Atribute Statice Public

- **static final int DEFAULT_HEALTH = 100**

Metode Protected

- **boolean isOnFloor ()**

  Metoda de verificare a așezarii, pe partea superioară a dalei, jucătorului.

- **boolean isOnTop ()**

  Metoda de verificare a coliziunii cu partea inferioară a dalei a jucătorului.

Atribute Protected

- **Handler handler**
- **float x**
- **int width**
- **Rectangle bounds**
- **boolean active = true**
- **int health**

Metode Public

- **Player (Handler handler, float x, float y)**

  *Constructor de inițializare al clasei **Player**.*

☺ **void update ()**

Metoda este implementată de clasele ce o extind.

- **void runRight ()**

Implementează mișcarea la dreapta, făcând și actualizare.

- **void runLeft ()**

Implementează mișcarea la stânga, făcând și actualizare.

- **void jumpUpdate ()**

Implementează săritura, făcând și actualizare.

- **void checkOffScreen ()**

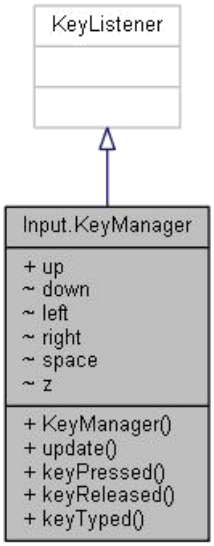
Implementează posibilitatea de a pierde un număr de vieti în joc, la depășirea coordonatei y cât și imposibilitatea de a merge spre o coordonată negativă a lui x.

- **void checkCollect1()**

Implementeaza colectarea obiectelor de pe mapa, verificand atat tastarea lui SPACE cat si coliziunile.

- **void die ()**

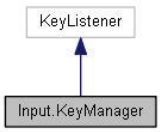
Metoda va fi implementată de clasele ce o vor extinde.



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.

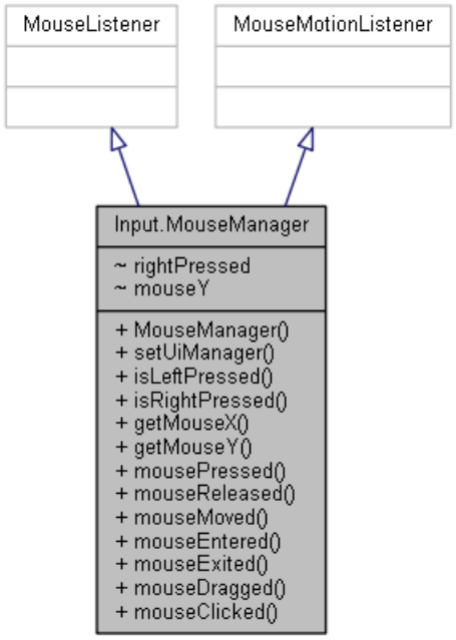
void keyPressed (KeyEvent keyEvent)

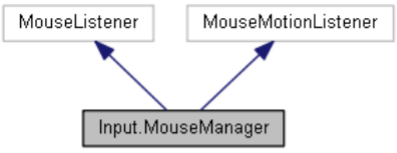
Funcție ce va fi apelată atunci când un un eveniment de tastă apăsată este generat.

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.

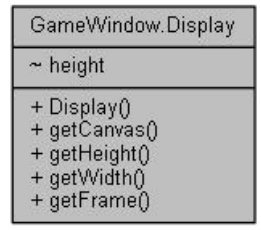
void mouseDragged (MouseEvent mouseEvent)

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 cheilor.

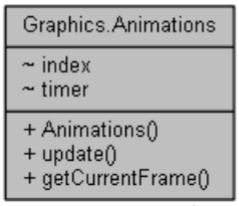


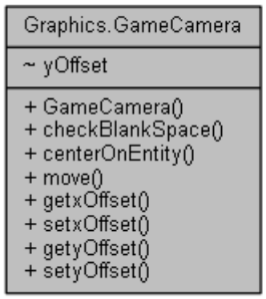
Package GameWindow:

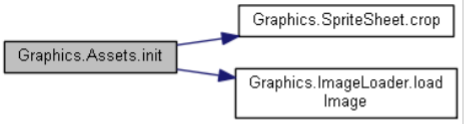
☺ GameWindow.Display (Implementează noțiunea de fereastră a jocului)

