Összetett alkalmazás

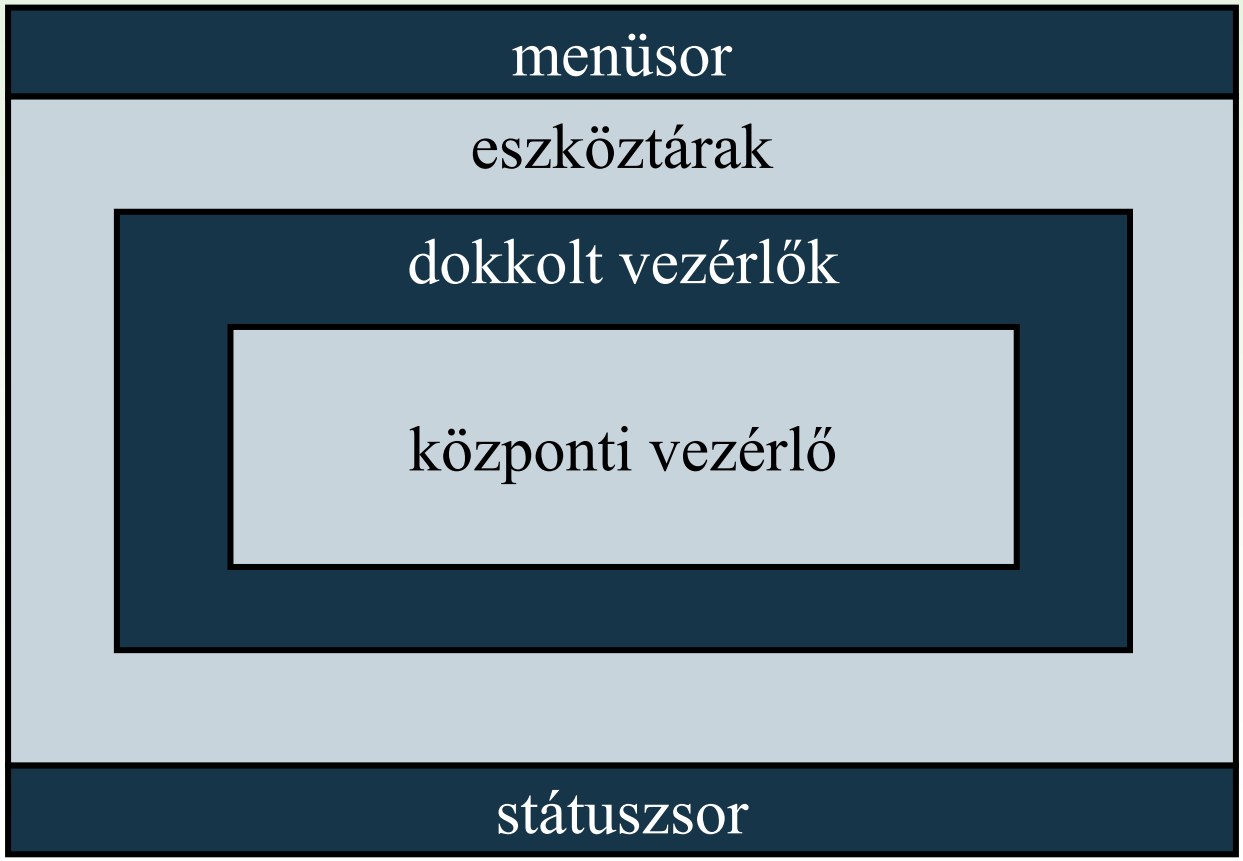
Ablakok

- A grafikus felületű alkalmazásokban a vezérlőket ablakokra helyezzük
  - ablaknak minősül bármely vezérlő, amely egy **`QWidget`**, vagy bármely leszármazottjának példánya, és nincs szülője
  - adottak speciális ablaktípusok is, pl.:
    - *üzenőablak* (**`QMessageBox`**), elsősorban üzenetek közlésére, vagy kérdések feltételére
    - *dialógusablak* (**`QDialog`**), amelynek eredménye van, elfogadható (**`accept`**), vagy elutasítható (**`reject`**)
    - *főablak* (**`QMainWindow`**), amely számos kiegészítést biztosít összetett ablakok megvalósítására

Főablak

- A *főablak* (**`QMainWindow`**) egy olyan speciális ablaktípus, amely megkönnyíti összetett, speciális vezérlőket tartalmazó ablakok létrehozását, úgymint
  - *menüsor* (*Menu Bar*): menüpontok gyűjteménye az ablak tetején
  - *státuszsor* (*Status Bar*): állapotkijelző sor az ablak alján
  - *eszköztár* (*Toolbar*): ikongyűjteményeket tartalmazó funkciógombok, amely az ablak bármely szélére elhelyezhetők
- Az ablakon belül további vezérlőket helyezhetünk el, amelyeket dokkolhatunk az ablak széléhez, vagy középre

Főablak


menüsor
eszköztárak
dokkolt vezérlők
központi vezérlő
státuszsor

Akciók

- A főablak különböző vezérlői (menük, ikonok, gyorsbillentyűk) sokszor ugyanazon funkciókat biztosítják
- A funkciókat egységesen *akció*ként (**`QAction`**) kezelhetjük, amely
  - hozzárendelhető egy tetszőleges menüponthoz, felhelyezhető az eszköztárra, összeköthető egy gyorsbillentyűvel, ezáltal
  - aktiválható közvetlenül egy gyorsbillentyűvel, az eszköztár megfelelő ikonjának kiválasztásával, vagy a kapcsolódó menüpont kijelölésével (egérkattintással, vagy a menüpontot azonosító karakter leütésével, vagy a menüpontra állva az <enter>-rel). Mindegyik esetben egy speciális szignál (**`triggered`**) váltódik ki, amelyhez tetszőleges eseménykezelőt rendelhetünk.
  - rendelkezik felirattal (**`text`**), ikonnal (**`icon`**), gyorsbillentyűvel (**`shortcut`**), segédüzenettel (**`statusTip`**)

