

Програмування GUI

На основі мови C++ та фреймворку Qt

Лекція 2

Клас QWidget

Клас QWidget є базовим для усіх класів віджетів Qt.

Конструктор:

```
QWidget (QWidget *parent=0; Qt::WindowFlags  
f=0),
```

де `parent` – покажчик на батьківський віджет,
`f` – задає вид (тип) вікна.

Якщо `parent=0` – будет створено вікно верхнього
урівня, з заголовками та системним меню;

якщо `parent` вказує на конкретний віджет, то нове
вікно буде створено на його поверхні

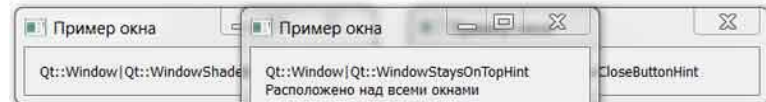
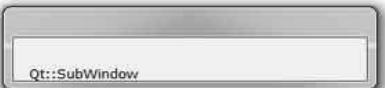
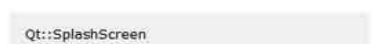
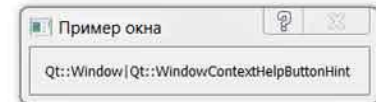
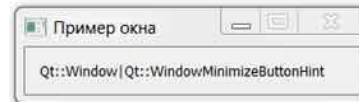
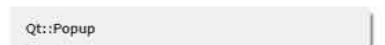
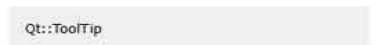
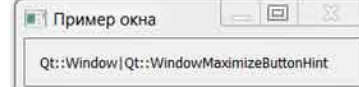
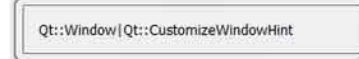
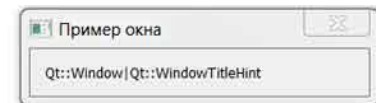
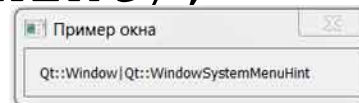
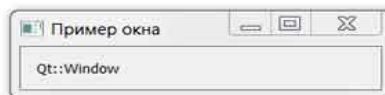
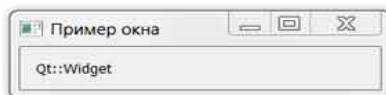
Клас QWidget

параметр: `Qt::WindowFlags` служить для задания
 властивостей вікна, і з його допомогою можна керувати
 зовнішнім виглядом вікна і режимом відображення


`QWidget w(0, Qt::Window);`

Метод `setWindowFlags ()`:

`w.setWindowFlags (Qt::Window |
 Qt::WindowStaysOnTopHint);`



Слоти та методи класу QWidget



```
QWidget *w=new QWidget();
```


- `setWindowTitle ()` встановлює напис заголовка вікна :



```
w->setWindowTitle ( "My Window " ) ;
```


- Слот `void show ()` – відображає віджет на екрані:

```
w->show();
```

- 
- Слот `void hide ()` – приховує віджет:

- Слот `setEnabled (bool)` дозволяє зробити віджет доступним \ недоступним для користувача :

- `Enabled (true)` – доступний



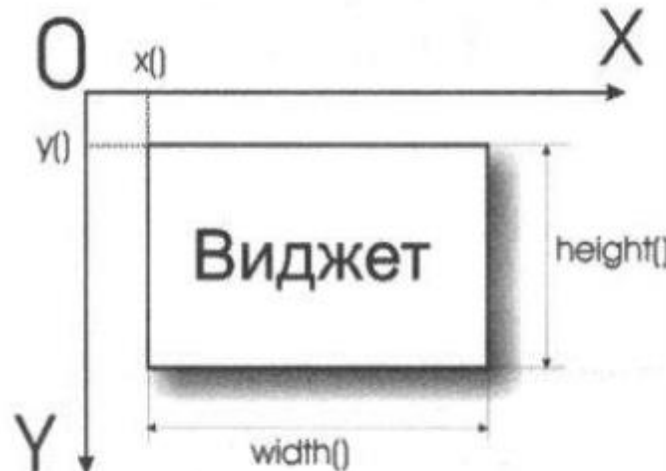
- `Enabled (false)` – недоступний (часто відображається блідо-сірим кольором)

```
w->setEnabled (false); // тепер вікно недоступне
```






Слоти та методи класу QWidget

- Методи `int height()`, `int width()` повертають відповідно висоту і ширину вікна;
- Методи `void setHeight(int h)`, `void setWidth(int w)` задають нові висоту і ширину вікна;
- Методи `int x()` и `int y()` повертають відповідно горизонтальну і вертикальну координати віджета;
- Метод `void setX(int x)`, `void setY(int y)` – задають нові координати віджета



