Egyszerű, egyablakos alkalmazások

Vezérlők, elrendezők, dialógus ablakok,

egyedi eseménykezelés

Ablakok

- A Qt-ben ablaknak minősül minden olyan vezérlő (**`QWidget`**-ből származtatott osztály példánya), amelynek nem adjuk meg a szülőjét.
- Az ablak speciális tulajdonságai:
  - módosítható a címe (**`windowTitle`**), ikonja (**`windowIcon`**) vagy megjeleníthető ezek nélkül: **`setWindowState(Qt::WindowFullScreen)`**
  - állítható a mérete teljes/normál módra, vagy lecsukható a tálcára (**`showMaximized`**, **`showNormal`**, **`showMinimized`**)
  - Lekérdezhető, hogy fókuszban van-e (**`isActiveWindow`**), vagy fókuszba tehető (**`activeWindow()`**)
- Egy ablak lehet
  - modális: csak bezárásával lehet az alkalmazás másik ablakát fókuszba tenni
  - nem modális: bezárása nélkül átválthatunk az alkalmazás másik ablakára

Grafikus vezérlők

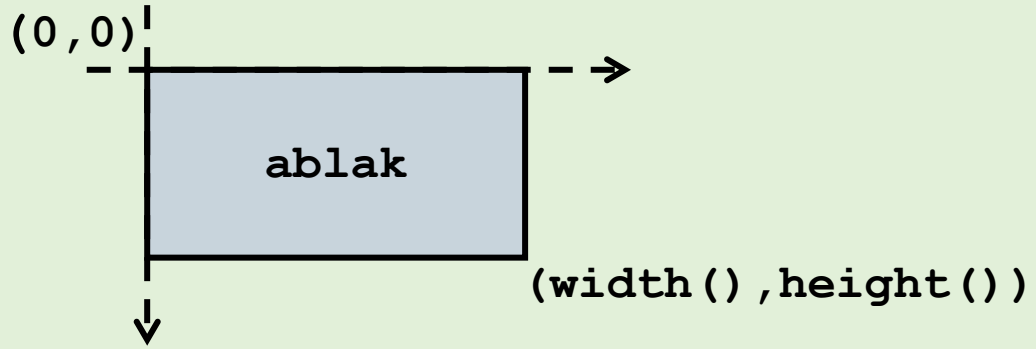
- A leggyakrabban használt (**`QWidget`**-ből származó, előre definiált) grafikus vezérlők:
  - címke (**`QLabel`**)
  - LCD kijelző (**`QLCDNumber`**), folyamatjelző (**`QProgressBar`**)
  - nyomógomb (**`QPushButton`**), kijelölő gomb (**`QCheckBox`**), rádiógomb (**`QRadioButton`**)
  - szövegmező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

Vezérlők fontosabb tulajdonságai

- méret (**`size`**), vagy geometria (elhelyezkedés és méret, **`geometry`**)
  - A vezérlők mérete többféleképpen befolyásolható: változtatható méretűek esetén külön állítható minimum (**`minimumSize`**), maximum (**`maximumSize`**), valamint az alapértelmezett (**`baseSize`**) méret. A méret rögzíthető (**`setFixedSize`**).
- szöveg (**`text`**), betűtípus (**`font`**), stílus (**`styleSheet`**), színpaletta (**`palette`**), előugró szöveg (**`toolTip`**)
  - A grafikus vezérlőkön (pl. **`QLabel`**, **`QLineEdit`**) elhelyezett szöveg formázható több módon pl. formátummal (**`textFormat`**), vagy HTML formázó utasításokkal.
- fókuszáltság (**`focus`**),
- láthatóság (**`visible`**)
- engedélyezés (használható-e a vezérlő, **`enabled`**)

Vezérlő ablakban

- Amennyiben egy grafikus vezérlőt egy ablakban helyezünk el, meg kell adnunk az elhelyezkedését (geometriáját), más szóval a pozícióját és méretét (**`setGeometry(int, int, int, int)`**).
  - Az ablak koordinátarendszere a bal felső sarokból indul a (0,0) koordinátával, és balra, illetve lefelé növekszik.



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

Vezérlők elhelyezkedési hierarchiája

- Egy alkalmazás grafikus vezérlői között elhelyezkedési hierarchiát állíthatunk fel, amely egy erdőnek megfelelő struktúra ír le.
  - A vezérlőnek lehet szülője (**`parent`**), amelyet a vezérlő konstruktorának paraméterével, vagy a **`parent`** tulajdonságával adhatunk meg.
  - A vezérlőnek lehetnek gyerekei (**`children`**).
  - Ha egy szülő vezérlőt elrejtünk/megjelenítünk, ki-/bekapcsolunk, vagy megsemmisítünk, akkor az összes gyerekein is végrehajtódik ugyanez a tevékenység.