Példa

```
// ikon és név megadása, a 'j' billentyűre gyorsnavigál a menüben:
    QAction newAct = new QAction(QIcon("new.png"), tr("Ú&j"), this);

// a keretrendszer által kirendelt "új" billentyűkombináció:
    newAct->setShortcuts(QKeySequence::New);
        // QKeySequence(Qt::CTRL + Qt::Key_N)

    newAct->setStatusTip(tr("Új fájl létrehozása"));

// eseménykezelő társítás:
    connect(newAct, SIGNAL(triggered()),this, SLOT(newFile()));
```

Menü

- A menüt (**QMenu**) a főablak **menuBar** tulajdonságán keresztül kezelhetjük, a menühöz felvehetünk almenüket, akciókat és elválasztókat (**separator**).
  - A menük tetszőlegesen egymásba ágyazhatók.

```
QMenu fileMenu = this->menuBar()->addMenu(tr("&Fájl"));
    // új almenü létrehozása
fileMenu->addAction(newAct);
    // menüpont felvétele
fileMenu->addSpearator();
    // elválasztó
fileMenu->addMenu(tr("&Legutóbbi fájlok"));
    // beágyazott almenü
```

Eszköztár

- Eszköztárakból (**`QToolBar`**) tetszőlegesen sokat vehetünk fel, amelyek alapértelmezetten az ablak tetején jelennek meg.
  - Ikonok sorozatát adják, esetleges elválasztókkal szeparálva.
  - Az eszköztárak alapértelmezés szerint utólag áthelyezhetőek bármely szélére az ablaknak, illetve lehetnek lebegő (**`floating`**) állapotban is.

```
QToolBar fileBar = this->addToolBar(tr("Fájl"));
    // új eszköztár felvétele
fileBar->addAction(newAct);
    // új akció felvétele
fileBar->addSpearator();
    // elválasztó
```

Státuszsor

- A státuszsor (**`QStatusBar`**) alapvetően státuszüzenetek kiírására szolgál, ugyanakkor bármilyen vezérlő ráhelyezhető
  - üzenetet kiírni a **`showMessage(`*`<üzenet>`*`)`** utasítással tudunk, törölni a **`clearMessage()`** utasítással
  - pl.: **`this->statusBar()->showMessage(tr("Kész"));`**

Főablak tartalma

- Az ablak területére célszerű egy külön vezérlőben elhelyezni a tartalmat, ez a központi vezérlő (**`centralWidget`**)
- Amennyiben több tartalmat helyeznénk az ablakra, lehetőségünk van azokat dokkolni a **`QDockWidget`** osztály segítségével, amelyet az **`addDockWidget(<vezérlő>)`** művelettel helyezhetünk az ablakra

Alkalmazásszintű tulajdonságok

- A Qt alkalmazásokat minden esetben egy alkalmazás (**`QApplication`**) objektum vezérli, amely számos értéket tárol, úgymint:
  - alkalmazás információk (**`applicationName`**, **`orgnizationName`**, **`applicationVersion`**),
  - környezeti információk (**`applicationDirPath`**, **`arguments`**, **`keyboardModifiers`**, **`clipboard`**),
  - grafikus környezeti adatok (**`allWindows`**, **`windowIcon`**, **`palette`**, **`styleSheet`**, **`font`**).
- Az alkalmazás értékeihez bárhonnan, statikus műveletekkel hozzáférhetünk.

Példa

```
QApplication::setOrganizationName("MySoft");
QApplication::setApplicationName("MyApp");
   // beállítunk némi információt
…
QString executableName = QApplication::arguments()[0];
   // lekérjük a programnevet
if (QApplication::arguments().size() > 1){
   // ha még van ezen felül argumentum
   QString arg1 = QApplication::arguments()[1];
}
```

Beállítások kezelése

- Nagyobb alkalmazások olyan alkalmazásszintű beállításokkal rendelkeznek, amelyek elmenthetők, és újabb futtatáskor betölthetők.
- A beállítások eltárolhatók egyedileg is, de használhatjuk a beépített **`QSettings`** osztályt, amely egyszerűsíti a beállítások kezelését.
  - A beállítások eltárolásának módja platformonként változik (Linux esetén konfigurációs fájlok, Windows esetén regisztrációs adatbázis), ezt az osztály elfedi, így a programozónak a tárolás módjával nem kell törődnie.
  - A beállítások egy adott alkalmazásra és felhasználóra vonatkoznak.
  - A beállításokba kulcs/érték párokat vehetünk fel a **`setValue(<kulcs>,<érték>)`** utasítással, ahol a kulcs szöveges, az érték tetszőleges **`QVariant`** lehet.
    - a **`value(<kulcs>)`** utasítás lekérdez
    - a **`contains(<kulcs>)`** függvény ellenőrzi a kulcs létezését

A **`QVariant`** egy általános típus, amely a primitív típusokat tudja „becsomagolni”, és rendelkezik konverziós műveletekkel (toInt(), value<típus>()).

Erőforrások

- Az alkalmazáshoz használt ikonokat (pl. a főablakon használt akciók ikonjait) és egyéb nem kód tartalmat (pl. adatállományokat) az alkalmazáshoz *erőforrás*ként (*resource*) lehet csatolni.
  - Az erőforrások tartalma belefordul a futtatandó állományba, így nem kell külön másolni őket.
  - Az erőforrásokat a projekthez tartozó **`.qrc`** fájlban nevezzük meg.
  - Az erőforrásként megadott fájlokat a **`:<elérési útvonal>`** hivatkozással hívhatjuk be: `QIcon(":/images/new.png")`

