Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások fejlesztése I

2. előadás

Egyszerű, egyablakos alkalmazások

Giachetta Roberto

A jegyzet az ELTE Informatikai Karának
2014. évi Jegyzetpályázatának támogatásával készült

Egyszerű, egyablakos alkalmazások

A grafikus felület

- A grafikus felhasználói felület ablakokból tevődik össze, amelyeken vezérlőket helyezünk el
  - a vezérlők objektumorientáltan valósulnak meg, öröklődés segítségével szerveződnek hierarchiába
  - minden vezérlő ősosztálya a **`QWidget`**, amelynek van egy további ősosztálya, a **`QObject`**
- A **`QObject`** azon típusok őse, amely kihasználja a Qt speciális vonásait, úgymint *események* és *eseménykezelők*, *tulajdonságok*, *időzítés*
  - a **`QObject`** példányok nem másolhatóak, ezért jórész mutatók és referenciák segítségével kezeljük őket

Egyszerű, egyablakos alkalmazások

Vezérlők

- A vezérlők számos *tulajdonság*gal rendelkeznek
  - tulajdonságnak nevezzük a vezérlők azon külsőleg elérhető értékeit (mezőit), amelyeket *lekérdező* (*getter*), illetve *beállító* (*setter*) műveletek segítségével szabályozhatunk
  - a lekérdező művelet neve a tulajdonság neve, a beállító művelet tartalmaz egy **`set`** előtagot
  - pl.:

```
QLabel label;
label.setText("Hello World!");
   // beállítjuk a címke szövegét (text)
QString text = label.text();
   // lekérdezzük a címke szövegét
```

Egyszerű, egyablakos alkalmazások

Vezérlők

- A vezérlők fontosabb tulajdonságai:
  - méret (**`size`**), vagy geometria (elhelyezkedés és méret, **`geometry`**)
  - szöveg (**`text`**), betűtípus (**`font`**), stílus (**`styleSheet`**), színpaletta (**`palette`**), előugró szöveg (**`toolTip`**) fókuszáltság (**`focus`**), láthatóság (**`visible`**)
  - engedélyezés (használható-e a vezérlő, **`enabled`**)
- A vezérlőkön (pl. **`QLabel`**, **`QLineEdit`**) elhelyezett szöveg formázható több módon
  - pl. formátummal (**`textFormat`**), illetve használhatóak HTML formázó utasítások is

Egyszerű, egyablakos alkalmazások

Vezérlők

```
#include <QPushButton>
…

int main(int argc, char *argv[]){
   …
   QPushButton myButton; // gomb
   myButton.resize(75, 30); // méret
   myButton.setFont(QFont("Times", 20)); // betűtípus
   myButton.setText("<h1>My Button<h1><br>This is
      my button!"); // formázott szöveg
   myButton.setToolTip("You can try klicking on
      it..."); // előugró szöveg
   myButton.show(); // gomb megjelenítése ablakként
   …
```

Egyszerű, egyablakos alkalmazások

Vezérlők

- A leggyakrabban használt vezérlők:
  - címke (**`QLabel`**)
  - LCD kijelző (**`QLCDNumber`**), folyamatjelző (**`QProgressBar`**)
  - nyomógomb (**`QPushButton`**), kijelölő gomb (**`QCheckBox`**), rádiógomb (**`QRadioButton`**)
  - szövegmező (**`QLineEdit`**), szövegszerkesztő (**`QTextEdit`**)
  - legördülő mező (**`QComboBox`**)
  - dátumszerkesztő (**`QDateEdit`**), időszerkesztő (**`QTimeEdit`**)
  - csoportosító (**`QGroupBox`**), elrendező (**`QLayout`**)
  - menü (**`QMenu`**), eszköztár (**`QToolBox`**)

