

HÁZI FELADAT

Programozás alapjai 3.

Végeleges

Filmtár

Szaszkó Szabolcs

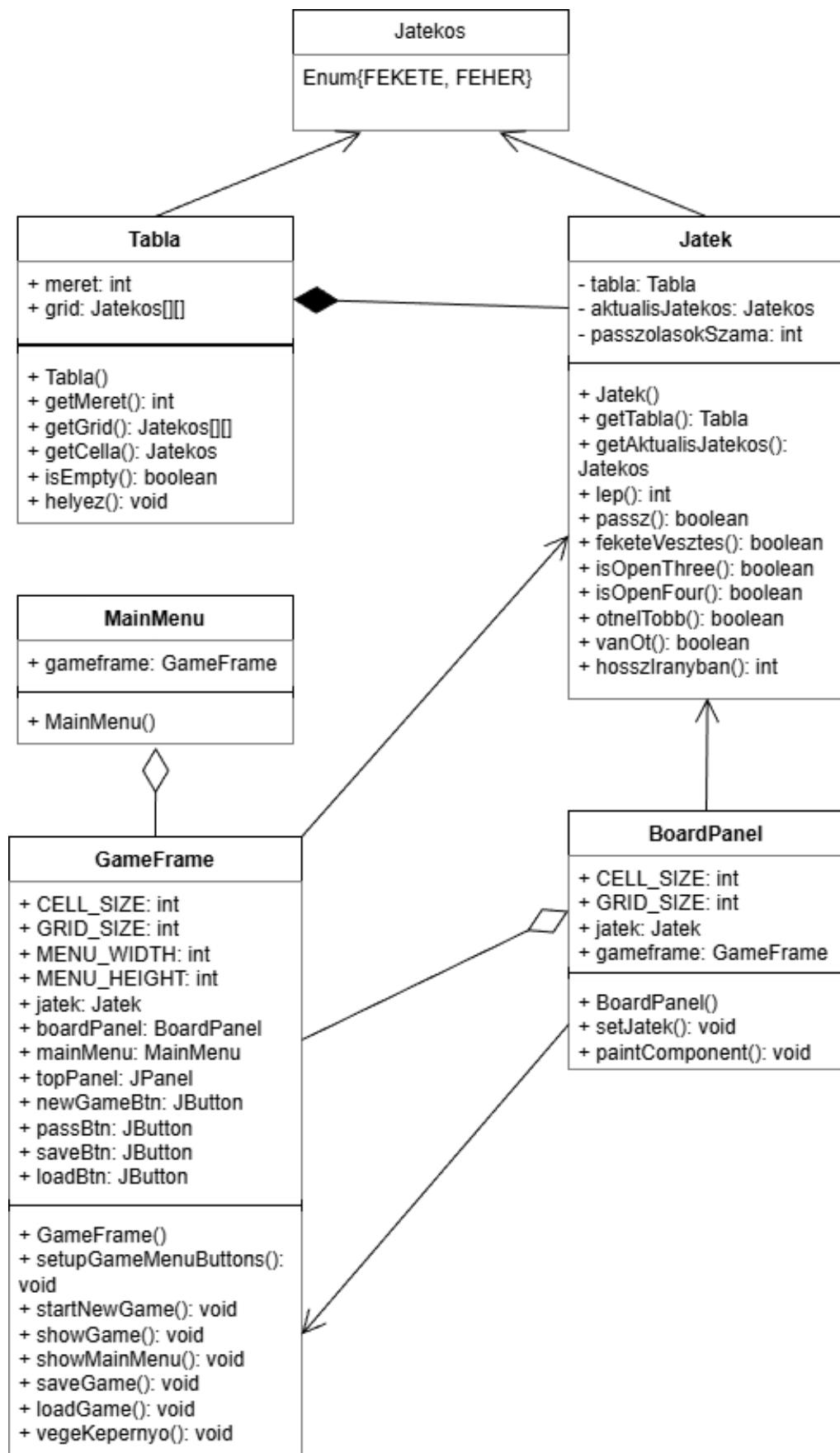
UWE3EQ

2025

Felhasználói segédlet:

A program elindítását követően, a felhasználó eldöntheti, hogy mit szeretne csinálni. Ehhez a főmenün a megfelelő gombokra kell kattintania. Az **Indítás** gomb megjeleníti a táblát és eltudunk kezdeni játszani. A **Betöltés** gomb megjeleníti a fájlkezelőt, ahol ki tudunk választani egy mentést, amit be szeretnénk tölteni. A **Kilépés** csak szimplán bezárja a játékot. Miután megjelenik a tábla, felváltva lehet lépkedni (először fekete, majd fehér), ehhez a rácsok közötti területre kell kattintani. Megjelent néhány új gomb is a játék tábla felett. Az **Új játék** letörli a pályát és egy újat lehet kezdeni. **Passz** ezzel az a játékos tud **passzolni**, akin éppen a sor van (ezzel nem kell lépnie, és egyből a következő játékos jön). **Mentés** ezzel szintén megjelenik a fájlkezelő, ahol eltudjuk menteni a játék állásunkat, és bárhogyan el is tudjuk nevezni. A **Betöltés** gomb ugyan azt csinálja, mint a főmenüben lévő. Miután valamelyik játékos nyert akkor kiírja, hogy melyik játékos nyert, és megjelenik újabb három gomb. Új játék, Főmenü, Kilépés. Itt csak a **Főmenü** lesz új gomb, ezzel egyből visszatudunk lépni a főmenübe, ahol elindítottuk a játékot.

