

HF Dokumentáció – Pac-Man

Svélecz Péter

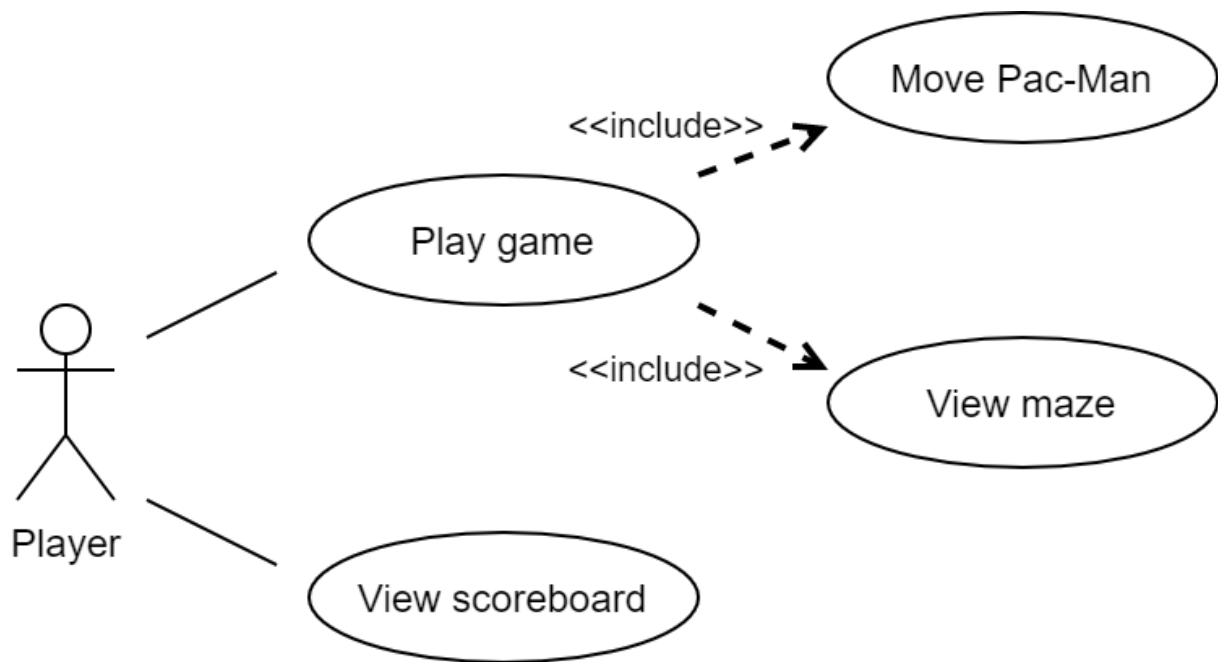
A játékmenet leírása (User manual)

A játék elindítása után meg kell adnunk egy nevet, ezt követően elindul a játékmenet. Ennek során a játékos a nyíl billentyűkkel tudja mozgatni Pac-Man-t a pályán, ami egy labirintus. A labirintusban három féle dolog található Pac-Man-en kívül, ezek a következők: pontok (SimplePellet), szuperpontok (PowerPellet) és szellemek (Ghost). A játék célja, hogy összegyűjtsük az összes pontot. Egy pontot megszerzünk, ha a Pac-Man-nel „megesszük” azt, vagyis ráállunk a pont koordinátaíra. Egy pont megszerzése 100 ponthoz (Score) juttatja a játékost. A szuperpontok a pálya négy különböző pontjában találhatók, ezek 5000 pontot érnek, valamint egy szuperpont megszerzése esetén a szellemek megrémült állapotba (Frightened) kerülnek (egészen addig amíg Pac-Man meg nem eszi az egyik szellemet), ha ilyen állapotukban keresztezik a játékos útját, a Pac-Man megeszi őket, ezzel 15000 ponthoz jutva. Ha viszont a szellemek „alapállapotukban” vannak, csökkentik egyel Pac-Man életét, ha ez nullára csökken, vége a játéknak. Vagyis akkor nyerhetünk, ha minimum egy életünk marad a játék végére és megszereztük az összes pontot. Ha nyertünk, a ranglistát (Scoreboard) megtekintve láthatjuk, hogy az eredményünk bekerült a korábbi nyertesek eredményei közé. (Ha nyerni szeretnénk először érdemes a szuperpontok segítségével megenni a szellemeket és utána összeszedni a pontokat, különben a szellemek véletlenszerű és gyors mozgása miatt elég nehéz dolgunk lesz.)