Tulajdonságok lekérdezése, módosítása

- A vezérlők tulajdonságai az adattagjainak lekérdező (*getter*), illetve beállító (*setter*) műveleteinek segítségével szabályozhatók
  - a lekérdező művelet neve a tulajdonság neve,
  - a beállító művelet tartalmaz egy **`set`** előtagot

a szöveg más nyelven történő megjelenítését szolgálja

```
QLabel myLabel;                          // címke létrehozása
myLabel.setText(tr("Hello World!"));// címke szövege(text)
QString text = myLabel.text();           // lekérdezzük a címke szövegét
```

Egyedi ablakok

- Saját testreszabott ablakainkat saját osztályból kell példányosítani.

```
class MyWindow : public QWidget
{
public:
    MyWindow(QWidget* parent = nullptr);
private:
    QPushButton* quitButton;  // gomb az ablakon
};
```

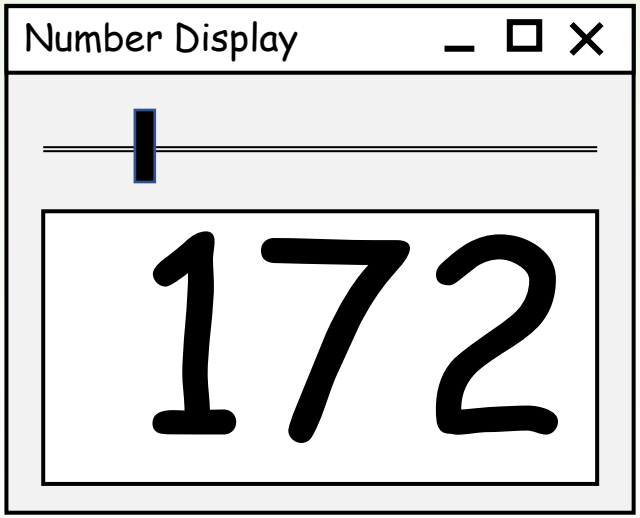
a konstruktor megkaphatja a szülőt

```
MyWindow::MyWindow(QWidget* parent) : QWidget(parent)
{
    setBaseSize(200, 120);
    setWindowTitle(tr("Demo Window"));
    quitButton = new QPushButton("Quit", this);
    quitButton->setGeometry(10, 40, 180, 40);

    connect(quitButton, SIGNAL(clicked()),
        QApplication::instance(), SLOT(quit()));
}
```

az eseménykezeléshez lekérdezzük az alkalmazás-példányt

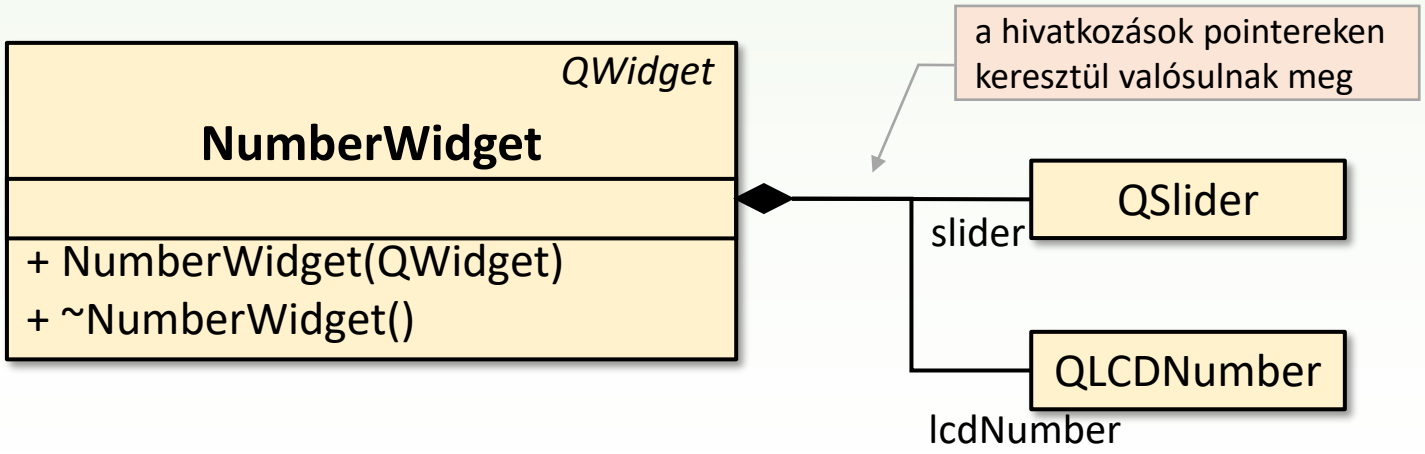
1.Feladat

Készítsünk egy egyszerű alkalmazást, amelyben egy csúszkával állíthatjuk a digitális kijelzőn megjelenő számot.