Egyszerű, egyablakos alkalmazások

Vezérlők hierarchiája

- A grafikus vezérlők között *hierarchiát* állíthatunk fel, amely egy fának megfelelő struktúrával reprezentálható
  - a vezérlőnek lehet *szülője* (**`parent`**), amelyen belül található
  - a vezérlőnek lehetnek *gyerekei* (**`children`**), azon vezérlők, amelyek rajta helyezkednek el
  - amennyiben egy vezérlőt megjelenítünk (**`show()`**), az összes gyerek vezérlője is megjelenik
  - ha egy szülő vezérlőt elrejtünk/megjelenítünk, kikapcsolunk/bekapcsolunk, vagy megsemmisítünk, akkor a gyerekein is megtörténik a tevékenység

Egyszerű, egyablakos alkalmazások

Ablakok

- A grafikus felületű alkalmazásokban a vezérlőket *ablak*okra helyezzük, amely a vezérlő szülője lesz
  - ablaknak minősül bármely vezérlő, amely egy **`QWidget`**, vagy bármely leszármazottjának példánya, és nincs szülője
  - vezérlő szülőjét konstruktor paraméterben, vagy a **`parent`** tulajdonságon keresztül adhatjuk meg
  - pl.:

```
QWidget parentWidget; // ablak
QPushButton childButton(&parentWidget); // gomb
…
parentWidget.show();
   // a gomb is megjelenik az ablakkal
```

Egyszerű, egyablakos alkalmazások

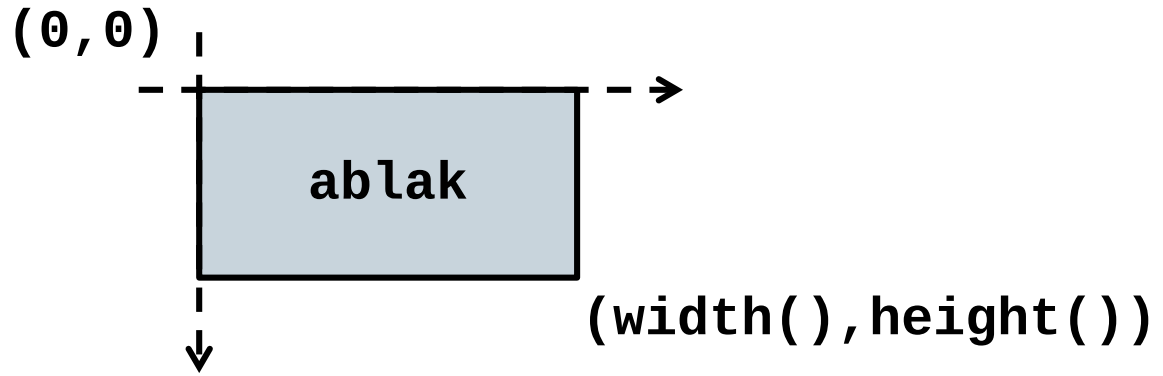
Ablakok

- Az ablakként használt vezérlő további beállításai:
  - cím (**`windowTitle`**), ikon (**`windowIcon`**)
  - állítható teljes/normál képernyőre, vagy a tálcára (**`showMaximized`**, **`showNormal`**, **`showMinimized`**)
  - egyszerre mindig csak egy aktív ablak lehet (**`isActiveWindow`**)
- A vezérlők mérete többféleképpen befolyásolható
  - alapból változtatható méretűek, ekkor külön állítható minimum (**`minimumSize`**), maximum (**`maximumSize`**), valamint az alapértelmezett (**`baseSize`**) méret
  - a méret rögzíthető (**`setFixedSize`**)

Egyszerű, egyablakos alkalmazások

Ablakok

- Amennyiben egy vezérlőt az ablakon helyezünk el, meg kell adnunk a pozícióját és méretét (**`setGeometry(int, int, int, int)`**)
  - az ablak koordinátarendszere a bal felső sarokban indul a (0,0) koordinátával, és balra, illetve lefelé növekszik



  - az ablak területébe nem számoljuk bele az ablak fejlécének területét, amit külön lekérdezhetünk (**`frameGeometry`**)