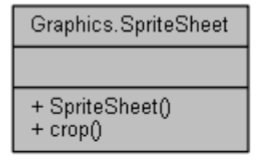
Package Graphics:

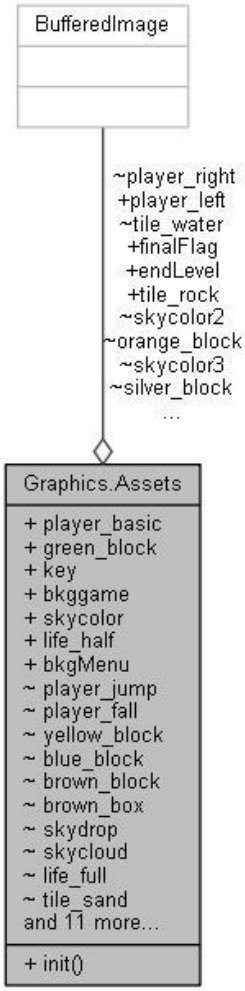
Graphics.Animations (Clasa incarcă animațiile(frame-urile) jucătorului)
Graphics.Assets (Clasa incarcă fiecare element grafic necesar jocului)
Graphics.GameCamera (Clasa ce implementează notiunea de cameră)
Graphics.ImageLoader (Clasa ce conține o metodă statică pentru încarcarea unei imagini în memorie)
Graphics.SpriteSheet (Clasa reține o referință către o imagine formată din dale (sprite sheet))











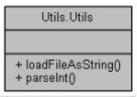
Metoda **crop()** returnează o dală de dimensiuni fixe (o subimagine) din sprite sheet de la adresa (x * lățimeDală, y * înălțimeDală).

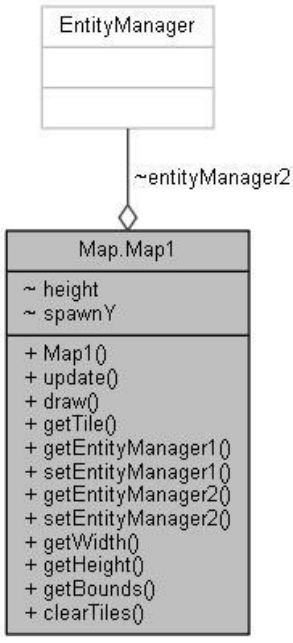
Funcția **init()** inițializează referințele către elementele grafice utilizate. Această funcție poate fi rescrisă astfel încât elementele grafice încărcate/utilizate să fie parametrizate. Din acest motiv referințele nu sunt finale.

Clasa *ImageLoader* oferă posibilitatea citirii imaginilor pentru a fi randate sau folosite, clasa *SpriteSheet* încarcă imaginile și are metoda de decupare pentru dale, clasa *GameCamera* oferă posibilitatea deplasării camerei pe toată suprafața mapei.

Package Utils:

☺ Utils.Utils(Clasă pentru funcții utilitare)





Package Map:

☺ Map.Map1 (Implementează noțiunea de hartă a jocului).

Metode Public

Map1 (Handler handler, String path)

Constructorul de inițializare al clasei.

void update ()

Actualizarea hărții în funcție de evenimente.

void draw (Graphics g)

Funcția de desenare a hărții.

Tile getTile (int x, int y)

Întoarce o referință către dala aferentă codului din matricea de dale.

EntityManager getEntityManager1 ()

Returnează un obiect de tip EntityManager.

void setEntityManager1 (EntityManager entityManager)

Setează un obiect de tip EntityManager.

int getWidth ()

Returnează variabila width.

int getHeight ()

R*eturnează variabila height.*

Rectangle getBounds (int x, int y)

Returnează un dreptunghi de coliziune.

Package State:

- States.GameState (Implementează/controleaza jocul)
- States.Level2State (Implementează/controlează jocul(nivelul 2))
- States.MenuState (Implementează noțiunea de meniu pentru joc)
- States.OptionsState (Implementează ideea de opțiuni prezente în joc)
- States.State (Implementează noțiunea abstractă de stare a jocului/programului)

Metode Public

State (Handler handler)

*Constructor de inițializare al clasei **State**.*

abstract void update ()