Слоти та методи класу QWidget

-  Метод `void setGeometry (int x, int y, int width, int height)` дозволяє одночасно змінити розташування і розміри віджета;
-  методи, які повертають (змінюють) розміри / координати віджета: `QSize size()`, `Qpoint pos()`, `QRect Geometry()` ;
-  Метод `void move(int x, int y)` – дозволяє змінити розташування віджета
- Метод `void resize(int width, int height)` – дозволяє змінити розміри вікна

 `w->move (5, 5) ;`

`w->resize (260, 330) ;`

`w->setGeometry (10, 20, 30, 40) ;`

Слоти та методи класу QWidget

Фон віджету

- властивість `autoFillBackground` (за замовчуванням дорівнює `false`):

```
wgt.setAuto FillBackground(true) ;
```

```
#include <QtWidgets>
```



```
int main(int argc, char *argv[])  
{  
    QApplication a(argc, argv);  
    QWidget wgt;  
    QWidget* pwgt1 = new QWidget (&wgt);  
    QPalette pall;  
    pall.setColor (pwgt1->backgroundRole(), Qt::blue) ;  
    pwgt1->setPalette (pall);  
    pwgt1->resize(100, 100);  
    pwgt1->move (25, 25);  
    pwgt1->setAutoFillBackground (true) ;  
    QWidget* pwgt2 = new QWidget (&wgt);  
    QPalette pal2;  
    pal2.setBrush (pwgt2->backgroundRole(), QBrush(QPixmap(":/stone.jpg")));  
    pwgt2->setPalette (pal2);  
    pwgt2->resize (100, 100);  
    pwgt2->move (75, 75);  
    pwgt2->setAutoFillBackground(true) ;  
    wgt.resize(200, 200);  
    wgt.show();  
    return a.exec();  
}
```



Слоти та методи класу QWidget

Зміна покажчика миші

Клас покажчика (курсора) миші `QCursor` визначено у файлі `QCursor`.

Встановити зображення покажчика можна методом `setcursor()`, передавши йому одне зі стандартних значень. У класі `QCursor` міститься метод `pos()`, який повертає поточну позицію покажчика миші щодо лівого верхнього кута екрану. За допомогою методу `setpos()` можна переміщати покажчик миші.

Приклад

```
#include <QtWidgets>

int main(int argc, char *argv[])
{
    QApplication app{argc, argv};
    QWidget wgt;
    QPixmap pix(":/clock.png");
    QCursor cur (pix);
    wgt.setCursor(cur);
    wgt.resize (160, 100);
    wgt.show ();
    return app.exec();
}
```

Клас QFrame



Призначений для створення рамки.



Успадкований від класу `QWidget`.



Конструктор:


```
QFrame (QWidget *parent=0,  
        Qt::WindowFlags f=0),
```




де `parent` – покажчик на батьківське вікно,
`f` – задає вид рамки (як у `QWidget`).



Клас QFrame



Основною відмінністю класу `QFrame` від `QWidget` є метод `void setFrameStyle(int style)`, що задає тип рамки:




`(QFrame::Box | QFrame::Plain` - рамка



`(QFrame::HLine | QFrame::Plain` –
горизонтальна лінія

`(QFrame::VLine | QFrame::Plain` –
вертикальна лінія



Товщину рамки в пікселях можна задати
методом `setLineWidth(int w)`.



Клас QLabel



Клас `QLabel` успадкований від класу `QFrame` і призначений для створення об'єктів текстових підписів (міток). Також дозволяє відображати графічну інформацію (GIF, MNG та т.д.).



Віджет напису служить для показу стану додатка або пояснюючого тексту і являє собою текстове поле, текст якого не підлягає зміні з боку користувача



Клас QLabel



Конструктор

`QLabel (const QString &text, QWidget
*parent=0, Qt::WindowFlags f=0),`



де `text` – задає текст мітки, інші параметри – як в `QWidget`.



Слот `void setAlignment (Qt::Alignment)` –
управляє розташуванням тексту в поле мітки



- `Qt::AlignHCenter` | `Qt::AlignVCenter` –
по центру по горизонталі і по вертикалі,
- `Qt::AlignTop` | `Qt::AlignLeft` – в
верхньому лівому кутку мітки




Клас QLabel


```
#include <QtWidgets>

int main(int argc, char *argv[])
{
    QApplication app{argc, argv};
    QPixmap pix;
    pix.load(":/mira.ipg");
    QLabel lbl;
    lbl.resize (pix.size());
    lbl.setPixmap (pix);
    lbl.show();
    return app.exec();
}
```


Клас QLabel




Метод `void setBuddy (QWidget * buddy)` пов'язує мітку з будь-яким віджетом, що володіє фокусом введення.