Képernyőkép a játék kezdeti állapotáról

Use case-ek leírása



A specifikációmban is szereplő Use-case diagram

1 Play game: A játékmenetet jelöli, a játékos a nickneve megadása után irányítja Pac-Mant-t és pontokat gyűjt.

1.1 Move Pac-Man: A játékos a nyíl billentyűk segítségével kijelöli Pac-Man mozgásának irányát.

1.2 View maze: A képernyőn folyamatosan frissül a labirintusban található elemek állapota (Pac-Man és a szellemek, valamint a pontok helyzete).

2 View scoreboard: A játékos megtekinti a ranglistát, amiben egymás alatt szerepelnek a nicknevek, a kör játékidője, valamint a szerzett pontok száma.

Megjegyzések

- A program készítése során sokat olvastam az eredeti Pac-Man-ről az alábbi oldalon: https://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1. Az én verzióm meg sem közelíti az eredeti játék összetettségét, de törekedtem az autentikus kinézet megtartására, így a képernyőméret, illetve a grafikai elemek aránya is azonos az eredetiekkel. A szellemek mozgását azonban jelentősen leegyszerűsítettem, mivel az eredeti verzióban meglehetősen összetett ez a mechanizmus.
- Az osztálydiagram elkészítéséhez az ObjectAid nevű Eclipse extensiont használtam, ha esetleg szükség van rá az ellenőrzés során, innen elérhető: <https://marketplace.eclipse.org/content/objectaid-uml-explorer#group-details>.

Osztályok, adatszerkezetek leírása

Direction (enumeráció): állapotai: UP, DOWN, LEFT, RIGHT. Pac-Man mozgási irányának osztályok közötti átadására szolgál.

Drawable (interfész): az interfészt megvalósító osztályoknak rendelkezniük kell egy public void draw(Graphics g) fejlécű függvénnyel. A játék elemeinek grafikai megjeleníthetőségét jelzi. A következő osztályok valósítják meg: PacMan, Ghost, SimplePellet, PowerPellet.

Position (osztály): egy X és Y koordináta párost reprezentál. Akkor használatos amikor több koordináta mozog objektumok között, illetve a játék során elérhető legális pozíciókat is egy ArrayList<Position> adatszerkezet tárolja, így ez alapján könnyen eldönthető pl. hogy egy játékelem hová mozoghat a következő lépésében.

Pellet (absztrakt osztály): a SimplePellet és PowerPellet osztályok közös őse. Absztrakt, mivel csak az imént említett kétféle pont van a játékban, így nincs értelme példányosítani, de mivel közös ősük van, egy heterogén kollekcióban kerül tárolásra az összes pont, függetlenül azok típusától. Ilyen módon, amikor Pac-Man megesz egy pontot azok típusuktól függően más működést idéznek elő, de ez a hívói oldalról nem látszik.

SimplePellet (osztály): a Pellet leszármazottja, ha Pac-Man megeszi 100 ponthoz juttatja a játékost és eltűnik a Pellet-ek listájából és ezzel egyúttal a képernyőről is.

PowerPellet (osztály): a Pellet osztály másik leszármazottja, Pac-Man 5000 pontot kap érte, ha megeszi, továbbá a szellemek rémült (Frightened) állapotba kerülnek tőle, mely során Pac-Man megeheti azokat.