Egyszerű, egyablakos alkalmazások

Ablakok

```
…
QWidget window; // ablak létrehozása
window.setBaseSize(200, 120); // méretezés
window.setWindowTitle("Demo Window");
   // ablakcímke megadása

QPushButton quit("Quit", &window);
   // gomb az ablakra
quit.setGeometry(10, 40, 180, 40);
   // elhelyezés az ablakon
QObject::connect(&quit, SIGNAL(clicked()),
                 &app, SLOT(quit()));
window.show(); // ablak megjelenítése
…
```

Egyszerű, egyablakos alkalmazások

Speciális ablakok

- Amellett, hogy ablak bármilyen vezérlő lehet, adottak speciális ablaktípusok, pl.:
  - *üzenőablak* (**`QMessageBox`**), elsősorban üzenetek közlésére, vagy kérdések feltételére, pl.:

```
QMessageBox::warning(this, trUtf8("Warning"),
    trUtf8("This is annoying.\nDo something!"));
  // figyelmeztető üzenet
```

  - *dialógusablak* (**`QDialog`**), amelynek eredménye van, elfogadható (**`accept`**), vagy elutasítható (**`reject`**)
  - *főablak* (**`QMainWindow`**), amely számos kiegészítést biztosít összetett ablakok megvalósítására (menü, állapotsor, beágyazott ablakok kezelése)

Egyszerű, egyablakos alkalmazások

Egyedi ablakok

- Célszerű a saját ablakainknak saját osztályt létrehozni
  - magában az osztályban szerkeszthetjük a tulajdonságait, eseménykezelését, nincs szükségünk a főprogramra
  - pl.:

```
class DemoWindow : public QWidget
{
public:
   DemoWindow(QWidget* parent = 0);
      // a konstruktor megkaphatja a szülőt
private:
   QPushButton* quitButton; // gomb az ablakon
};
```

Egyszerű, egyablakos alkalmazások

Egyedi ablakok

```
DemoWindow::DemoWindow(QWidget* parent)
   : QWidget(parent) // ős konstruktor meghívása
{
   setBaseSize(200, 120);
   setWindowTitle("Demo Window");
   quitButton = new QPushButton("Quit", this);
      // gomb az ablakra
   quitButton->setGeometry(10, 40, 180, 40);

   connect(quitButton, SIGNAL(clicked()),
      QApplication::instance(), SLOT(quit()));
      // az eseménykezeléshez lekérdezzük az
      // alkalmazás példányt
}
```

Egyszerű, egyablakos alkalmazások

Példa

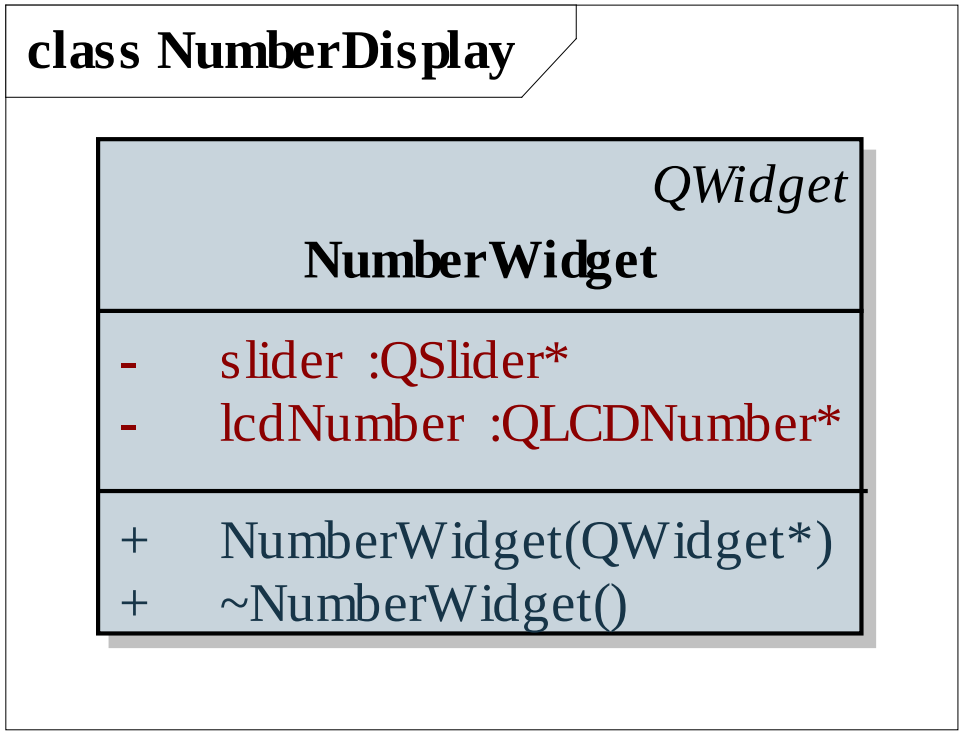
Feladat: Készítsünk egy egyszerű alkalmazást, amelyben egy csúszkával állíthatunk egy digitális kijelzőn megjelenő számot.