Якщо текст напису містить "&", то символ, перед яким він стоїть, буде підкресленим, і при спільному натисканні клавіші з цим символом і клавіші <Alt>, фокус введення перейде до віджету, встановленому методом `setBuddy ()`.



```
QLabel *lbl = new QLabel ("&Input",  
    this);
```



```
QLineEdit *edit = new QLineEdit(this);  
lbl->setBuddy(edit);
```


Клас `QAbstractButton`



Клас `QAbstractButton` - базовий для всіх кнопок.



У додатках застосовуються три основні види кнопок:



- Кнопки, що натискаються (`QPushButton`), які зазвичай називають просто кнопками;
- прапорці (`QCheckBox`);
- перемикачі (`QRadioButton`).



Клас QAbstractButton

Установка тексту і зображення

1. Передати текст кнопки в конструкторі першим параметром;
2. Встановити за допомогою методу `setText()` ;
3. `setIcon()` - встановити растрове зображення

Клас QAbstractButton

Для взаємодії з користувачем клас `QAbstractButton` надає наступні сигнали :

- `clicked ()` - відправляється при натисканні кнопкою миші;
- `pressed ()` - відправляється при натисканні на кнопку миші;
- `released ()` - відправляється при відпуску кнопки миші;
- `toggled ()` - відправляється при зміні стану кнопки, що має статус вимикача.









Клас QAbstractButton



Опитування стану


Для опитування поточного стану кнопок в класі


`QAbstractButton` визначені три методи:


- 
- `isDown ()` - повертає значення `true`, якщо кнопка знаходиться в натиснутому стані. Змінити поточний стан може або користувач, натиснувши на кнопку, або виклик методу `setDown ()` ;
 -  `isChecked ()` - повертає значення `true`, коли кнопка перебуває у включеному стані. Змінити поточний стан може або користувач, натиснувши на кнопку, або виклик методу `setChecked ()` ;
 -  `isEnabled ()` - повертає значення `true`, коли кнопка доступна, тобто реагує на дії користувача. Змінити поточний стан можна викликом методу `setEnabled ()` .
- 
- 
- 
- 
- 

Клас QPushButton

Конструктор:

 QPushButton (const QString
&text, QWidget *parent=0),

 де text – задає текст на кнопці,
parent – покажчик на батьківський об'єкт.

 QPushButton *btn1=new QPushButton
("OK", 0);

Клас QPushButton

Кнопка натискання може функціонувати як:

- «Звичайна» кнопка (Normal button)
- Кнопка-вимикач (Toggle Button)
- «Плоска» кнопка (Flat Button)
- Кнопка із зображенням (Pixmap Button)



Клас QPushButton



```
#include <QtWidgets>
```

```
int main(int argc, char *argv[])  
{
```

```
    QApplication app{argc, argv};
```

```
    QWidget wgt;
```

```
    QPushButton* pcmdNormal = new QPushButton("&Normal Button");
```

```
    QPushButton* pcmdToggle = new QPushButton("&Toggle Button");
```

```
    pcmdToggle->setCheckable (true);
```

```
    pcmdToggle->setChecked (true) ;
```

```
    QPushButton * pcmdFlat = new QPushButton("&Flat Button");
```

```
    pcmdFlat->setFlat (true);
```

```
    QPixmap pix(":/ChordsMaestro.png");
```

```
    QPushButton* pcmdPix = new QPushButton("&Pixmap Button");
```

```
    pcmdPix->setIcon (pix);
```

```
    pcmdPix->setIconSize(pix.size());
```

```
    //Layout setup
```

```
    QVBoxLayout* pvbxLayout = new QVBoxLayout;
```

```
    pvbxLayout->addWidget(pcmdNormal);
```

```
    pvbxLayout->addWidget (pcmdToggle);
```

```
    pvbxLayout->addWidget (pcmdFlat);
```

```
    pvbxLayout->addWidget(pcmdPix);
```

```
    wgt.setLayout(pvbxLayout);
```

```
    wgt.show();
```


```
    return app.exec();
```

```
}
```

Класс CheckBox



Конструктор:




```
QCheckBox (const QString &text,  
           QWidget *parent=0) ;
```