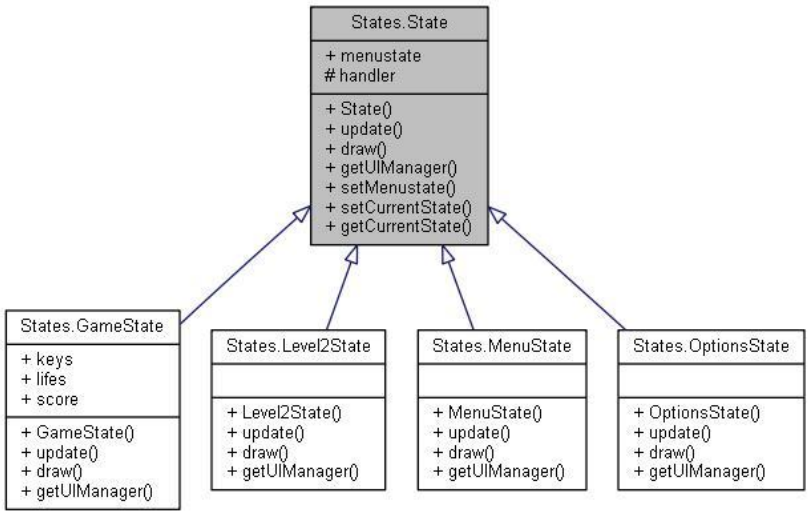
Metodă de actualizare și este implementată de clasele ce o extind.

abstract void draw (Graphics g)

Metodă de desenare și este implementată de clasele ce o extind.

abstract UIManager getUIManager ()

Metoda abstracta ce va fi implementata de clasele ce o extind.



Metode Statice Public

static void setMenustate (boolean menustate)

Setează menustate.

static void setCurrentState (State state)

Setează starea curenta a jocului.

static State getCurrentState ()

Returnează starea curenta a jocului.

Atribute Statice Public

static boolean menustate = false

Atribute Protected

Handler handler

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* implementează cele 4 stări ale jocului: *GameState*, *MenuState, OptionsState și Level2State*. 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, Options - pentru a afla instrucțiunile și Exit - pentru a părasi jocul. In GameState vom găsi doar un buton de Back pentru a ne întoarce la meniu, iar în OptionsState vom găsi un buton de Back pentru a ne întoarce la meniu si un buton de Start pentru a începe jocul.

Package UI:

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

Metode Public

UIImageButton (float x, float y, int width, int height, BufferedImage[] images, ClickListener clicker)
*Constructor, inițializeaza un obiect de tip **UIImageButton** 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.

Metode Public

UIImageOptions (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.

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**.*

Handler getHandler ()
Returnează variabila handler.

ArrayList< UIObject > getObjects ()
Returnează vectorul de obiecte.

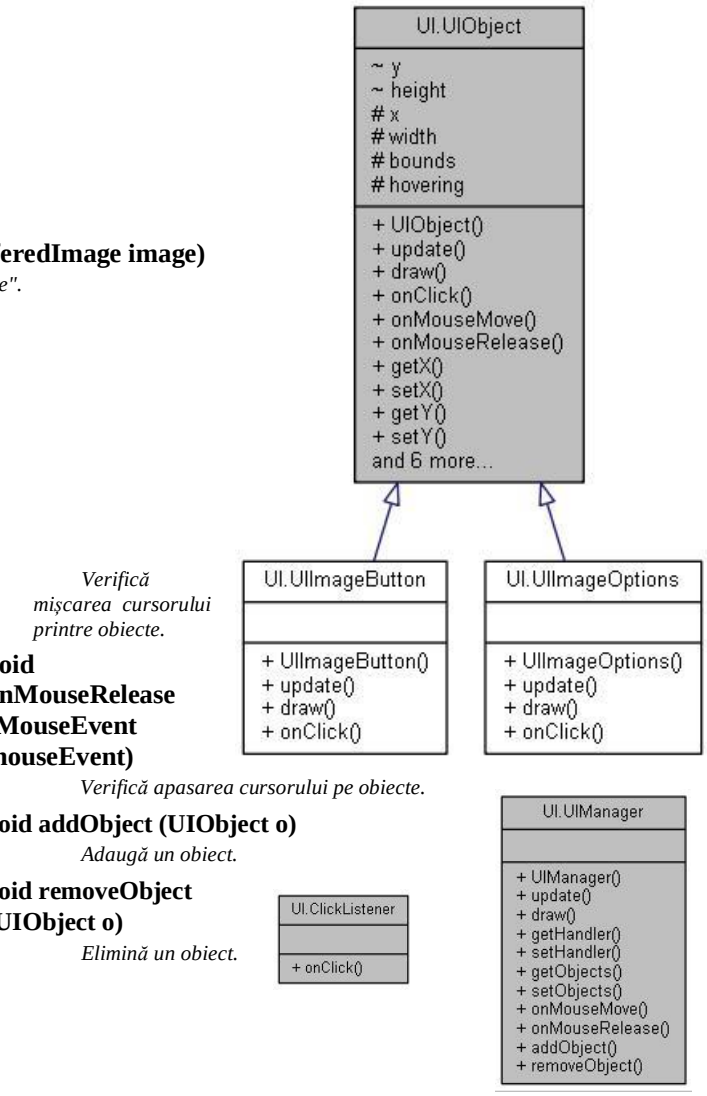
void setObjects (ArrayList< UIObject > objects)
Setează vectorul de obiecte.