UML:



Programozói dokumentáció:

Jatekos.java

Az enum az amőba játékban részt vevő játékosokat reprezentálja. Az enum két állapotot definiál, amelyek a játék során lehelyezhető bábuk színét jelölik. Ennek segítségével a tábla cellái egyértelműen megkülönböztethetik, hogy melyik játékos lépett az adott pozícióba.

Tabla.java

A Tabla osztály reprezentálja a játéktáblát. A tábla egy kétdimenziós rácsból áll, ahol minden cella egy Jatekos értéket tartalmazhatja (ha nem tartalmazza akkor null). A cellák a játékosok által lerakott bábukat jelölik.

A tábla mérete fix, 15×15 mező.

A konstruktor, létrehozza a 15x15-ös táblát és minden mezőt null értékre állít be. A `getMeret()` visszaadja a méretet. A `getGrid()` visszadja a teljes rácst. A `getCella()` visszaadja egy adott cellának az értékét, hogy milyen játékos áll benne, vagy hogy nem áll benne semmi. `isEmpty()` megadja, hogy egy adott cella üres-e. A `helyez()` függvény lehelyez egy bábut egy adott cellára.

Jatek.java

A Jatek osztály a játék szabályrendszerét valósítja meg. Feladata a tábla állapotának kezelése, a játékosok közti váltás, a lépések érvényességének vizsgálata, valamint a különböző győzelmi és veszteségi feltételek ellenőrzése. A megvalósítás támogatja a döntetlent, a passzolás kezelését, továbbá a speciális szabályokat, amelyek a fekete játékosra vonatkozó tiltott formációkat (nyitott hármasok, négyesek) figyelik.

A konstruktor létrehoz egy új táblát, és beállítja a kezdőjátékost, ami a fekete játékos. A `getTabla()` visszaadja az aktuális táblát. A `getAktualisJatekos()` lekérdezi, hogy melyik játékos lép jelenleg.

A `lep()` függvény már egy kicsit bonyolultabb. Meg kell adni egy sort és oszlopot (vagyis cellát), ahová lépni szeretnénk. Először megvizsgálja, hogy üres-e, tehát lehet oda lépni, ha üres akkor 0-s értéket ad vissza. Ha nem üres akkor a passzolások számát először 0-ra kell állítani, ha ezt nem tennénk, akkor passz után egy rendes lépés után, sose nullázódna. Lehelyezünk egy bábút a kiválasztott cellára és ezután megvizsgáljuk, hogy nyert-e valaki. Ha a fekete játékos szabálytalan lépést lépett akkor 2-es értéket adunk vissza (ezt a `feketeVesztes()` függvénytel ellenőrizzük). Vizsgáljuk, hogy az adott játékos 5-nél több bábút helyezett-e le egymás mellé, ha igen akkor 3-mas értékkel tér vissza, ha 5 db van akkor 4-es értékkel tér vissza. Ha simán lefutott és senki nem nyert/veszített akkor 1-es értékkel tér vissza.

A `passz()` függvény növeli a passzolások számát, megnézi, hogy a passzolások száma 2 vagy több, ha igen akkor true értékkel tér vissza, ha nem akkor false és átállítja az aktuális játékost fehérre.

A `feketeVesztes()` azt vizsgálja, hogy van-e 2 vagy több nyitott hármas és nyitott négyes (a

fekete játékosnál). Először keres egy cellát, ahol fekete bábú van majd elkezdi nézni minden a négy irányba a feltétleket más függvények segítségével (`isOpenThree`, `isOpenFour`).

`isOpenThree()`-nek meg kell adni egy cellát, hogy melyik játékost vizsgálja, és hogy melyik irányba kereshet (ha a `dx = 0` akkor azon az értéken nem fog változtatni, ha `1` akkor jobbra fog lépni, ha `-1` akkor balra, és a `dy` hasonló képpen működik). A függvény megvizsgálja, hogy az adott cellától balra null érték van-e, 3-al odébb is null érték van-e, és hogy ezek között csak fekete van-e, ha nem akkor visszatérünk, hogy nem nyitott hármas. Azért így vizsgálja, mert az egész táblán igazából végig megy (a feketeVesztes függvény miatt), és hogy ne számoljon 1 db nyitott hármat 3-nak így az első kettő feltételre szükség van, mert ezek csak akkor lesznek igazak, ha egy adott irányba az első-től vizsgáljuk.

`isOpenFour()` nagyon hasonlóan működik, mint az `isOpenThree()`.

`otnelTobb()` azt vizsgálja, hogy 5-nél több bábú van-e egy irányba. Ehhez a `hosszIranyba()` függvényt hívja meg.

`vanOt()` azt nézi, hogy 5 db bábú van-e egy irányba (csak öt lehet).

`hosszIranyban()` egy adott irányba (hasonlóan az `isOpenThree()` függvényhez) megszámolja, hogy hány darab azonos bábú van, majd visszaadja, hogy hány darabot számolt meg.