Neptun kód: ASPP08

Feladat

Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a labirintus sárkányát.

Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik irányba. Ha a sárkány a játékossal szomszédos területre jut, akkor a játékos meghal. Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a labirintus felépítését, távolabb nem. Tartsuk számon, hogy a játékos mennyi labirintuson keresztül jutott túl és amennyiben elveszti az életét, mentsük el az adatbázisba az eredményét. Egy menüpontban legyen lehetőségünk a 10 legjobb eredménnyel rendelkező játékost megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot indítani egy másik menüből. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon kezdjenek.

Elemzés[1]

Minden GameLevel jellemzői: gameID(GameID), rows(int), cols(GameID), level(LevelItem[][]), player(Position), dragon(Position), numBoxes(int), numBoxesInPlace(int), doneLevels(int), gameEnd(boolean)

Minden Game jellemzői: gameLevels(GameLevel), database(Database), isBetterHighScore(boolean)

Minden LevelItem jellemzői: EXIT('$'), DRAGON('*'), WALL('#'), EMPTY(' '), DRAGON_OUTSIDE('*')

Board jellemzői: game(Game), scale(double), scaled_size(int), tile_size(int)

MainWindow jellemzői: game(Game), board(Board), gameStatLabel(JLabel), currLevel (int) , doneLevels(int) , time(Timer) , k(int)

[1] Ez az elemzés rész a hallgatói beadandó dokumentációjából elhagyható, az átalakítási táblázatokat a tervezés részben elég feltüntetni.

Terv[1]

Szükségünk lesz egy Board osztályra a Board konstruktoron keresztül fogjuk megadni. Ezen felül létre kell hoznunk a megfelel metódusokat, setScale(double scale), boolean refresh(), paintComponent(Graphics g)

Szükségünk lesz továbbá egy MainWindow osztályra,

WASD lenyomásakor lép a játékos és a sárkány, amíg ki nem viszi a pályát vagy meg nem hal,ha kivitte jön a következő szint(1-10),ezt a GameLevel fogja kezelni

metódusoai:

refreshGameStatLabel() – label frissítése

createGameLevelMenuItems(JMenu menu) – Újrakezdés

createScaleMenuItems(JMenu menu, double from, double to, double by) – Nagyítás

Szükségünk lesz továbbá egy Game osztályra, ami a szinteket kezeli,illetve a databaset

Szükségünk lesz továbbá egy GameLevel osztályra, ami a konstruktorában létrehozza a pályát és a mozgásokat, metódusoai: isValidPosition(Position p), isFree(Position p), isExit(Position p), movePlayer(Direction d), isKilled(),moveDragon(Direction d), printLevel(),getNumBoxes(),getNumBoxesInPlace(),getDoneLevels(),setDoneLevels(int doneLevel), getGameEnd()

Szükségünk lesz továbbá egy LevelItem enumra

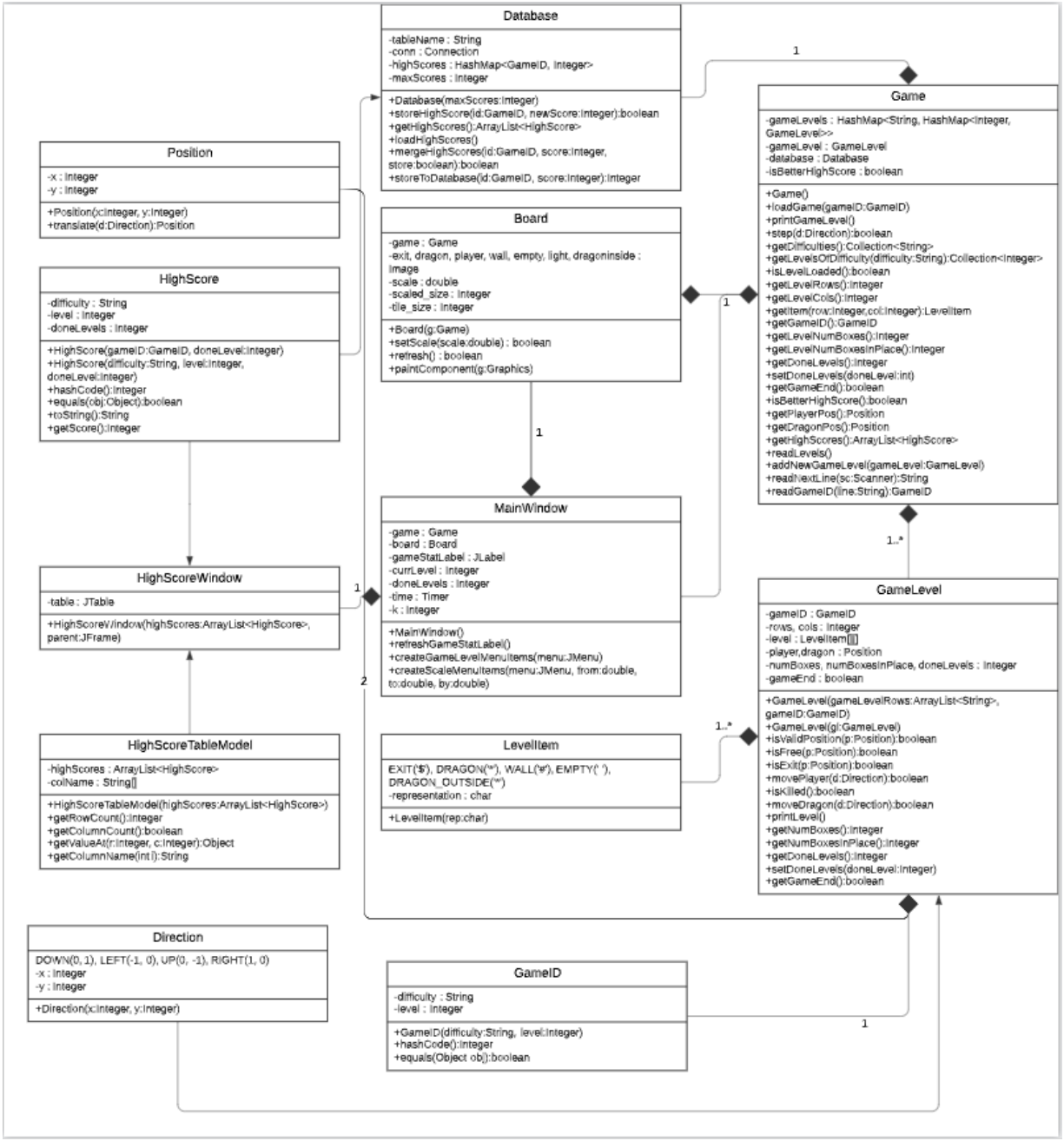
Szükségünk lesz továbbá egy Direction enumra

Szükségünk lesz továbbá egy GameID osztályra

Szükségünk lesz továbbá egy Position osztályra,pozíciót adja vissza

Szükségünk lesz továbbá a HighScore, Database osztályra

[1] A szöveges magyarázatra a hallgatói beadandók dokumentumaiban nincs szükség.



Tesztelési terv

Falnak ütközés

Kijátszás

Meghalás