void onMouseMove (MouseEvent mouseEvent)
Verifică mișcarea cursorului printre obiecte.

void onMouseRelease (MouseEvent mouseEvent)
Verifică apasarea cursorului pe obiecte.

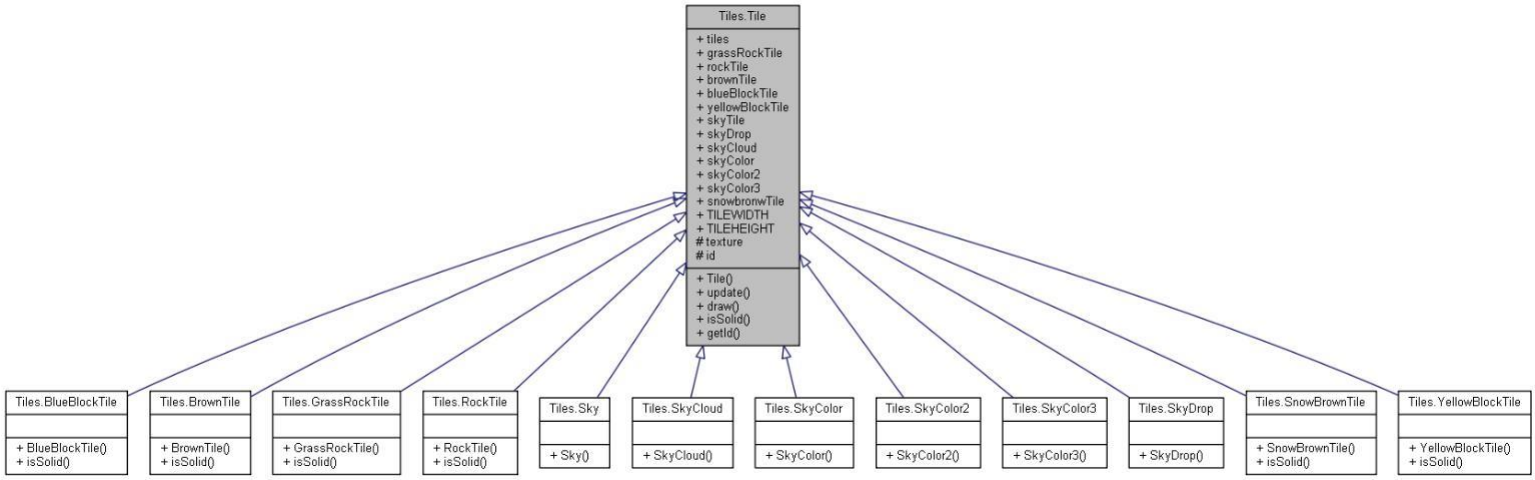
void addObject (UIObject o)
Adaugă un obiect.

void removeObject (UIObject o)
Elimină un obiect.



Package Tile:

Tiles.BlueBlockTile (Abstractizează noțiunea de dală de tip cutie albastră)
Tiles.BrownTile (Abstractizează noțiunea de dală de tip nisip)
Tiles.GrassRockTile (Abstractizează noțiunea de dală de tip iarbă pe piatră)
Tiles.RockTile (Abstractizează noțiunea de dală de tip piatra)
Tiles.Sky (Abstractizează noțiunea de dală de tip cer)
Tiles.SkyCloud (Abstractizează noțiunea de dală de tip cer-nor)
Tiles.SkyColor (Abstractizează noțiunea de dală de tip cer1)
Tiles.SkyColor2 (Abstractizează noțiunea de dală de tip cer2)
Tiles.SkyColor3 (Abstractizează noțiunea de dală de tip cer3)
Tiles.SkyDrop (Abstractizează noțiunea de dală de tip cer-picatura)
Tiles.SnowBrownTile (Abstractizează noțiunea de dală de tip zăpadă-nisip)
Tiles.Tile (Retine toate dalele într-un vector și oferă posibilitatea regăsirii dupa un id)
Tiles.YellowBlockTile (Abstractizează notiunea de dală de tip cutie galbenă)