1.Feladat: tervezés

- Az alkalmazás számára létrehozunk egy saját ablak osztályt (**`NumberWidget`**), felhelyezünk rá egy csúszkát (**`QSlider`**), és egy digitális számkijelzőt (**`QLCDNumber`**).
  - a konstruktorban állítjuk be a vezérlők tulajdonságait, és itt adjuk meg a működéshez szükséges signal-slot társítást
  - a destruktor üres, hiszen az ősosztály destruktora törli az ablakot annak gyermekeivel együtt.

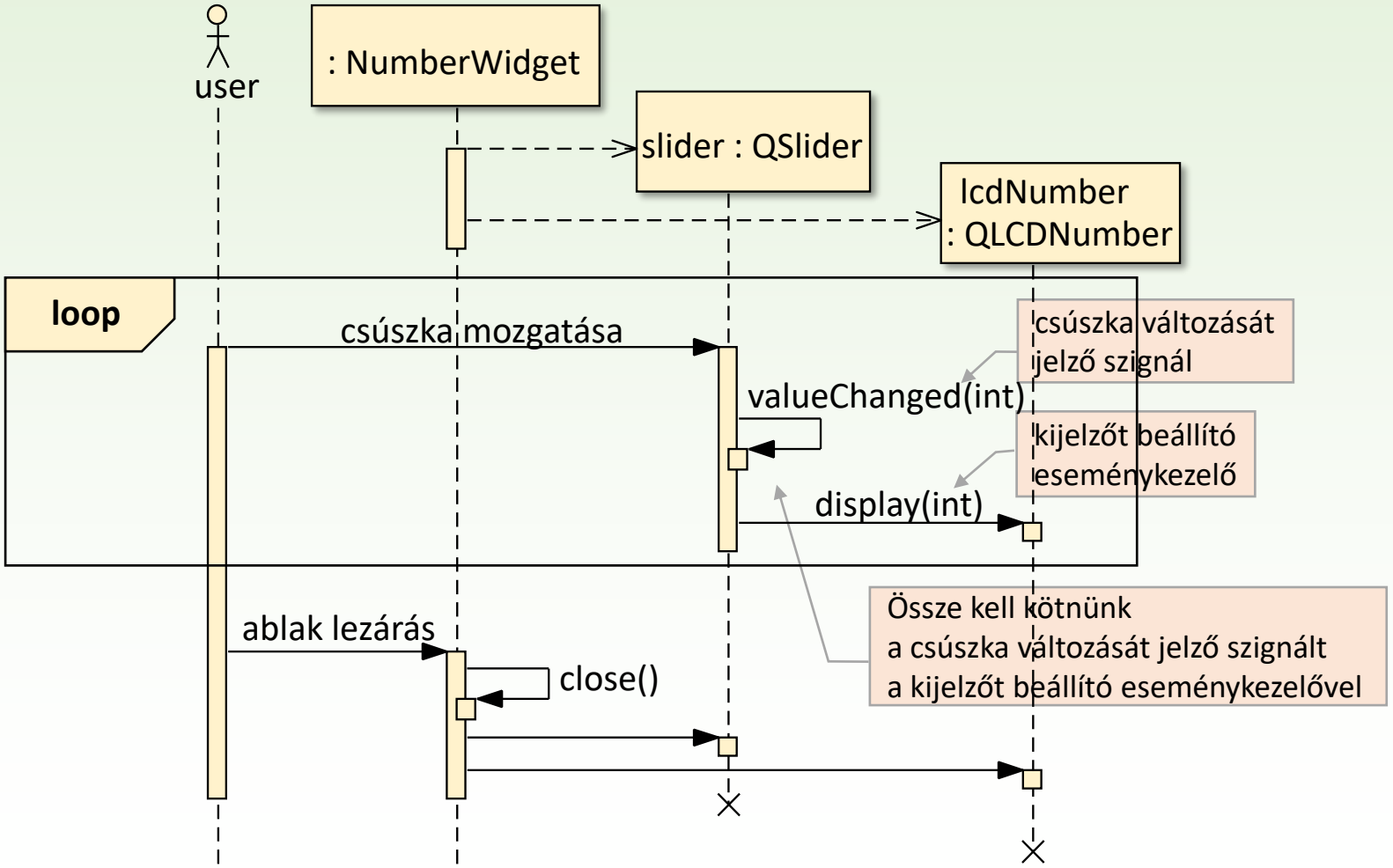


1.Feladat: megvalósítás kezdete

```
#include <QApplication>
#include "numberwidget.h"
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    NumberWidget w;
    w.show();
    return a.exec();
}
```

```
#include <QWidget>
#include <QSlider>
#include <QLCDNumber>
class NumberWidget : public QWidget
{
public:
    NumberWidget(QWidget *parent = nullptr);
    ~NumberWidget(){};
private:
    QSlider* _slider;
    QLCDNumber* _lcdNumber;
};
```