PacMan (osztály): a játék „főszereplője”, a játékos által irányított Pac-Man-t reprezentálja. Négy irányba mozoghat, ennek megfelelően változik textúrájának iránya. (Ezt a textúraváltást nem a PacMan osztály kezdeményezi, hanem kívülről a GamePanel, mivel így irányváltásonként egyszer kerül csak meghívásra, nem pedig minden egyes képernyőfrissítésnél.)

Ghost (osztály): a Pac-Man-t üldöző szellemek osztálya. Létrehozásakor meg kell adni, hogy milyen képpel jelenjen meg. Mozgása során minden lépése során a következő módon jár el: a jelenlegi koordinátájával szomszédos pozíciók közül véletlenszerűen választ egyet és onnantól kezdve az lesz az új helyzete, de sosem lép az előző 5 lépése során érintett pontokra, így mindegyik szellem véletlenszerű utakat jár be folyamatosan a labirintusban.

GamePanel (osztály): a JPanel osztály leszármazottja és megvalósítja az ActionListener interfészt. A játékelemek megjelenítéséért és frissítéséért felelős. Működése röviden: adott időközönként egy Timer meghívja az osztály actionPerformed metódusát az pedig minden hívásnál végrehajtja a játékelemek mozgatását (amennyiben az szükséges), újra kirajzolja a megváltozott helyzetű (és a statikus helyzetű) elemeket, majd megvizsgálja, hogy vége van-e a játéknak.

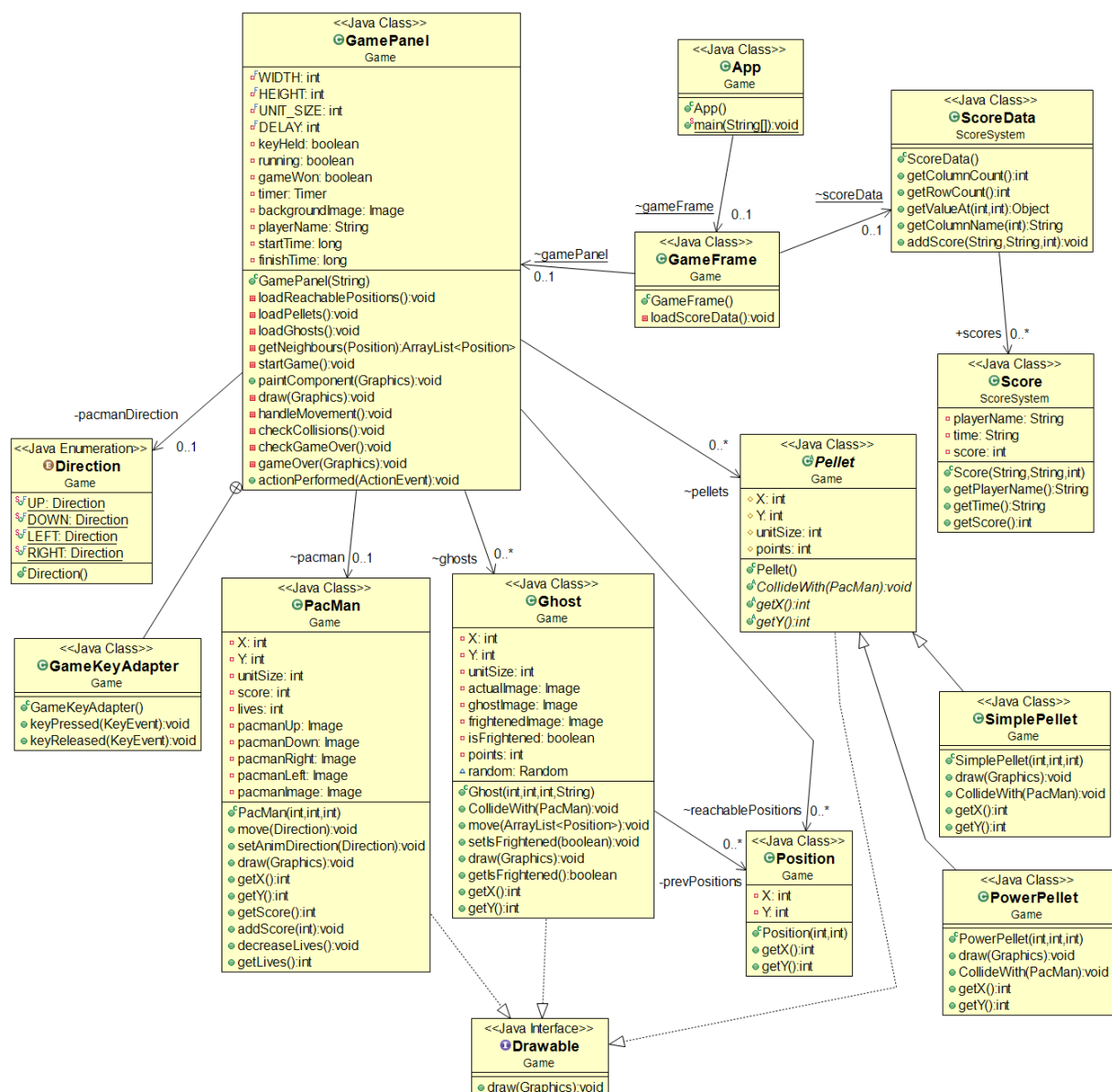
GameFrame (osztály): a JFrame osztály leszármazottja. A játék ablakát reprezentálja, rajta kap helyet a menü, amin keresztül elérjük a ranglistát, valamint a játék felülete is. Létrehozásakor inicializálja a játékhoz szükséges objektumait, ebbe beletartozik a ranglista fájlban tárolt