Atribute Statice Public

static Tile[] tiles = new Tile[NO_TILES]
static Tile grassRockTile = new GrassRockTile(0)
static Tile rockTile = new RockTile(1)
static Tile brownTile = new BrownTile(2)
static Tile blueBlockTile = new BlueBlockTile(3)
static Tile yellowBlockTile = new YellowBlockTile(9)
static Tile skyTile = new Sky(4)
static Tile skyDrop = new SkyDrop(10)
static Tile skyCloud = new SkyCloud(11)
static Tile skyColor = new SkyColor(5)
static Tile skyColor2 = new SkyColor2(6)
static Tile skyColor3 = new SkyColor3(7)
static Tile snowbronwTile = new SnowBrownTile(8)
static final int TILEWIDTH = 48
static final int TILEHEIGHT = 48

Atribute Protected

BufferedImage texture
final int id

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.

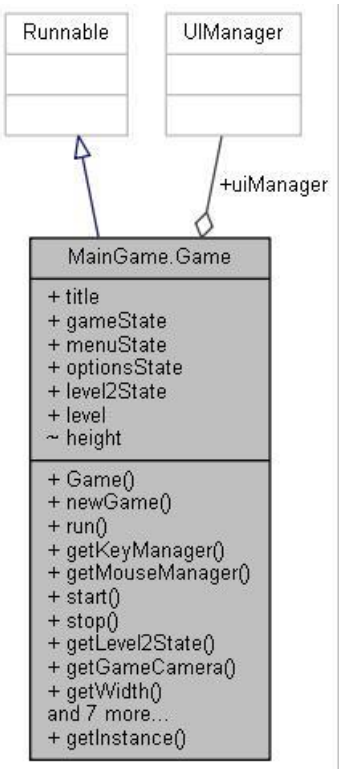
int getId ()

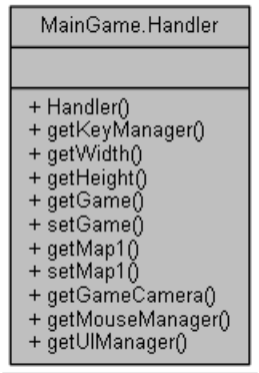
Returnează id-ul dalei

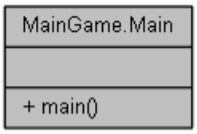
Clasa *Tile* este clasă de bază pentru clasele ce reprezintă dale. Acestea se identifică printr-un ID unic care servește utilizarea dalelor într-unț/-o vector/matrice de dale.

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)
MainGame.Main







Public Member Functions

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 mouseul.

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

synchronized void stop ()
Oprește execuție thread-ului.

State getLevel2State ()
Returnează starea level 2 a jocului.

GameCamera getGameCamera ()
Returnează camera jocului, posibilitatea camerei de a urmări jucătorul.

int getWidth ()
Returnează lățimea ferestrei.

int getHeight ()
Returnează înălțimea ferestrei.

State getGameState ()
Returnează referința către joc.

void setGameState (State gameState)
Setează o nouă referință către joc.

State getMenuState ()
Returnează referința către meniu.

void setMenuState (State menuState)
Setează o nouă referința către meniu.

State getOptionsState ()
Returnează referința către opțiuni.