1.Feladat: tervezés



1.Feladat: konstruktor


```
NumberWidget::NumberWidget(QWidget *parent) : QWidget(parent)
{
    setWindowTitle(tr("Number Display")); // ablakcím
    setFixedSize(400, 175);              // rögzített méret beállítása
    _slider = new QSlider(this);         // a vezérlő szülője az ablak
    _slider->setMinimum(0);              // számhatárok beállítása
    _slider->setMaximum(1000);
    _slider->setValue(0);                // aktuális érték beállítása
    _slider->setOrientation(Qt::Horizontal);// csúszkairány
    _slider->setGeometry(5, 5, 390, 30);    // elhelyezkedés
    _lcdNumber = new QLCDNumber(4, this);   // a számjegyek száma
    _lcdNumber->display(0);                  // érték megjelenítése
    _lcdNumber->setGeometry(5, 50, 390, 120);

    connect(_slider, SIGNAL(valueChanged(int)),
            _lcdNumber, SLOT(display(int)));
}
```

Össze kell kötnünk
a csúszka változását jelző szignált
a kijelzőt beállító eseménykezelővel

Egyedi szignálok és eseménykezelők

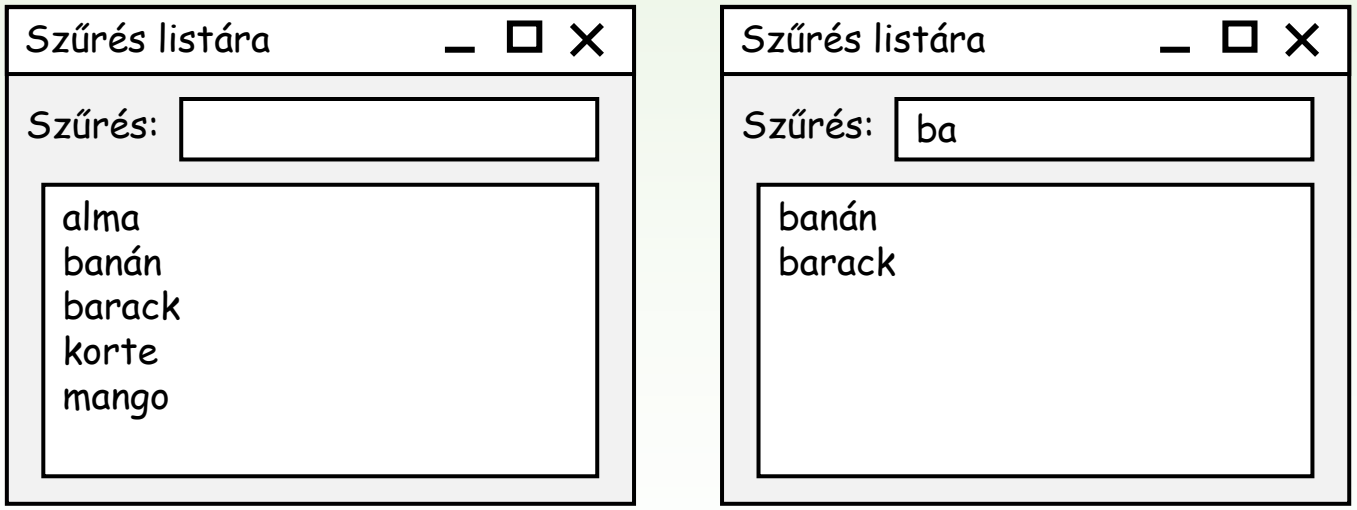
- Egyedi szignálokat és kezelőket csak azon **`QObject`** osztályból származtatott osztályokban használhatunk, amelyek tartalmazzák a **`Q_OBJECT`** makrót.
- A szignálokat és kezelőiket az osztály **`signals`** és **`slots`** részében kell deklarálni **`void`** típussal, tetszőleges paraméterezéssel, a láthatóságuk megadásával együtt.

```
class MyObject : public QObject {
    Q_OBJECT
private signals:
    void mySignal(int param = 0);
public slots:
    void mySlot(int param){ … }
};
…
connect(this, SIGNAL(mySignal(int)),
        this, SLOT(mySlot(int)));
```

- A szignálnak legalább annyi paraméterrel kell rendelkeznie, mint a neki megfeleltetett eseménykezelőnek. A paraméterek átadása sorrendben történik, ezért a társításnál (connect) csak a típusokat jelezzük. A szignál paramétereinek lehet alapértelmezett értéke is.