- az alkalmazás számára létrehozunk egy új ablak osztályt (**`NumberWidget`**), amelyre felhelyezünk egy csúszkát (**`QSlider`**), valamint egy számkijelzőt (**`QLCDNumber`**)
- összekötjük a csúszka változást jelző eseményét (**`valueChanged(int)`**) a kijelző számbeállító eseménykezelőjével (**`display(int)`**), így egyben paraméterben át is adódik az aktuális érték
- az összekötéseket a konstruktorban megfogalmazhatjuk, így már csak a destruktort kell megvalósítanunk, amely törli a vezérlőket

Egyszerű, egyablakos alkalmazások

Példa

Tervezés:



Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`main.cpp`*):*

```
#include <QtGui/QApplication>
#include "numberwidget.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    NumberWidget w;
    w.show();
        // a főprogram csak példányosítja és
        // megjeleníti az ablakot

    return a.exec();
}
```

Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`numberwidget.cpp`*):*

```
NumberWidget::NumberWidget(QWidget *parent)
    : QWidget(parent) {
    // meghívjuk az ős konstruktorát
    setWindowTitle("Number Display"); // ablakcím
    setFixedSize(300, 175);
       // rögzített méret beállítása
    slider = new QSlider(this);
       // a vezérlő megkapja szülőnek az ablakot
    …
    connect(slider, SIGNAL(valueChanged(int)),
            lcdNumber, SLOT(display(int)));
       // esemény és eseménykezelő összekötése
    …
```

Egyszerű, egyablakos alkalmazások

Egyedi események és eseménykezelők

- A saját osztályainkban lehetőségünk van egyedi események és eseménykezelők létrehozására, továbbá tetszőleges eseményt kiválthatunk
  - az osztályt el kell látni a **`Q_OBJECT`** makróval, és a **`QObject`** osztály leszármazottjának kell lennie
  - eseményeket az **`<eseménynév>(<paraméterek>)`** utasítással válthatunk ki, pl.: **`clicked(false);`**
  - új eseményeket az osztálydefiníció **`signals`** részében helyezhetünk el
  - új eseménykezelőket az osztálydefiníció **`slots`** részében helyezhetünk el, és az eseménykezelőnek adhatunk láthatóságot is

Egyszerű, egyablakos alkalmazások

Egyedi események és eseménykezelők

- az események, illetve eseménykezelők eljárások (**`void`** típussal), tetszőleges paraméterezéssel
- eseményeket csak deklarálnunk kell, az eseménykezelőket definiálni is kell

- Pl.:

```
class Demo : public QObject{
   Q_OBJECT // az osztályban definiálhatunk
            // eseményt és eseménykezelőt
signals: // saját események
   void demoSignal(int param);
public slots: // publikus eseménykezelők
   void demoSlot(int param){ … }
};
```

Egyszerű, egyablakos alkalmazások

Események paraméterezése és kiváltása

- Az események paraméterezhetőek
  - az esemény paraméterátadását az eseménykezelőnek a társításnál adhatjuk meg, pl.:

```
connect(&demo, SIGNAL(demoSignal(int)),
        &demo, SLOT(demoSlot(int)));
```