Feladat

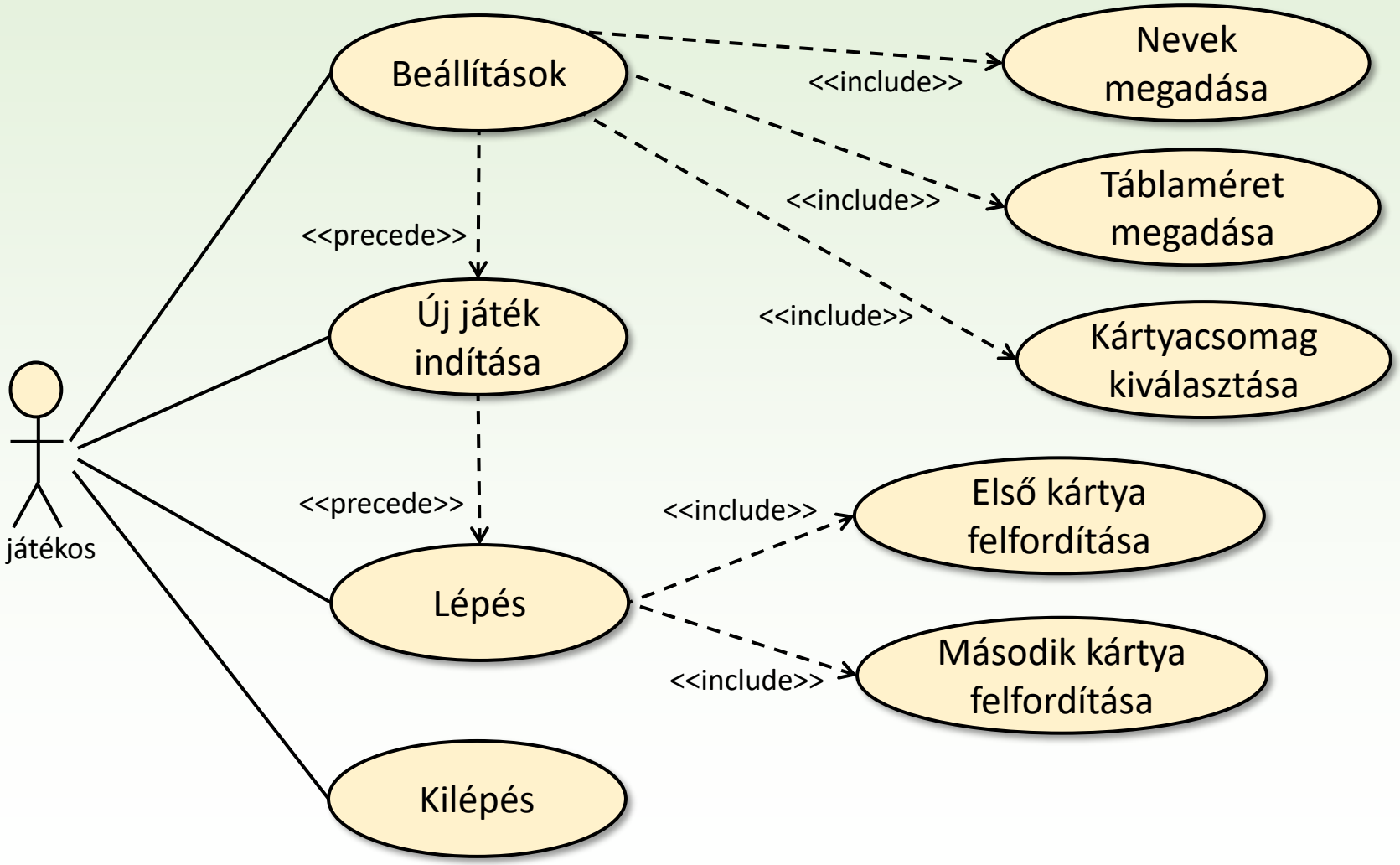
Készítsünk egy *Memory* kártyajátékot, amelyben két játékos küzd egymás ellen. A játékmezőn kártyapárok találhatóak, és a játékosok feladata ezek megtalálása.

- A játékot különböző kártyacsomagokkal lehessen játszani, amelyek előzetesen feltöltött könyvtárakból legyenek betölthetők. Minden ilyen könyvtárban legyen egy **`name.txt`**, ami a csomag nevét tartalmazza, emellett tetszőleges számú kép (ezek a kártyák), valamint a kártyák hátlapjának képe (**`back`** fájlnévvel).
- Legyen lehetőségünk kiválasztani egy kártyacsomagot, megadni a játéktábla méretét (amely csak páros számú, de legalább 4 kártyából álljon), valamint a játékosok neveit.

Feladat folytatása

- Kezdetben minden kártya le van fordítva, a játékosok felváltva lépnek, minden lépésben felfordíthatnak két kártyát.
- Amennyiben a kártyák egyeznek, úgy felfordítva maradnak és a játékos ismét léphet, különben 1 másodperc múlva visszafordulnak, és a másik játékos következik.
- A játékot az nyeri, aki több kártyapárt talált meg.
- Megnyert játékok számát göngyölítve jelenítjük meg, amíg új játékosokat nem állítunk be.
- A felületen folyamatosan megjelenítjük a játékosok adatait (sikeres, sikertelen lépések száma, megnyert játszmák száma).

Elemzés


Beállítások
Nevek megadása
<<include>>
Táblaméret megadása
<<include>>
Kártyacsomag kiválasztása
<<include>>
<<precede>>
Új játék indítása
<<precede>>
Első kártya felfordítása
<<include>>
Lépés
Második kártya felfordítása
<<include>>
Kilépés
játékos

Modell adatai

- Kártyacsomag (név, kártyaképek sorozata, közös hátlap képe)
- Játéktábla
  - kártyaképek kártyacsomagbeli sorszámait (indexeit) duplán tartalmazó, de véletlen módon összekevert sorozat
  - a már felfedett kártyapárok megjelölése
  - a már felfedett kártyapárok darabszáma
  - a játéktábla sor és oszlopszáma
- Két játékos: (név, az adott játékban sikeres és sikertelen próbálkozásainak száma, az eddig elért győzelmeinek száma
- A játék fontos tulajdonságai
  - a soron következő játékos sorszáma
  - az elsőként megfordított kártya sorszáma
  - a másodikként megfordított kártya sorszáma
- Időzítő, hogy a két felfordított, de eltérő kártyakép rövid ideig látszódjon.