void setOptionsState (State optionsState)
Setează o nouă referința către opțiun

Metode Statice Public

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

Atribute Public

- String title
- State gameState
- State menuState

- State optionsState
- State level2State
- UIManager uiManager

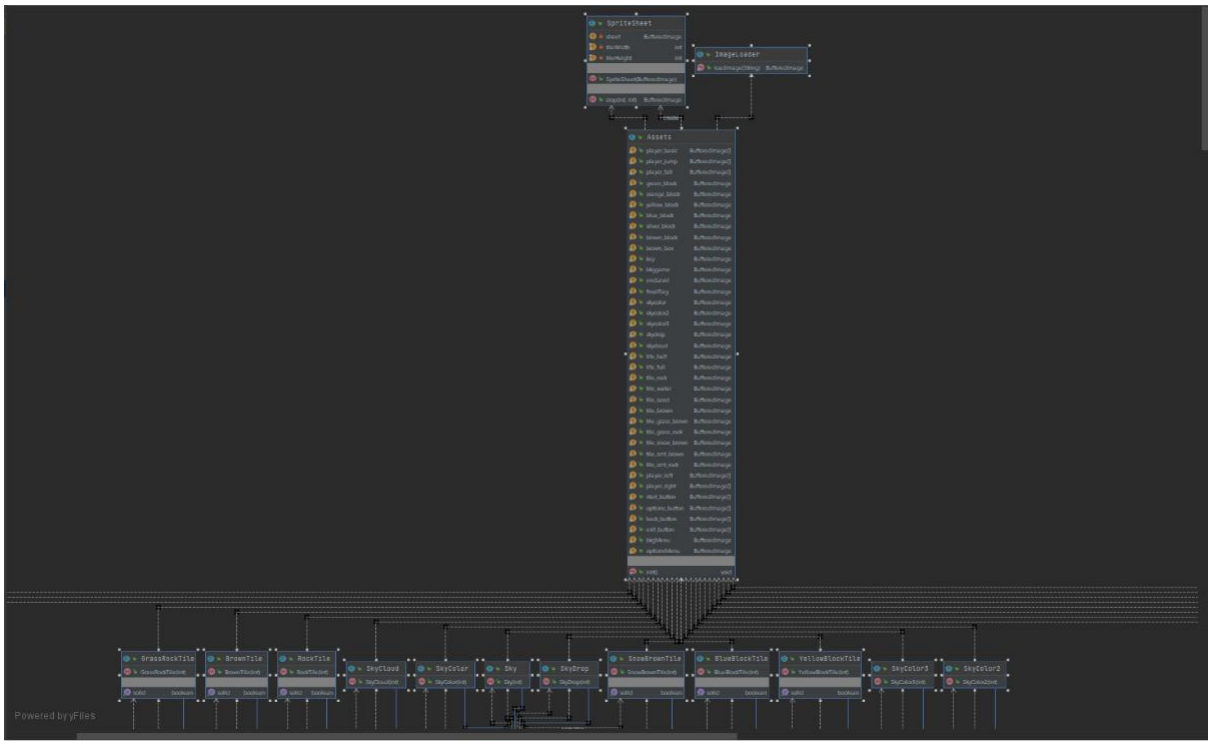
Atribute Statice Public