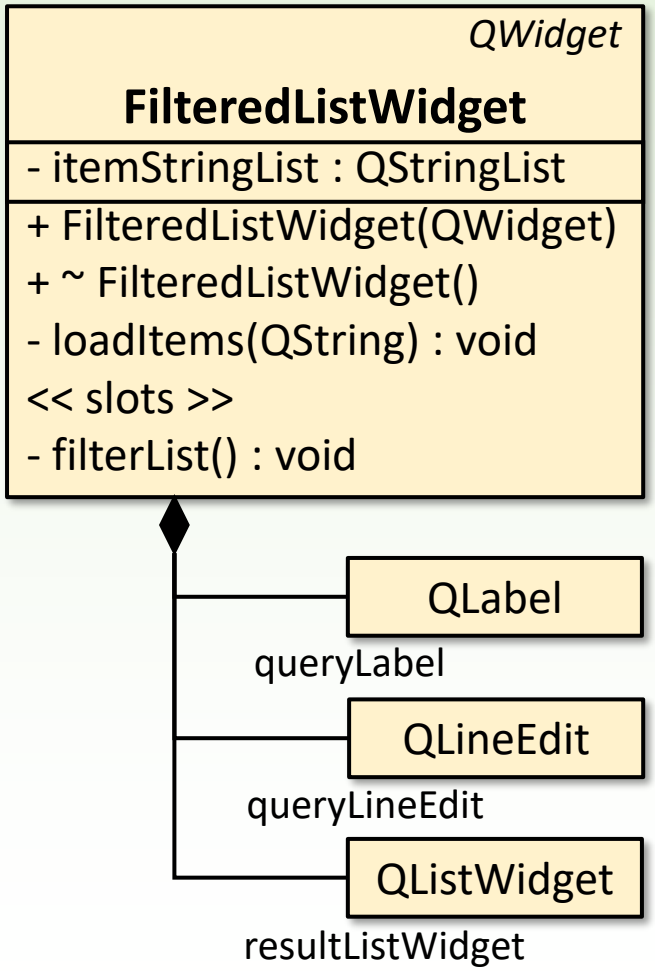
2.Feladat

Készítsünk egy egyszerű alkalmazást egy szavakból álló lista megjelenítésére, amely tartalmát egy szövegdobozban megadott sztring alapján szűrhetjük. A szavakat szöveges állományból töltjük be.