MemoryGame
- cardPack : CardPack - cardIndexes : int[ ] - cardFound : bool[ ] - foundCardsNumber : int - currentNumCols : int - currentNumRows : int - players : Player[2] - currentPlayerIndex : int - firstCardIndex : int - secondCardIndex : int - timer : QTimer
…

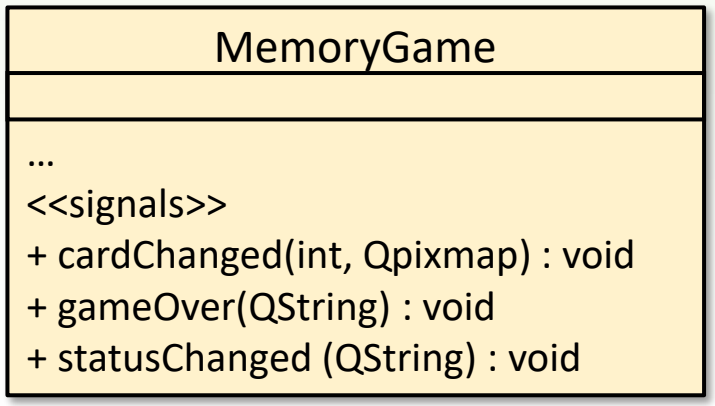
Modell tevékenységei

- Beállítások
  - játékosok neve: *setPlayers(Qstring, QString)*
  - kártyacsomag: *loadPack(CardPack)*
- Új játék indítása: *newGame(int, int, bool)*
  - megkapja a játék tábla méreteit
  - megkapja, hogy új meccs is indul-e (játékos változáskor automatikus)
  - kártyák megkeverése *shuffleCard()*
  - kártyák közül egy sincs még felfedve
- Lépés *selectCard(int)*
  - Elsőként, vagy másodikként megjelölt kártya indexe
- Időzítő eseménykezelése *hideCards()*
  - Ha a két felfordított kártyakép eltérő, akkor azok rövid idő leteltével újra a hátlapjukat mutatják.

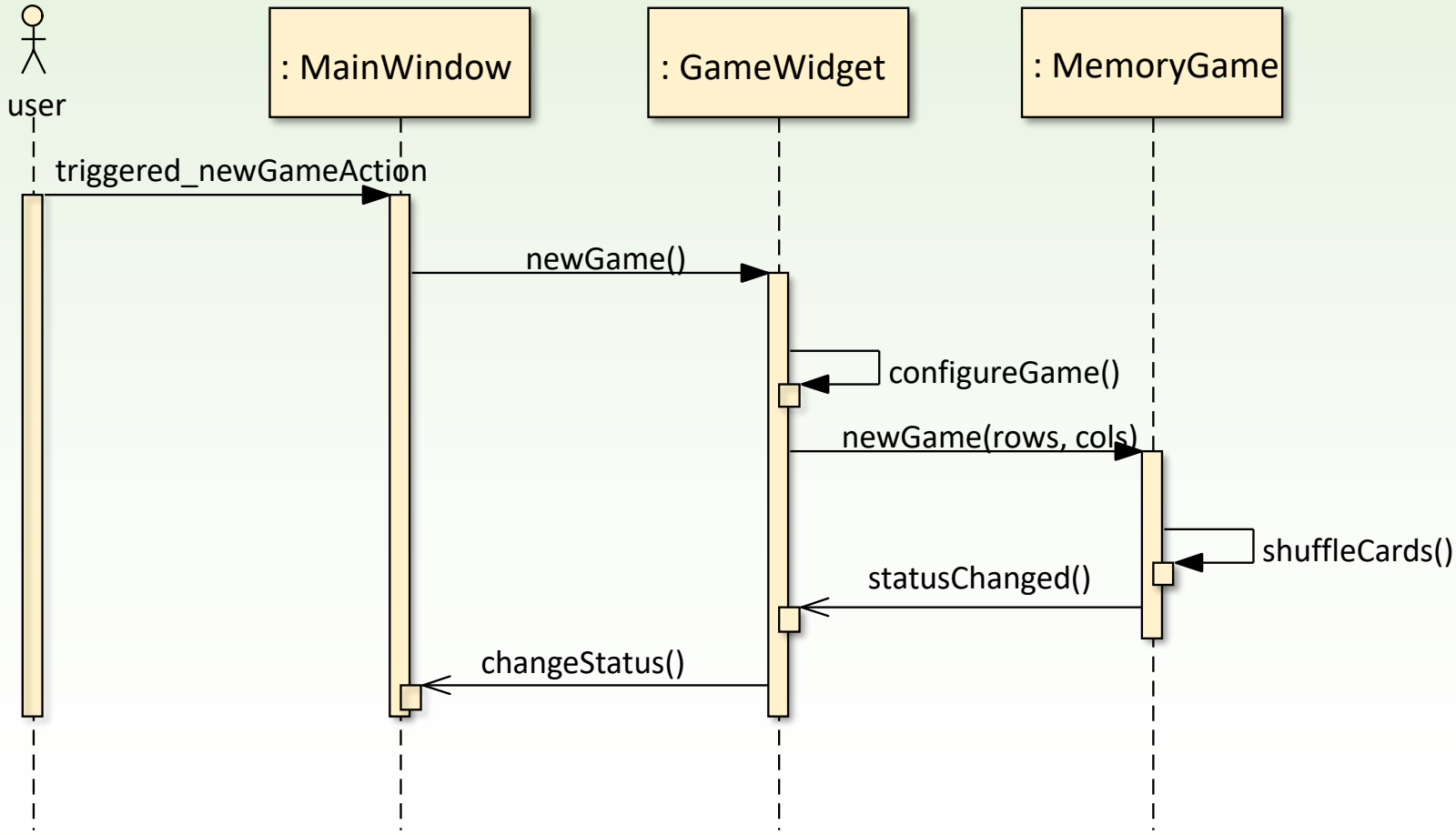
MemoryGame
…
+ setPlayers(Qstring, QString) : void + loadPack(CardPack) : void + newGame(int, int, bool) : void - shuffleCard() : void + selectCard(int) : void <<slots>> - hideCards() : void