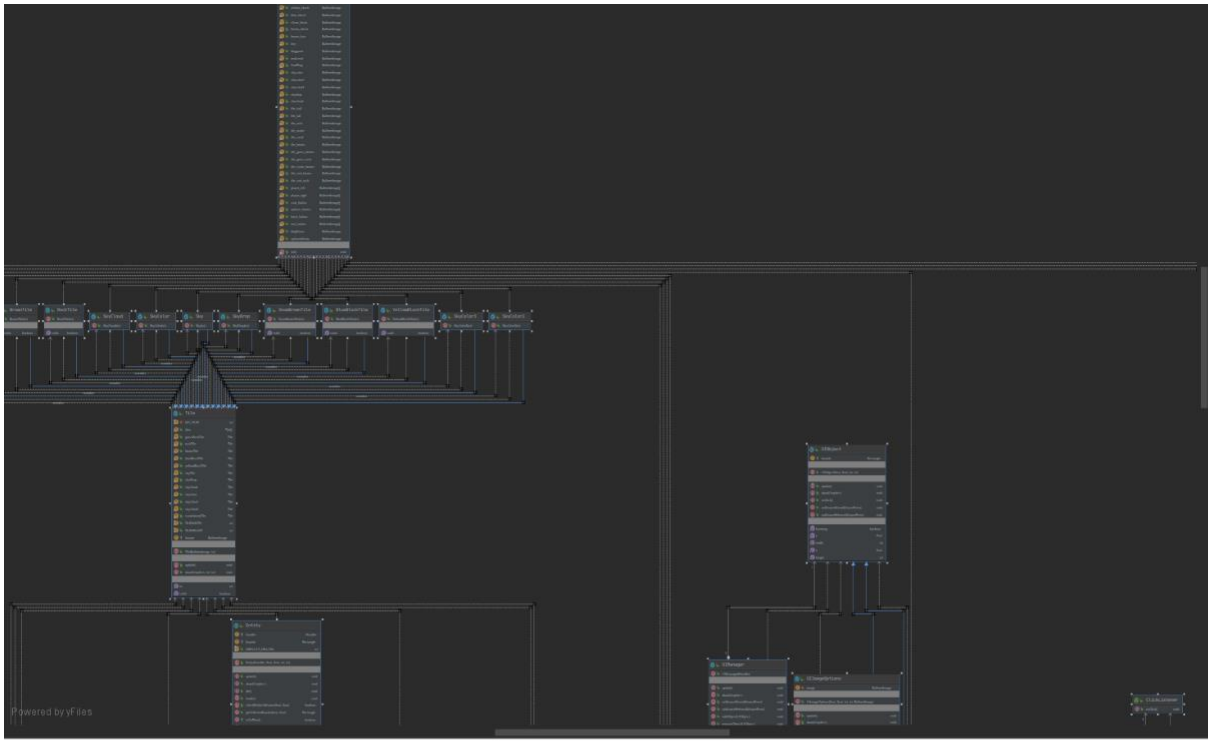
static int level = 1

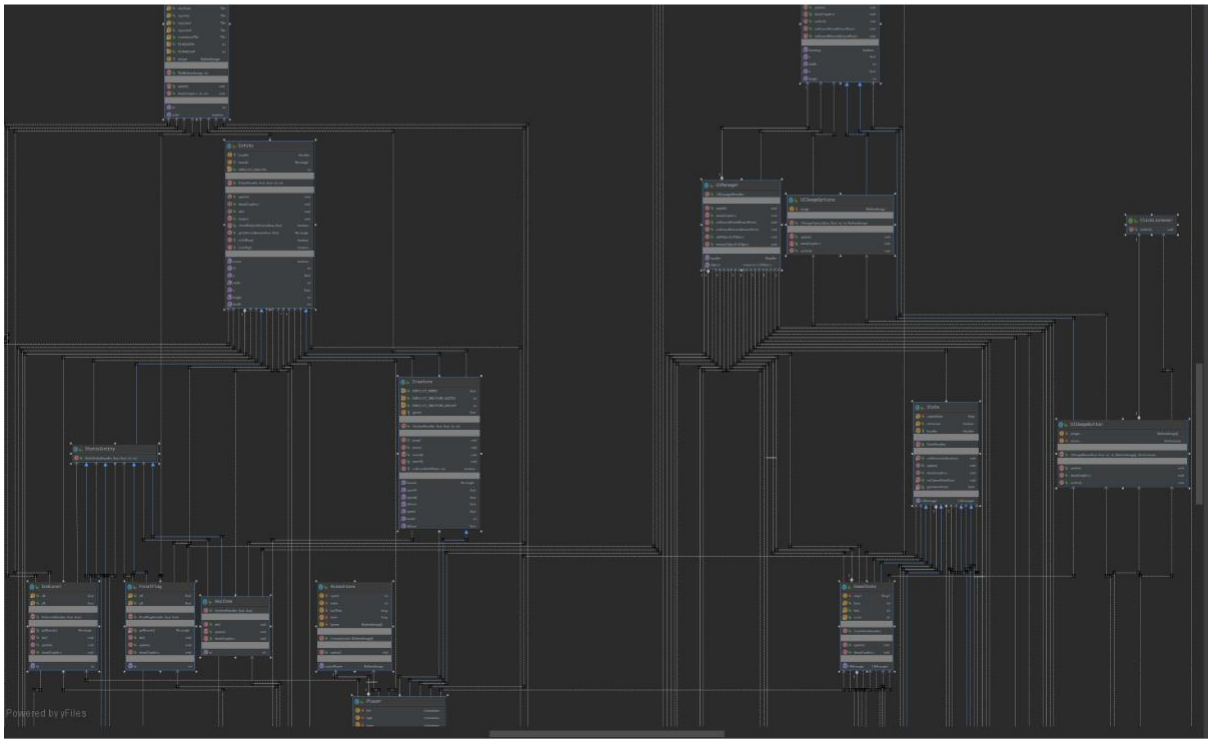
Interacțiunea dintre componentele sistemului:

RobiRo se poate deplasa pe dalele solide pentru a căuta chei, însă dacă acesta întâlnește dale nesolide, va trece prin ele și va pierde o viață.

Diagramă UML:




Powered by yFiles


Powered by yFiles


Powered by yFiles


Powered by yFiles