  - a paraméterek átadása sorrendben történik, ezért csak a típust jelezzük
  - az eseménynek legalább annyi paraméterrel kell rendelkeznie, mint az eseménykezelőnek
  - lehetnek alapértelmezett paraméterek is, pl.:

```
signals:
   void demoSignal(int param = 0);
```

Egyszerű, egyablakos alkalmazások

Példa

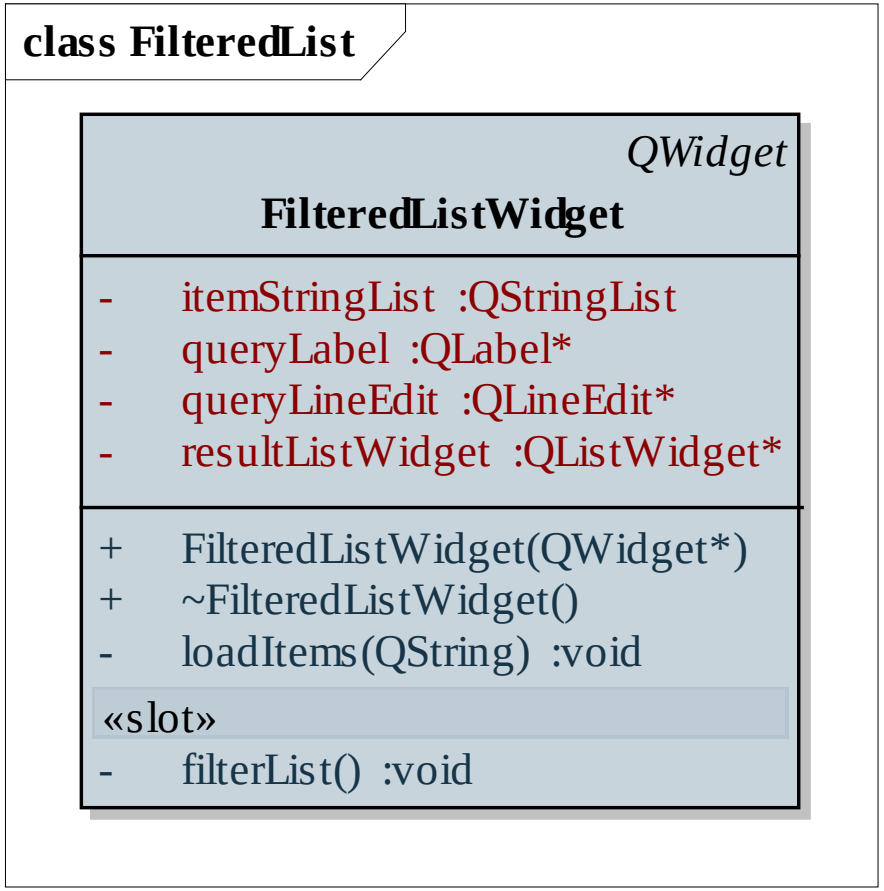
Feladat: Készítsünk egy egyszerű alkalmazást, amelyben egy szavakból álló listát jelenítünk meg, és egy szövegdoboz segítségével szűrhetjük a tartalmat. A szavakat szöveges fájlból töltjük be.

- a saját ablakban (**`FilteredListWidget`**) felveszünk egy listamegjelenítőt (**`QListWidget`**) és egy szövegdobozt (**`QLineEdit`**)
- szükségünk van egy egyedi eseménykezelőre (**`filterList`**), amely a szűrést elvégzi
- a betöltés az **`input.txt`** fájlból történik, először egy szöveglistába (**`QStringList`**), ehhez Qt-s fájlkezelést alkalmazunk (**`QFile`**, **`QStringList`**)

Egyszerű, egyablakos alkalmazások

Példa

Tervezés:



Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`filteredlistwidget.cpp`*):*

```
class FilteredListWidget : public QWidget {
   Q_OBJECT
   …
private slots: // eseménykezelők
   void filterList(); // lista szűrése
private:
   …
   QStringList itemStringList; // szavak listája
   QLabel *queryLabel; // címke
   QLineEdit *queryLineEdit; // sorszerkesztő
   QListWidget *resultListWidget;
      // listamegjelenítő
};
```