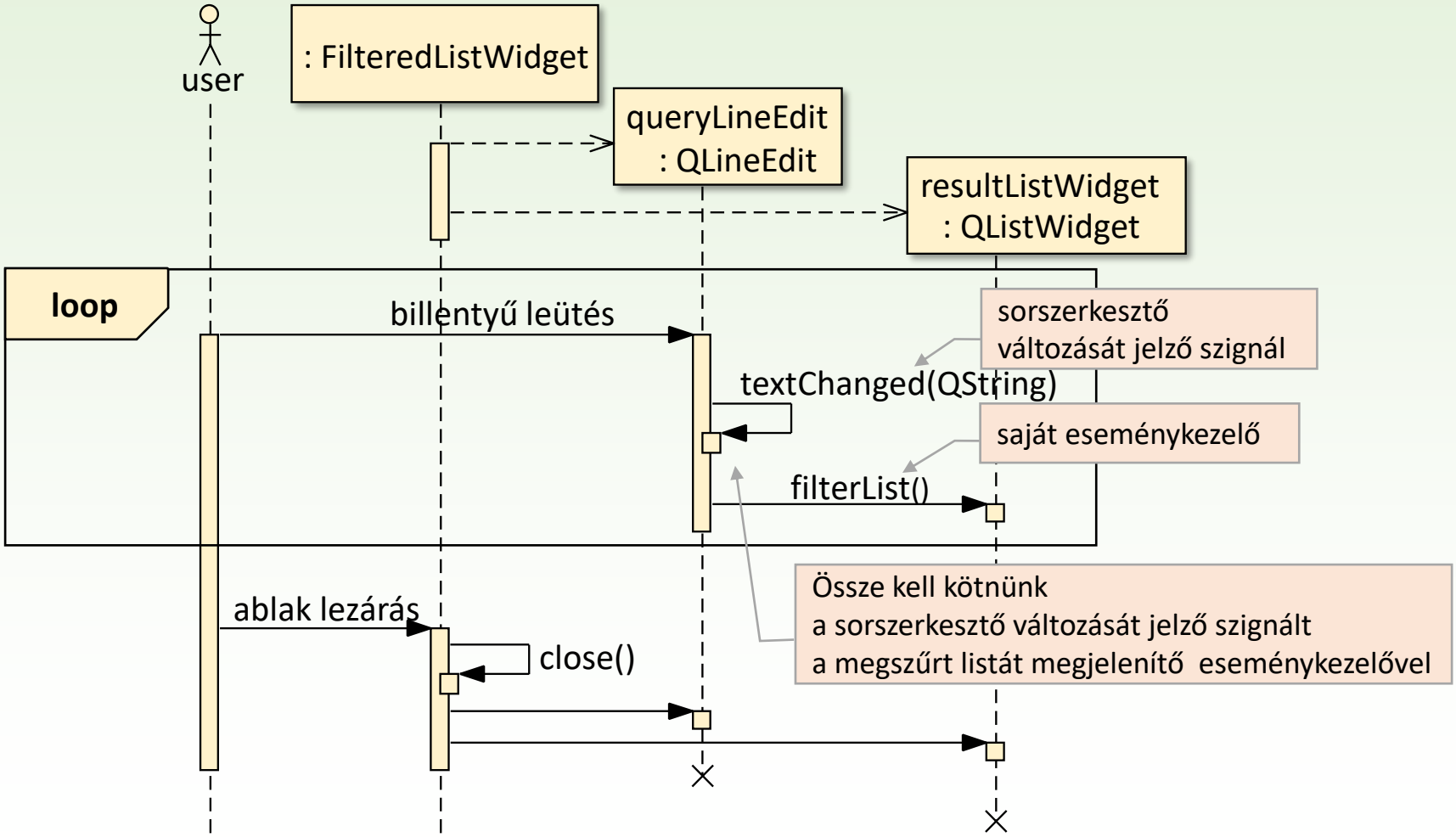


2.Feladat: tervezés

- Az új ablak (**`FilteredListWidget`**) grafikus felülete tartalmaz egy listamegjelenítőt (**`QListWidget`**) és egy szövegdobozt (**`QLineEdit`**) egy címkével (**`QLabel`**).
- A háttérben a szűretlen szavak listáját egy szöveglistában tároljuk (**`QStringList`**). Ezt az **`input.txt`** fájlból töltjük fel (**`loadItems()`**) Qt-s fájlkezelést használva (**`QFile`**).
- Szükségünk van továbbá egy egyedi eseménykezelőre (**`filterList()`**), amely a szűrést elvégzi.



2.Feladat: tervezés



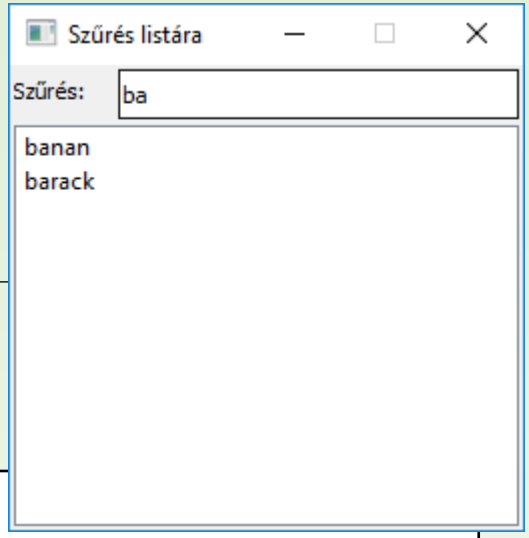
2.Feladat: megvalósítás

```
#include <QWidget>
#include <QLabel>
#include <QLineEdit>
#include <QListWidget>

class FilteredListWidget : public QWidget {
    Q_OBJECT
public:
    FilteredListWidget(QWidget *parent = nullptr);
    ~FilteredListWidget();
private slots:
    void filterList();    // lista szűrése
private:
    void loadItems(QString fileName); // adatok betöltése fájlból

    QStringList _itemStringList;  // szavak listája
    QLabel *_queryLabel;          // címke
    QLineEdit *_queryLineEdit;    // sorszerkesztő QListWidget
    *_resultListWidget;           // listamegjelenítő
};
```

2.Feladat: megvalósítás

```
FilteredListWidget::FilteredListWidget(QWidget *parent)
 : QWidget(parent)
{
    setFixedSize(354, 232);
    setWindowTitle(tr("Szűrés listára"));

    _queryLabel = new QLabel(tr("Szűrés:"), this);
    _queryLabel->setGeometry(2, 2, 50, 20);

    _queryLineEdit = new QLineEdit(this);
    _queryLineEdit->setGeometry(54, 2, 200, 25);

    _resultListWidget = new QListWidget(this);
    _resultListWidget->setGeometry(2, 30, 352, 200);

    connect(_queryLineEdit, SIGNAL(textChanged(QString)),
                        this, SLOT(filterList()));

    loadItems("input.txt");
}
```

2.Feladat: megvalósítás




```
void FilteredListWidget::filterList()
{
    _resultListWidget->clear(); // kitöröljük a korábbi tartalmat

    if (_queryLineEdit->text().isNull()) { // ha nincs szűrés
        _resultListWidget->addItems(_itemStringList);
            // mindent felveszünk a listára
    } else { // ha van szűrés
        for (int i = 0; i < _itemStringList.size(); i++)
            if (_itemStringList[i].contains(_queryLineEdit->text()))
                _resultListWidget->addItem(_itemStringList[i]);
                    // felvesszük a listára, ha tartalmazza a
                    // megadott szöveget
    }
}
```