Vezérlés

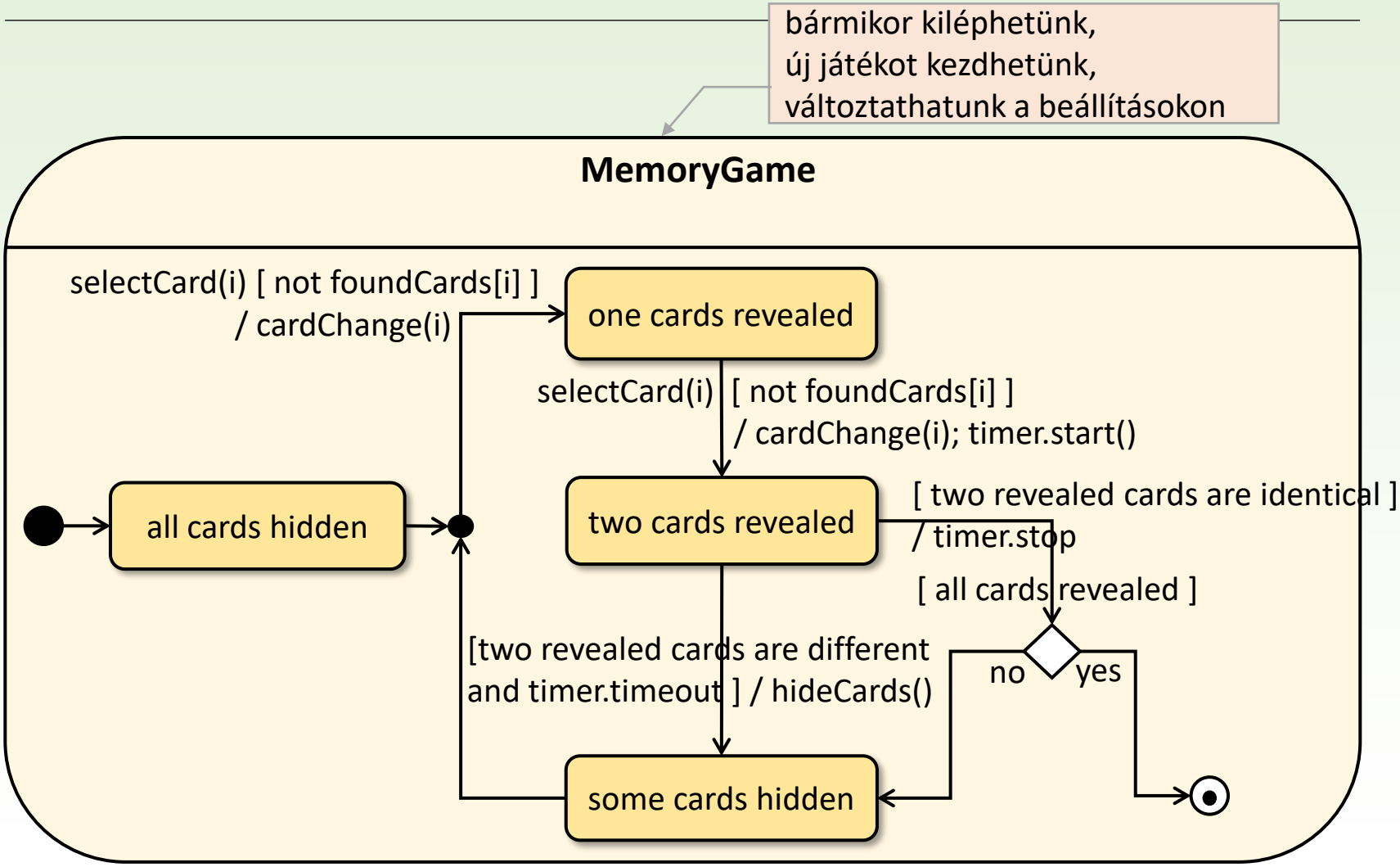
- A modell szignálok segítségével vezérli a nézetet
  - Amikor egy játékos egy lépésben egy kártyát kiválaszt, akkor annak képét a modell azonosítja be, majd utasítja a nézetet ennek a képnek a megmutatására: **`cardChanged`**
  - Amennyiben minden kártyapár felfedésre került, ezt jelezzük a nézetnek: **`gameOver`**
  - Minden lépésben frissíteni kell a játékosokról kiírt statisztikát: **`statusChanged()`**