Egyszerű, egyablakos alkalmazások

Példa

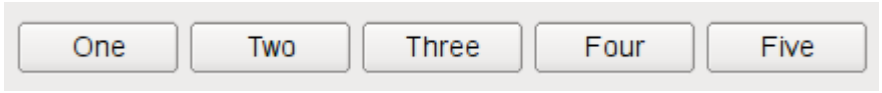
Megvalósítás (`filteredlistwidget.cpp`*):*

```
void FilteredListWidget::filterList(){
   …
      for (int i = 0; i < itemStringList.size();
           i++)
         if (itemStringList[i].contains(
             queryLineEdit->text()))
            // ha tartalmazza a megadott szöveget
            resultListWidget->addItem(
               itemStringList[i]);
               // akkor felvesszük a listára
   }
}
```

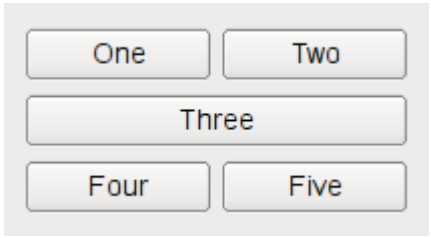
Egyszerű, egyablakos alkalmazások

Vezérlők elrendezése

- Mivel az ablak átméretezésével a vezérlők elhelyezkedését módosítani kell, célszerű az átméretezhető ablakoknál *elhelyezés*eket (*layout*) használni
- Az elhelyezések gyerekvezérlőiket megfelelő sorrendben jelenítik meg, automatikusan áthelyezik és átméretezik, pl.:

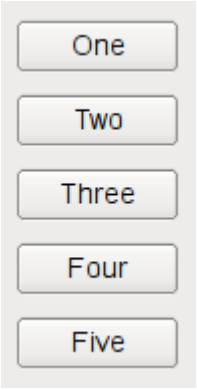


QHBoxLayout



QGridLayout

QFormLayout



QVBoxLayout

Egyszerű, egyablakos alkalmazások

Vezérlők elrendezése

- Az elhelyezéseket ráállíthatjuk a vezérlőre (elsősorban az ablakra) a **`setLayout(QLayout*)`** utasítással
- Számos formának megfelelően helyezhetjük a vezérlőket
  - vízszintes (**`QHBoxLayout`**), függőleges (**`QVBoxLayout`**), rács (**`QGridLayout`**)
  - űrlap (**`QFormLayout`**), amelyen címkézhetjük a vezérlőket
  - keret (**`QBorderLayout`**), amely az oldalához, vagy középre tudja igazítani az elemeket
  - dinamikus (**`QStackedLayout`**), ahol változhat a megjelenő elem
  - az elemek távolsága szabályozható (**`spacing`**)

Egyszerű, egyablakos alkalmazások

Vezérlők elrendezése

- Pl.:

```
QGridLayout* layout = new QGridLayout();
layout->addWidget(firstButton, 0, 0);
   // gomb behelyezése az 1. sor 1. oszlopába
layout->addWidget(thirdButton, 0, 1, 1, 2);
   // gomb behelyezése a 2. sor 1. oszlopában úgy,
   // hogy két oszlopon is átnyúljon
QFlowLayout* innerLayout = new QFlowLayout();
   // belső, folyamatos elhelyezés
…
layout->addLayout(innerLayout);
   // elhelyezés beágyazása
setLayout(layout);
   // elhelyezés beágyazása az ablakba
```

Egyszerű, egyablakos alkalmazások

Példa

Feladat: Módosítsuk az előző alkalmazást úgy, hogy lehessen átméretezni az ablakot, és a tartalom alkalmazkodjon az új mérethez, továbbá lehessen tetszőleges szöveges fájl tartalmát betölteni