értékeinek betöltése is. Ő felelős továbbá a játék elején a játékos nevének beolvasásáért is (ezt egy JOptionPane-el valósítja meg).

Score (osztály): a nyertesek adatait tartalmazza, úgy, mint: nicknevük, győzelemhez szükség idejük és az elért pontjuk. Megvalósítja a Serializable interfészt, így fájlba kiírása és onnan való beolvasása könnyen megoldható.

ScoreData (osztály): a ranglistát megjelenítő JTable-höz való adatmodell, az AbstractTableModel osztály leszármazottja, annak megfelelő metódusait felüldefiniálva biztosítja a ranglista megjelenítését és frissülését.

GameKeyAdapter (GamePanel osztálya): a KeyAdapter osztály leszármazottja, a billentyű le- és felengedések figyeléséért felelős, ezek alapján beállítja Pac-Man mozgásának, illetve textúrájának irányát.



Az osztálydiagram

Osztályok belsejének dokumentációja

Direction (enumeráció)

- **Állapotai:** UP, DOWN, LEFT, RIGHT.

Drawable (interfész)

- **public void draw(Graphics g):** az interfészt megvalósító osztály képernyőn való megjelenítését végzi.

Position (osztály)

- **private int X:** a pozíció x koordinátája.
- **private int Y:** a pozíció y koordinátája.
- **public Position(int x, int y):** a két koordináta alapján létrehoz egy Position példányt.
- **public int getX():** X gettere.
- **public int getY():** Y gettere.

Pellet (absztrakt osztály)

- **protected int X:** X koordináta
- **protected int Y:** Y koordináta
- **protected int unitSize:** a játékelemek, „egységének” mérete, egy szorzó, amit az összes Drawable és a GamePanel is használ az arányos képmegjelenítéshez. Értéke megegyezik az eredeti játékban találhatóval, 16 pixel.
- **protected int points:** ennyi ponthoz juttatja PacMan-t, ha megeszi.
- **public abstract void CollideWith(PacMan pacman):** a PacMan-nel való ütközéskor lefutó működés.
- **public abstract int getX():** X gettere.
- **public abstract int getY():** Y gettere.