2.Feladat: megvalósítás

```
void FilteredListWidget::loadItems(QString fileName)
{
    QFile file(fileName);                  // logikai fájl létrehozása
    if (file.open(QFile::ReadOnly)) { // megnyitás csak olvasásra
        _itemStringList.clear();       // régi elemek törlése
        QTextStream stream(&file);     // szövegként olvassuk be a fájlt
        QString line = stream.readLine(); // soronként olvasunk
        while (!stream.atEnd()) {      // !line.isNull() is lehetne
            _itemStringList << line;  // _itemStringList.append(line);
            line = stream.readLine();
        }
        _queryLineEdit->clear();       // töröljük tartalmat
        _resultListWidget->clear();    // töröljük tartalmat
        _resultListWidget->addItems(_itemStringList); // új elemek
    } else
        QMessageBox::warning(this, tr("Hiba!"), tr("A ")
                + fileName + tr(" fájl nem található!"));
            // ha nem sikerült megnyitni, előugró ablakot mutatunk
}
```

Dialógus ablakok

- Dialógus ablakok a **`QDialog`** osztályból származtatott osztályok példányai, amely lezárása után lekérdezhető, hogy a felhasználó milyen szándékkal lépett ki belőlük.
- A dialógus ablakokat az **`accept()`** vagy a **`reject()`** eseménykezelővel zárhatjuk be: modális hívás esetén az **`exec()`** metódus igaz értékkel tér vissza, ha az **`accept()`**-et használjuk, hamissal, ha a **`reject()`**-et.
- Egy dialógus ablak megjelenítésének módja
  - modális, ha a híváshoz az **`exec()`** metódust használjuk,
  - nem modális, ha a híváshoz a **`show()`** metódust használjuk és nem állítottuk a **`setModal()`** metódussal eleve modálisra.

Dialógus ablakok

- Léteznek előre definiált dialógus ablakok:
  - Rögzített dialógusok: **`QFileDialog`**, **`QColorDialog`**, **`QFontDialog`**, **`QPrintDialog`**, **`QInputDialog`**, **`QProgressDialog`**, **`QErrorMessage`**.

```
QString fileName = QFileDialog::getOpenFileName(this,
 tr("Open file"), "/home", tr("Text files (*.txt)"));
     // szövegfájl megnyitása a home könyvtárból
```

  - Konfigurálható üzenőablak (**`QMessageBox`**), amely alkalmas üzenet (**`information`**) , hiba (**`critical`**) , figyelmeztetés (**`warning`**) közlésére, vagy kérdés (**`question`**) feltételére.

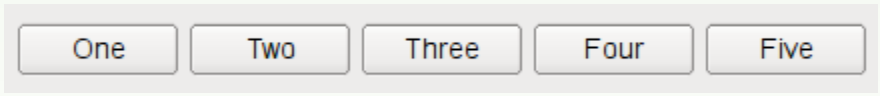
```
QMessageBox::question(this, tr("Confirm"), tr("Do you want to
exit?"), QMessageBox::Yes| QMessageBox::Default, QMessageBox::No);
```

Grafikus vezérlők elrendezése

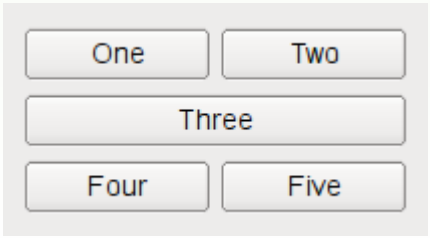
- Mivel az ablak átméretezésével a vezérlők elrendezését módosítani kell, célszerű az átméretezhető ablakoknál elrendezéseket (*layout*) használni.
- Az elrendezések a gyerekvezérlőiket megfelelő sorrendben jelenítik meg, automatikusan áthelyezik és átméretezik.
- Az elrendezések hierarchikusan egymásba ágyazhatók (**`addLayout()`**), és a hierarchia tetején levő elrendezést a **`setLayout(QLayout*)`** utasítással állíthatunk rá az azt tartalmazó vezérlőre (elsősorban az ablakra).
- Az elemek távolsága egy elrendezésen belül szabályozható (**`spacing`**).

Grafikus vezérlők elrendezői

- Számos formának megfelelően rendezhetjük 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