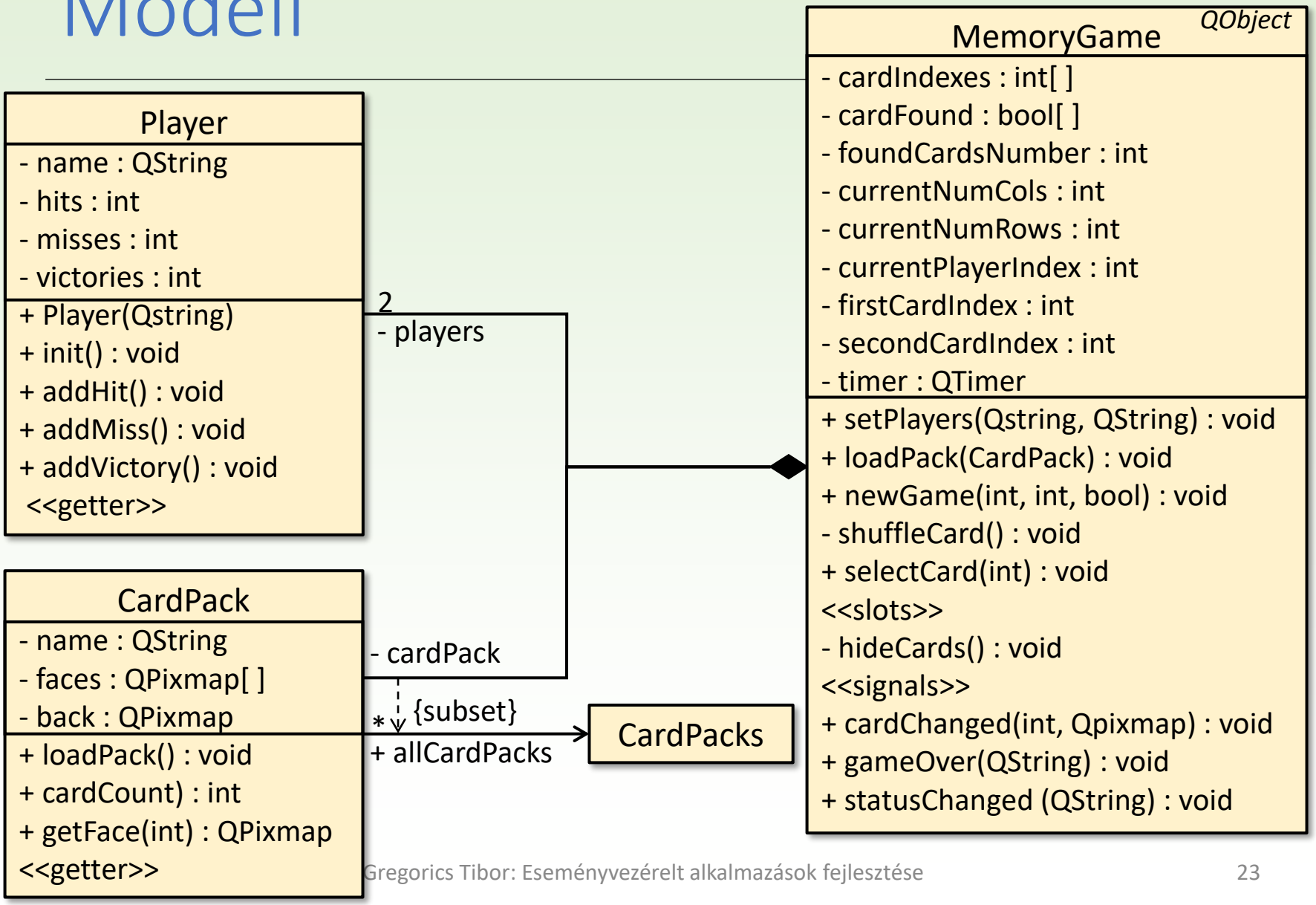
Új játék indítása



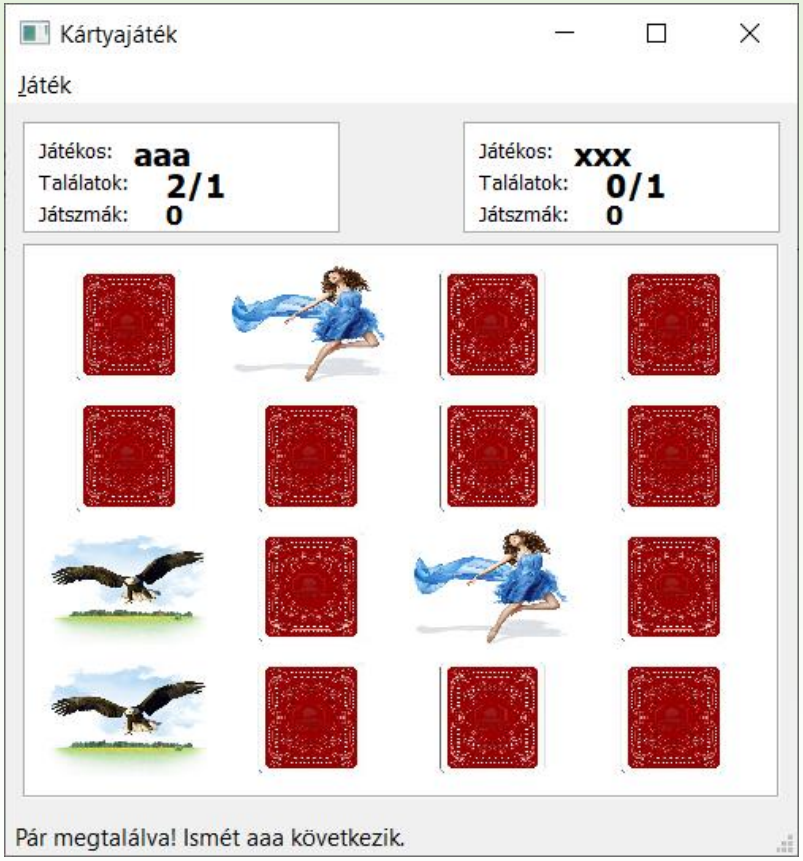
A játék egy lépése



Modell


MemoryGame
QObject
- cardIndexes : int[ ]
- cardFound : bool[ ]
- foundCardsNumber : int
- currentNumCols : int
- currentNumRows : int
- currentPlayerIndex : int
- firstCardIndex : int
- secondCardIndex : int
- timer : QTimer
+ setPlayers(Qstring, QString) : void
+ loadPack(CardPack) : void
+ newGame(int, int, bool) : void
- shuffleCard() : void
+ selectCard(int) : void
<<slots>>
- hideCards() : void
<<signals>>
+ cardChanged(int, Qpixmap) : void
+ gameOver(QString) : void
+ statusChanged (QString) : void
Player
- name : QString
- hits : int
- misses : int
- victories : int
+ Player(Qstring)
+ init() : void
+ addHit() : void
+ addMiss() : void
+ addVictory() : void
<<getter>>
2
- players
CardPack
- name : QString
- faces : QPixmap[ ]
- back : QPixmap
+ loadPack() : void
+ cardCount) : int
+ getFace(int) : QPixmap
<<getter>>
- cardPack
{subset}
*
+ allCardPacks
CardPacks