SimplePellet (osztály)

- **public SimplePellet(int x, int y, int unitSize):** konstruktor, létrehoz egy adott koordinátán elhelyezkedő példányt, a unitSize-t használja a megjelenítési arány kiszámolásához.
- **@Override public void draw(Graphics g):** megjeleníti a pontot a képernyőn.
- **@Override public void CollideWith(PacMan pacman):** amikor PacMan-nel ütközik 100 ponttal növeli eddigi pontjainak számát.
- **@Override public int getX():** X gettere.
- **@Override public int getY():** Y gettere.

PowerPellet (osztály)

- **public PowerPellet(int x, int y, int unitSize):** konstruktor, létrehoz egy adott koordinátán elhelyezkedő példányt, a unitSize-t használja a megjelenítési arány kiszámolásához.
- **@Override public void draw(Graphics g):** megjeleníti a pontot a képernyőn.

- **@Override public void CollideWith(PacMan pacman):** amikor PacMan-nel ütközik 5000 ponttal növeli eddigi pontjainak számát, illetve a még életben lévő szellemeket rémült (Frightened) állapotba teszi, így azokat PacMan meg tudja enni.
- **@Override public int getX():** X gettere.
- **@Override public int getY():** Y gettere.

PacMan (osztály)

- **private int X:** PacMan X koordinátája.
- **private int Y:** PacMan Y koordinátája.
- **private int unitSize:** a megjelenítéshez használt arány.
- **private int score:** ennyi pontja van éppen a játékosnak, alapértéke 0.
- **private int lives:** ennyi élete van éppen a játékosnak, alapértéke 3.
- **private Image pacmanUp:** PacMan felfelé néző textúrája.
- **private Image pacmanDown:** PacMan lefelé néző textúrája.
- **private Image pacmanRight:** PacMan jobbra néző textúrája.
- **private Image pacmanLeft:** PacMan balra néző textúrája.
- **private Image pacmanImage:** PacMan aktuális textúrája, ez változik mozgásának irányától függően.
- **public PacMan(int x, int y, int unitSize):** létrehoz egy PacMan-t az adott X és Y koordinátákon, a játék indulásakor PacMan jobbra néz.
- **public void move(Direction d):** PacMan X és Y koordinátáit változtatja meg a mozgás irányának megfelelően.
- **public void setAnimDirection(Direction d):** PacMan textúrájának irányának settere.
- **@Override public void draw(Graphics g):** megjeleníti PacMan-t a képernyőn.
- **public int getX():** X gettere.
- **public int getY():** Y gettere.
- **public int getScore():** score gettere.
- **public void addScore(int points):** megadott mennyiséggel növeli PacMan pontjait.
- **public void decreaseLives():** egyel csökkenti PacMan életeinek számát.
- **public int getLives():** lives gettere.

Ghost (osztály)

- **private int X:** a szellem X koordinátája.
- **private int Y:** a szellem Y koordinátája.
- **private ArrayList<Position> prevPositions:** a szellem előző 5 pozícióját tároló lista. Azért kell ezeket nyilvántartani, mert mozgása során ezekre nem léphet vissza.
- **private int unitSize:** a megjelenítéshez használt arány.
- **private Image actualImage:** a szellem aktuális textúrája.
- **private Image ghostImage:** a szellem normál textúrája.
- **private Image frightenedImage:** a szellem rémült textúrája, ez mindegyik szellemnek azonos.
- **private boolean isFrightened:** rémült-e jelenleg a szellem.
- **private int points:** ha rémült a szellem és PacMan megeszi, ennyi ponthoz juttatja azt.