де text – задає текст флажка,

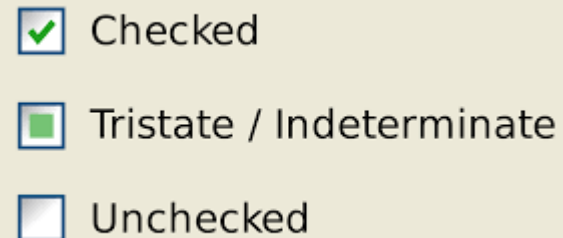


parent – покажчик на батьківський віджет.



```
QCheckBox *chbox= new QCheckBox ("1",  
    this) ;
```

Може мати 2 або 3 стани:

- 
- ☒ Checked
 - ☐ Tristate / Indeterminate
 - ☐ Unchecked

Клас CheckBox

Прапорець з 2 станами

`void setChecked(bool)` – встановити стан флажку

– true - встановити

– false - скинути

• `bool isChecked()` – перевірити стан флажку

– true - встановлено

– false - скинуто

• `void toggled(bool)` – при зміні стану прапорець висилає сигнал

– true - прапорець включений

– false - прапорець виключений

Клас CheckBox

Прапорець з 3 станами

`void setTristate (bool y=true)` – для
переведення звичайного прапорця (2 стану) в
прапорець з 3 станами

- `y=true` - tristate
- `Y=false` – twostate

- `bool isTristate ()` – перевірка кількості станів
 - `true` - 3
 - `false` - 2

- `Qt::CheckState checkState ()` – метод для
перевірки стану прапорця з 3 станами
 - `Qt::Unchecked` – вимкнений (скинутий)
 - `Qt::PartiallyChecked` – частково включений
 - `Qt::Checked` – включений (встановлений)

Клас CheckBox

Прапорець з 3 станами

- `void setCheckState (Qt::CheckState)` – встановити новий стан прапорця з 3 станами
- `void stateChanged (int)` - сигнал при зміні стану
 - 0 – `Qt::Unchecked` – вимкнений (скинутий)
 - 1 – `Qt::PartiallyChecked` – частково вимкнений
 - 2 – `Qt::Checked` – включений (установлений)



Клас CheckBox

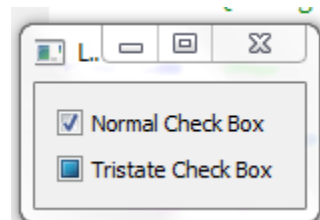
```
#include <QtWidgets>

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QWidget wgt;
    QCheckBox* pchkNormal = new QCheckBox("&Normal Check Box");
    pchkNormal->setChecked (true);
    QCheckBox * pchkTristate = new QCheckBox("&Tristate Check Box");
    pchkTristate->setTristate(true);
    pchkTristate->setCheckState (Qt::PartiallyChecked) ;
    //Layout setup
    QVBoxLayout* pvbxLayout = new QVBoxLayout;
    pvbxLayout->addWidget (pchkNormal) ;
    pvbxLayout->addWidget (pchkTristate) ;
    wgt.setLayout (pvbxLayout) ;
    wgt. show ();

    return a.exec();
}
```





Клас RadioButton



 Конструктор:

```
QRadioButton (const QString &text,  
              QWidget *parent=0) ,
```

 де `text` – задає текст перемикача, `parent` – вказує на батьківський віджет.

 Робота з перемикачами аналогічна роботі з прапорцями з двома станами.

Перевести вимикач у включений \ вимкненому стані
можна слотом `void setChecked(bool)`

 `true` – включений

`false` – виключений

Перевірка стану перемикача– `bool isChecked()`.

Клас RadioButton



Сигнали:

При натисканні на перемикачі висилається сигнал `void clicked ()`.



При зміні стану перемикача висилається сигнал `void toggled (bool state)`

- `true` – включений
- `false` – вимкнений



Якщо все перемикачі розташовані на поверхні одного віджета, то при включенні одного з них решта будуть виключатися. Тому часто для більш зручної обробки сигналів перемикачів їх доцільно об'єднувати в групу.



Клас QPushButtonGroup



Для об'єднання будь-яких кнопок і, зокрема, перемикачів в групу можна використовувати клас QPushButtonGroup. Даний клас не має графічної реалізації, а тільки логічно об'єднує кнопки в групу.



Конструктор

`QPushButtonGroup (QObject *parent=0) ,`

де `parent` - указує на батьківський віджет.



Клас QPushButton



Методи і сигнали класу QPushButton:



Метод `void addButton (QAbstractButton * button)` - додає кнопку в групу, `button` - покажчик на яку потрібно додати кнопку



Метод `void removeButton (QAbstractButton * button)` - видаляє кнопку з групи.



Сигнал `void buttonClicked (QAbstractButton * button)` - висилається при натисканні на одній з кнопок групи, `button` - вказує на обраний (клацнути) об'єкт.