- felveszünk egy új gombot, amely a betöltésre szolgál, és hozzá egy új eseménykezelőt (**`loadFile`**)
- a felhasználó egy fájl kiválasztó dialógusablakban (**`QFileDialog`**) adhatja meg a fájl nevét
- a felületen felveszünk két elrendezést, egy vízszinteset (**`QHBoxLayout`**) a felső sornak, és egy függőlegeset a teljes tartalomnak (**`QVBoxLayout`**)

Egyszerű, egyablakos alkalmazások

Példa

Tervezés:

Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`filteredlistwidget.cpp`*):*

```
FilteredListWidget::FilteredListWidget(QWidget
    *parent) : QWidget(parent) {
    …
    mainLayout = new QVBoxLayout;
    mainLayout->addLayout(upperLayout);
       // másik elrendezés felvétele
    mainLayout->addWidget(resultListWidget);
    mainLayout->addWidget(loadButton);

    setLayout(mainLayout); // elrendezés beállítása
    …
}
```

Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`filteredlistwidget.cpp`*):*

```
void FilteredListWidget::loadFile() {
   QString fileName =
      QFileDialog::getOpenFileName(this,
         trUtf8("Fájl megnyitása"), "",
         trUtf8("Szöveg fájlok (*.txt)"));
         // fájl megnyitó dialógus használata,
         // megadjuk a címét és a szűrési
         // feltételt
      if (!fileName.isNull())
         // ha megadtunk valamilyen fájlnevet és
         // OK-val zártuk le az ablakot
         loadItems(fileName);
}
```

Egyszerű, egyablakos alkalmazások

A felülettervező

- A *felülettervező* (*Qt Designer*) lehetőséget ad a felület gyors elkészítésére
  - az elkészített terv XML-ben mentődik (**`<ablaknév>.ui`**), majd abból Qt osztály készül (**`moc_<ablaknév>.h`**)
  - a generált osztály az tervezőben adott név (**`name`**) tulajdonságot kapja névként, valamint az **`Ui_`** előtagot (ehelyett használhatjuk az **`Ui`** névteret)
  - a vezérlőkre a megfelelő névvel hivatkozhatunk, a kialakítás a generált osztály **`setupUi(QWidget* parent)`** metódusába kerül
  - az így generált osztályt a saját osztályokban attribútumként használjuk fel, és hivatkozunk rajta keresztül a vezérlőkre

Egyszerű, egyablakos alkalmazások

A felülettervező

```
#include "ui_demowindow.h"
   // tervező által generált osztály
…
class DemoWindow : public QWidget {
   Q_OBJECT
public:
   DemoWindow(QWidget *parent = 0) : QWidget(parent){
      ui->setupUi(this);
         // innentől használhatóak a vezérlők
         // pl. ui->quitButton->…
   …
private:
   Ui::DemoWindow ui;
};
```