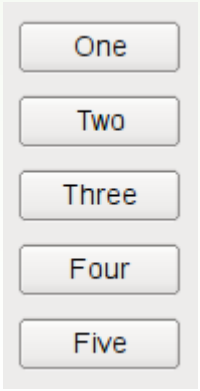


`QHBoxLayout`



`QGridLayout`

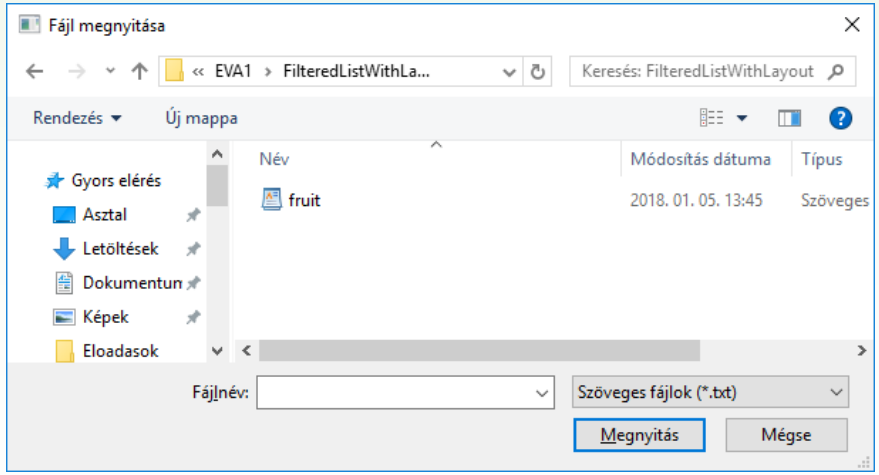
`QFormLayout`



`QVBoxLayout`

3.Feladat

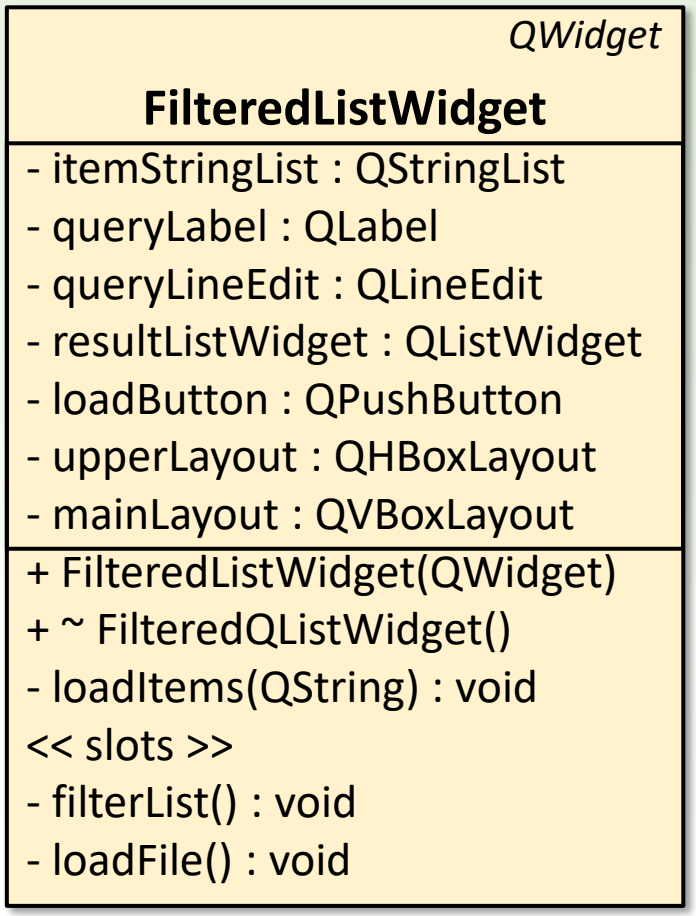
Módosítsuk az előző alkalmazást úgy, hogy lehessen átméretezni az ablakot, és a tartalom alkalmazkodjon az új mérethez, továbbá lehessen tetszőleges szöveges fájl tartalmát betölteni.



3.Feladat: tervezés

Az eddigieken túl

- Felveszünk egy új nyomógombot, amelynek clicked() szignálja a szöveges állományból történő beolvasást indítja el. (**`loadFile()`**).
- A beolvasás egy fájlválasztó dialógus-ablak (**`QFileDialog`**) megjelenésével kezdődik, majd meghívja a már meglevő **`loadItems()`** metódust.
- Alkalmazunk elrendezőket a felületen: a felső sornak egy vízszinteset (**`QHBoxLayout`**), a teljes tartalomnak egy függőlegeset (**`QVBoxLayout`**).



3.Feladat: megvalósítás

```
FilteredListWidget::FilteredListWidget(QWidget *parent) :
QWidget(parent) {
    …
    _upperLayout = new QHBoxLayout;
    _upperLayout->addWidget(_queryLabel);
    _upperLayout->addWidget(_queryLineEdit);

    _mainLayout = new QVBoxLayout;
    _mainLayout->addLayout(_upperLayout);
    _mainLayout->addWidget(_resultListWidget);
    _mainLayout->addWidget(_loadButton);

    setLayout(_mainLayout);
    …
}
```

elrendezők használata

```
void FilteredListWidget::loadFile() {
    QString fileName =
        QFileDialog::getOpenFileName(this,
        tr("Fájl megnyitása"), "", tr("Szöveg fájlok(*.txt)"));
    if (!fileName.isNull()) loadItems(fileName);
}
```

fájl megnyitó dialógus

ha OK-val zártuk le a fájl dialógust