- **Random random:** a lépés véletlenszerű irányának megválasztásához használatos.
- **public Ghost(int x, int y, int unitSize, String resourceName):** létrehoz egy szellemet az adott X és Y koordinátákon, a resourceName-ben megadott kép lesz a szellem normál állapotú textúrája.
- **public void CollideWith(PacMan pacman):** a PacMan-nel való ütközés logikája. Ha a szellem nem rémült csökkenti egyel PacMan életét, ha viszont rémült 15000-rel növeli annak jelenlegi pontszámát.
- **public void move(ArrayList<Position> neighbours):** a szellem egy, a jelenlegiével szomszédos pozícióra mozog, de nem lép az előző 5 pozíciójának egyikére sem. Először frissíti az előző pozícióit a jelenlegivel, majd addig generál egy véletlen számot (ami egy index 0 és a neighbours.size()-1 között) amíg legális pozíciót nem talált következő lépéséhez. A szomszédait paraméterben kapja meg.
- **public void setIsFrightened(boolean value):** isFrightened, vagyis a rémült állapot settere, a PowerPellet és a GamePanel osztály vezérli.
- **@Override public void draw(Graphics g):** megjeleníti a szellemet a képernyőn.
- **public boolean getIsFrightened():** isFrightened gettere.
- **public int getX():** X gettere.
- **public int getY():** Y gettere.

GamePanel (osztály)

- **private final int WIDTH:** a játémező szélessége pixelben, értéke egyezik az eredeti játékban találhatóval: 448.
- **private final int HEIGHT:** a játémező magassága pixelben, értéke egyezik az eredeti játékban találhatóval: 576.
- **private final int UNIT_SIZE:** a grafikus elemek megjelenítéséhez használt szorzó, értéke egyezik az eredeti játékban találhatóval: 16 pixel. Egy „rácselem” pl. a játékban 16x16 pixeles.
- **private final int DELAY:** a képfrissítéseket kezelő timer késleltetése, ha nulla lenne, az minden ezredmásodpercben kerülne meghívásra, értéke: 60, így a játék 1000/60 azaz kb. 16 FPS-el fut. Ennek értékétől függ PacMan és a szellemek mozgásának sebessége, valamint általában is a képfrissítések gyakorisága.
- **private boolean keyHeld:** le van-e nyomva jelenleg valamelyik nyílbillentyű, alapértéke: false.
- **private boolean running:** fut-e még a játék, azaz nyert/vesztett/játszik a játékos, alapértéke: false.
- **private boolean gameWon:** megnyerte-e a játékot a játékos.
- **private Timer timer:** a képfrissítés gyakoriságát kezelő Timer.
- **private Image backgroundImage:** a JPanel háttérképe, vagyis a labirintus képe.
- **private String playerName:** az aktuális játékos neve.
- **ArrayList<Position> reachablePositions:** lista a pályán legálisnak számító X, Y koordinátákról.
- **ArrayList<Pellet> pellets:** a labirintusban található pontok (SimplePellet és PowerPellet példányokat tartalmazó heterogén kollekció).