Egyszerű, egyablakos alkalmazások

Példa

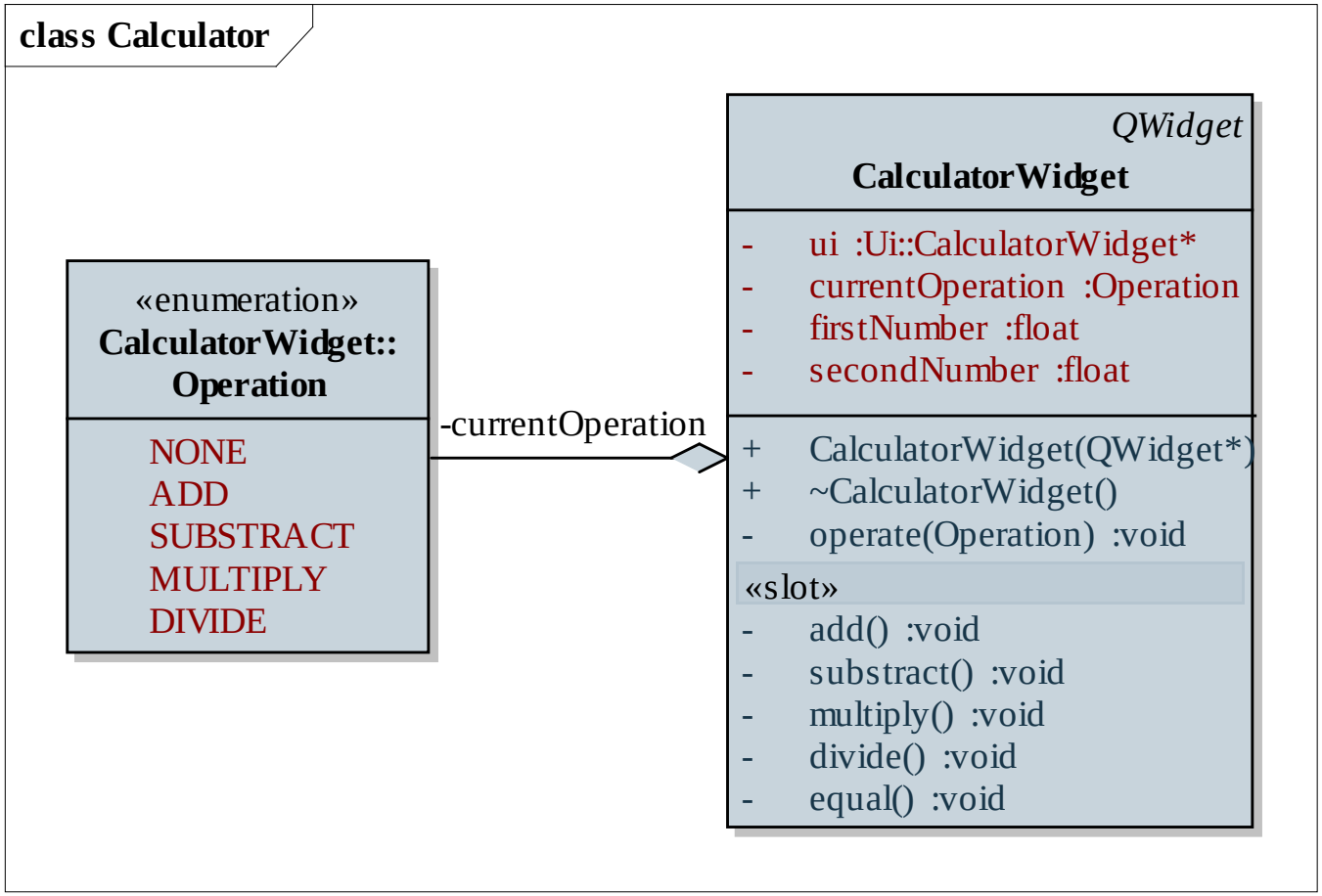
Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, egy beviteli mezővel, amely az előző művelet eredményét jeleníti meg.

- az alkalmazás felületét a felülettervezővel készítjük el, elhelyezünk 5 gombot, valamint egy szövegbeviteli mezőt használunk
- az ablak osztályban (`CalculatorWidget`) létrehozunk öt eseménykezelőt a gombokra, amelyek a megfelelő műveleteket végzik el
- ügyelnünk kell arra, hogy mindig az előző műveletet végezzük el, ne az aktuálisan megadottat, ezért az előző műveletet, illetve az értéket mindig eltároljuk
- a szövegmezőbe csak számok bevitelét tesszük lehetővé

Egyszerű, egyablakos alkalmazások

Példa

Tervezés:



Egyszerű, egyablakos alkalmazások

Példa

Megvalósítás (`calculatorwidget.cpp`*):*

```
CalculatorWidget::CalculatorWidget(QWidget
    *parent) : QWidget(parent),
    ui(new Ui::CalculatorWidget)
{       // grafikus felület létrehozása
    ui->setupUi(this);
       // grafikus felület összeállítása
    setFixedSize(172,250); // méret rögzítése
    …
    ui->numberLineEdit->setFocus();
       // a szövegmezőre állítjuk a fókuszt
    ui->numberLineEdit->selectAll();
       // az összes szöveg kijelölése
}
```