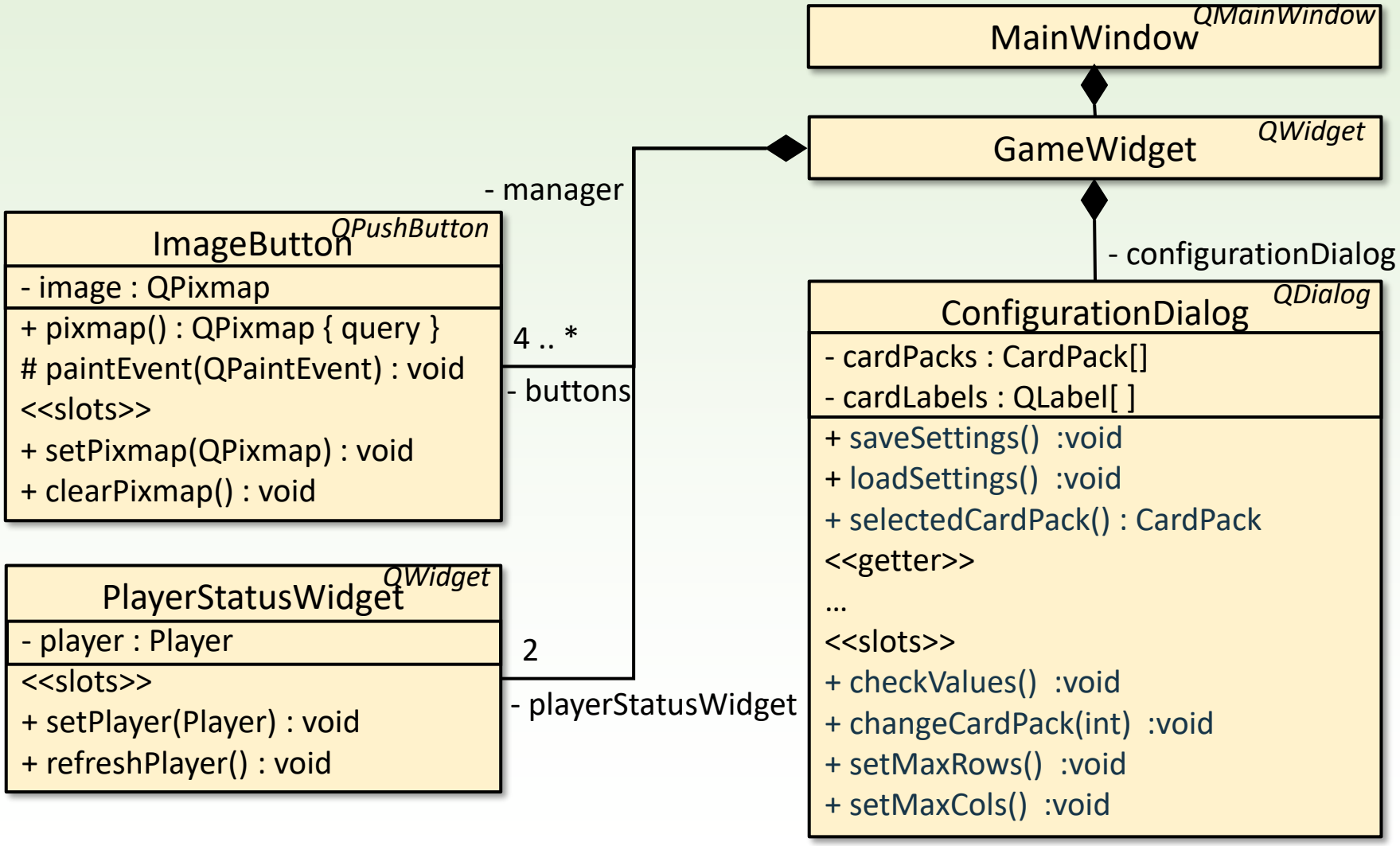
Felület terve



Nézet elemei

- A nézet tartalmazza:
  - a főablakot (**`MainWindow`**) menüvel és státuszsorral,
  - a beállításokat végző segédablakot (**`ConfigurationDialog`**),
  - a játéktáblát megjelenítő vezérlőt (**`GameWidget`**), amely tartalmazza a játékmezővel kapcsolatos tevékenységeket,
  - a játékos információkat kiíró vezérlőt (**`PlayerStatusWidget`**, ezt előléptetett vezérlővel állítjuk be a felülettervezőben),
  - valamint a képet megjeleníteni tudó egyedi gombokat (**`ImageButton`**).
- Egy csomag kártyát erőforrásként csatolunk az alkalmazáshoz (**`packs.qrc`**), hogy mindig legyen legalább egy ilyen.