- **ArrayList<Ghost> ghosts:** a labirintusban található szellemek.
- **PacMan pacman:** a labirintusban található PacMan.
- **private Direction pacmanDirection:** PacMan mozgásának jelenlegi iránya.
- **private long startTime:** a játék kezdetén a rendszeridő ezredmásodpercben.
- **private long finishTime:** a játék végén a rendszeridő ezredmásodpercben.
- **public GamePanel(String playerName):** az aktuális játékmenethez létrehoz egy GamePanel-t, a játékos nevét paraméterben kapja meg. Betölti a legális koordináták listáját egy fájlból, majd a pontokat, illetve a szellemeket is elhelyezi azok kezdőpozícióiban.
- **private void loadReachablePositions():** egy szöveges fájlból betölti a játék során elérhető koordinátákat.
- **private void loadPellets():** a koordináták alapján betölti a labirintusban megjelenő pontokat.
- **private void loadGhosts():** a koordináták alapján betölti a szellemeket.
- **private ArrayList<Position> getNeighbours(Position p):** egy adott pozícióhoz visszaadja az azzal szomszédos pozíciók listáját, ezt a szellemek használják mozgásukhoz.
- **private void startGame():** elindítja a játékmenetet.
- **@Override public void paintComponent(Graphics g):** megjeleníti a JPanel-t.
- **private void draw(Graphics g):** a paintComponent(Graphics g), illetve a repaint() metódusok hívják meg, elvégzi a Drawable-k draw() metódusának meghívását. Továbbá, ha vége a játéknak a győztes/vesztes képernyőt fogja kirajzolni.
- **private void handleMovement():** PacMan és a szellemek mozgásért felelős függvényeit hívja meg, PacMan-ét csak akkor ha a kívánt irányban valóban egy legális pozíció található.
- **private void checkCollisions():** vizsgálja, hogy PacMan koordinátái megegyeznek-e egy Pellet/Ghost-ével, ennek megfelelően meghívja azok CollideWith(PacMan pacman) metódusait, ha szükséges eltávolítja a Pellet/Ghost-okat a listákból, így azok többet nem jelennek meg a képernyőn.
- **private void checkGameOver():** a játék vége feltételeket vizsgálja, ha PacMan-nek nulla élete van akkor veszített a játékos, ha ennél több és elfogyott az összes pont a labirintusból, nyert a játékos.
- **private void gameOver(Graphics g):** a draw(Graphics g) függvény hívja meg, hogy megjelenítse a megfelelő győztes/vesztes képernyőt.
- **public void actionPerformed(ActionEvent e):** ha a játék fut a következő metódusokat hívja meg ilyen sorrendben: handleMovement(), checkCollisions(), checkGameOver(), ezek után pedig a repaint()-et, így az elemek megváltozott helyzete/állapota a képernyőn is frissül. A timer hívja meg a beállított késleltetés alapján rendszeresen, ez a „game loop”.
- **@Override GameKeyAdapter.keyPressed(KeyEvent e):** a keyHeld-et true-ra állítja, emellett PacMan mozgásának irányát és ehhez tartozó textúrját is frissíti.
- **@Override GameKeyAdapter.keyReleased(KeyEvent e):** a keyHeld-et false-ra állítja.

GameFrame (osztály)

- **static JPanel gamePanel:** a futás során használt egyetlen JPanel példány.
- **static ScoreData scoreData:** a futás során használt egyetlen ScoreData példány.
- **public GameFrame():** az osztály konstruktora, inicializálja az egész ablakot. Első lépésben betölti a ranglista tartalmát. Ezután bekéri a jelenlegi játékos nevét a JOptionPane.showInputDialog() segítségével. Ha a név mező üresen marad, vagy Cancel-t nyom a felhasználó ennek értéke „Player” lesz. Ezután elhelyezi a menüt és a GamePanelt önmagán és megjelenik a képernyőn.
- **@Override public void scoreBoard.actionPerformed(ActionEvent e):** a scoreBoard (típusa JMenuItem) kattintásra aktiválód ActionListener, megjeleníti a ranglistát a JOptionPane.showMessageDialog() segítségével.
- **@Override public void windowClosing(WindowEvent e):** az ablak bezárását figyelő listener, meghívásakor a scores.dat fájlba menti a ranglista jelenlegi tartalmát.
- **private void loadScoreData():** a konstruktor hívja meg, betölti a ranglista tartalmát a scores.dat fájlból.

Score (osztály)

- **private String playerName:** a játékos neve.
- **private String time:** a játékos nyereséhez szükséges ideje.
- **private int score:** a játékos pontja.
- **public Score(String playerName, String time, int score):** létrehoz egy Score típusú objektumot a paraméterben adott névvel, idővel és ponttal.
- **public String getPlayerName():** playerName gettere.
- **public String getTime():** time gettere.
- **public String getScore():** score gettere.

ScoreData (osztály)

- **public ArrayList<Score> scores:** a ranglistát felépítő Score példányok listája.
- **@Override public int getColumnCount():** a Score mezőinek számát adja vissza, ez 3.
- **@Override public int getRowCount():** a scores.size() gettere.
- **@Override public Object getValueAt(int rowIdx, int colIdx):** egy adott x, y koordinátán található mező értékét adja vissza a táblázatból.
- **@Override public String getColumnName(int idx):** egy adott oszlop nevét adja vissza a paraméterben kapott indexhez.
- **public void addScore(String playerName, String time, int score):** segítségével új Score objektumot adhatunk a jelenlegiekhez.

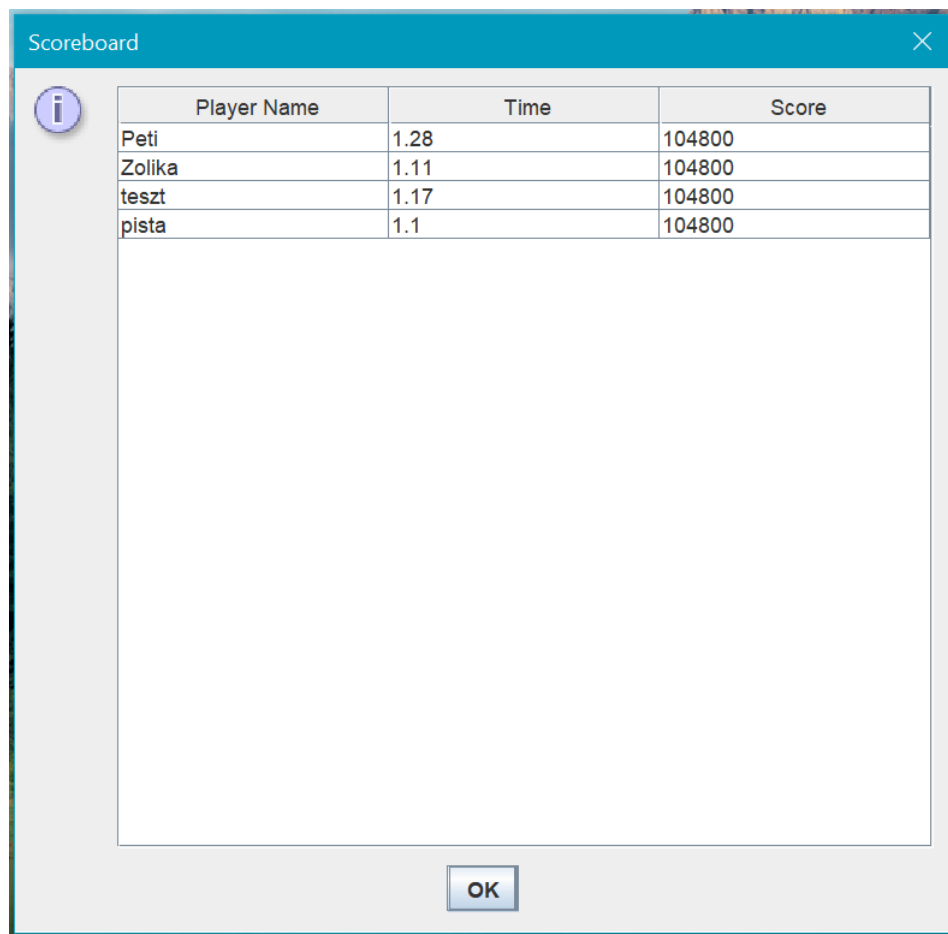
Fájlok szerkezetének leírása

scores.dat: Score típusú sorosítható objektumokat tárol.

mazedata.txt: a labirintus PacMan és a szellemek által elérhető koordinátáit tartalmazza soronként a következő formátumban: [x koordináta];[y koordináta]. Példa a fájlból:

1;4
2;4
3;4
4;4
5;4
6;4
7;4
8;4
9;4
10;4

Ezeket a koordinátákat a program sosem használja „önmagában”, mindig, amikor X-el és Y-al leírt pozíciókkal számol azok fel vannak szorozva a UNIT_SIZE-al, aminek értéke 16.



The screenshot shows a window titled "Scoreboard" with a close button (X) in the top right corner. On the left side of the window is a vertical bar containing an information icon (i). The main area of the window contains a table with three columns: "Player Name", "Time", and "Score". The table has four data rows. Below the table is a large empty rectangular area, and at the bottom center is an "OK" button.

Player Name	Time	Score
Peti	1.28	104800
Zolika	1.11	104800
teszt	1.17	104800
pista	1.1	104800

A ranglista