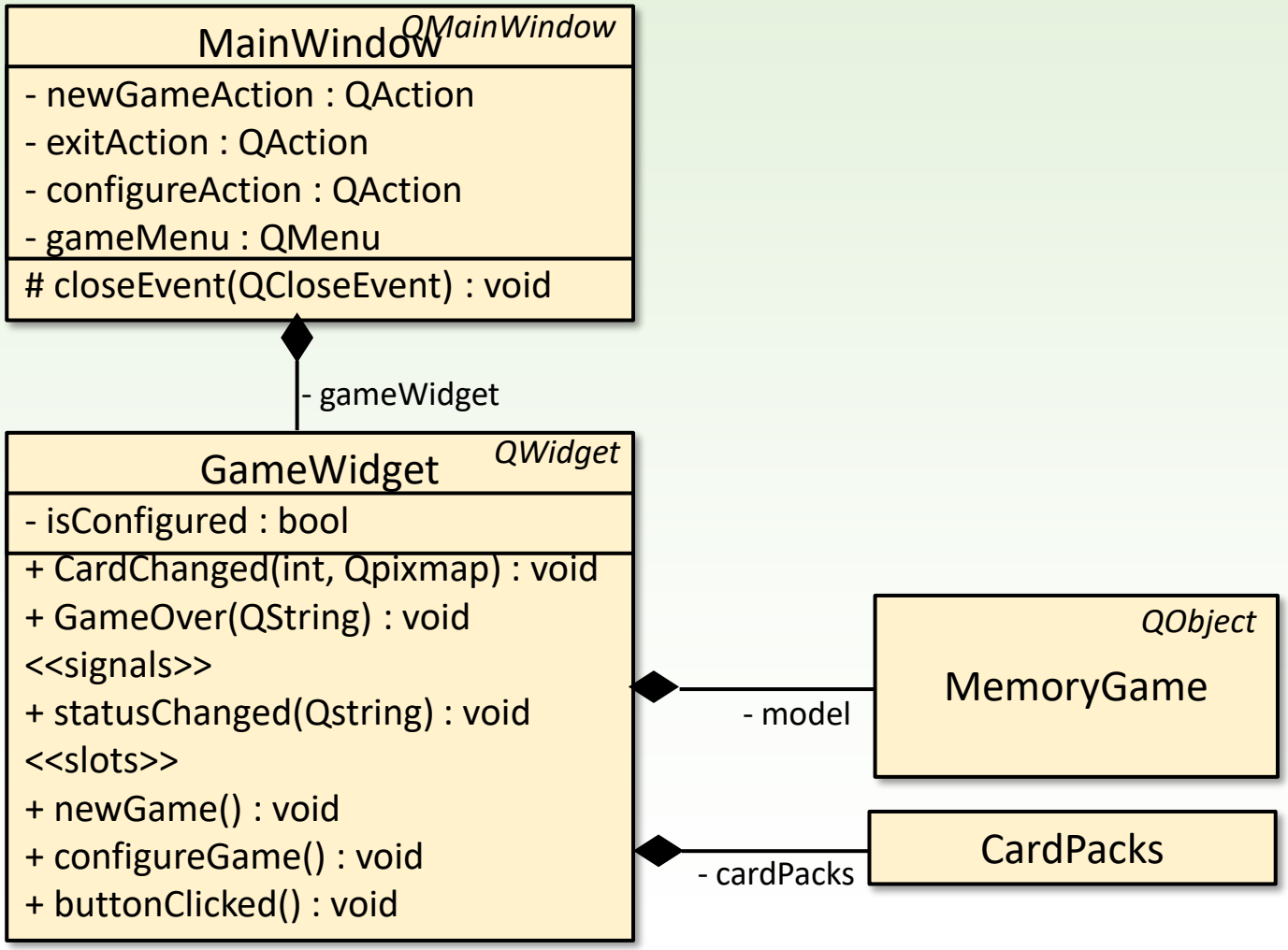
Nézet osztálydiaramja



Nézet tevékenységei

- MainWindow
  - A játék beállítását a főablakban kell kiváltanunk a megfelelő akció (**`configureAction`**) segítségével. Ennek hatására fut le **`GameWidget configureGame`** metódusa, amelyik megnyitja a beállításokat végző dialógus ablakot, majd beállítja a modellt.
  - Új játékot a főablak **`newGameAction`** akciójának triggerelésével indíthatunk. Ez hívja a **`GameWidget newGame()`** metódusát, amely többek között meghívja a modell hasonló nevű metódusát.
  - A harmadik akció (**`exitAction`**) az alkalmazás leállásához vezet.
- GameWidget
  - Feldolgozza a játéktáblán elhelyezett nyomógombokra történt kattintásokat, és hívja a modell **`selectCard()`** metódusát.
  - Reagál a modell szignáljaira.

Modell-Nézet kapcsolat



Megvalósítás: eseménykezelés

```
MainWindow::MainWindow()
{
    …
    connect(_newGameAction, SIGNAL(triggered()),
        _gameWidget, SLOT(newGame()));
    connect(_configureAction, SIGNAL(triggered()),
        _gameWidget, SLOT(configureGame()));
    connect(_exitAction, SIGNAL(triggered()),
        this, SLOT(close()));

    connect(_gameWidget, SIGNAL(statusChanged(QString)),
        this->statusBar(), SLOT(showMessage(QString)));
}
```

Megvalósítás: eseménykezelés

```
GameWidget::GameWidget(QWidget *parent) :
    QWidget(parent), _ui(new Ui::GameWidget)
{
    _ui->setupUi(this);
    _manager = new GameManager();
    _configurationDialog = 0;
    _isConfigured = false; // kezdetben nincs konfigurálva a játék

    connect(_manager, SIGNAL(statusChanged(QString)),
        this, SIGNAL(statusChanged(QString)));
    connect(_manager, SIGNAL(statusChanged(QString)),
        _ui->firstPlayerStatus, SLOT(refreshPlayer()));
    connect(_manager, SIGNAL(statusChanged(QString)),
        _ui->secondPlayerStatus, SLOT(refreshPlayer()));
    connect(_manager, SIGNAL(cardChanged(int,QPixmap)),
        this, SLOT(gameManager_CardChanged(int, QPixmap)));
    connect(_manager, SIGNAL(gameOver(QString)),
        this, SLOT(gameManager_GameOver(QString)));
}
```

a logikai réteg szignálja egy újabb szignált vált ki

Megvalósítás: eseménykezelés

```
ConfigurationDialog(QVector<CardPack*> cps, QWidget *parent) :
    QDialog(parent), _ui(new Ui::ConfigurationDialog), _cardPacks(cps)
{
    _ui->setupUi(this);
    …
    foreach(CardPack* pack, _cardPacks)
        _ui->comboCardPack->addItem(pack->getName());
    connect(_ui->comboCardPack, SIGNAL(currentIndexChanged(int)),
        this, SLOT(changeCardPack(int)));
    connect(_ui->spinRows, SIGNAL(valueChanged(int)), this, SLOT(setMaxCols()));
    connect(_ui->spinCols, SIGNAL(valueChanged(int)), this, SLOT(setMaxRows()));
    connect(_ui->buttonOk, SIGNAL(clicked()), this, SLOT(checkValues()));
    connect(_ui->buttonCancel, SIGNAL(clicked()), this, SLOT(reject()));
    if (_cardPacks.size() > 0){
        _ui->comboCardPack->setCurrentIndex(0);
        changeCardPack(0);
    }
    loadSettings();
}
```

Csomag diagram