

# Програмування GUI

На основі мови C++ та фреймворку Qt

## Лекція 5

# Клас QComboBox



Призначений для створення об'єкта комбінованого списку. Елементами списку можуть бути рядки і іконки.



Конструктор:

`QComboBox (QWidget *parent=0) ,`



де `parent` – покажчик на батьківський віджет.



# Методи і слоти класу QComboBox



1. Додавання елемента в список можна реалізувати методом `void addItem (const QString &text, const QVariant &data=QVariant())`,

де `text` – текст елемента, що додається,

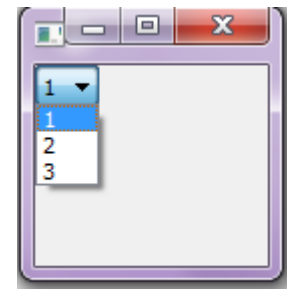
`data` - вказує на асоційовані з елементом довільні дані.

```
QComboBox *cb=new QComboBox (this);
```

```
QString str[3]= {"1", "2", "3"};
```

```
for (int i=0; i<3; i++)
```

```
{ cb->addItem(str[i]); }
```



# Методи і слоти класу QComboBox

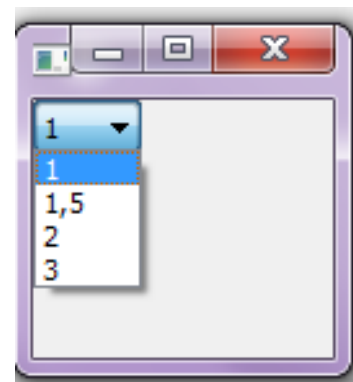


## 2. Вставка елемента:

```
void insertItem (int index, const  
QString &text, const QVariant  
&data=QVariant()),
```

index – позиція елемента, що вставляється (0), text -  
текст елемента.

```
cb->insertItem(1, "1, 5");
```



# Методи і слоти класу QComboBox



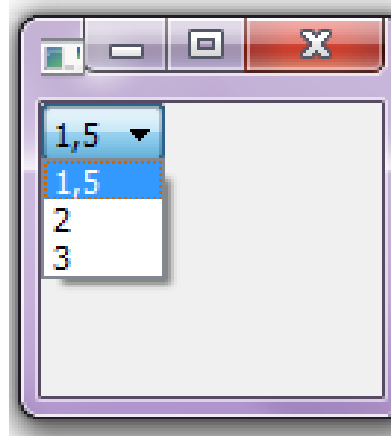
## 3. Видалення елемента :

```
void removeItem (int index, const  
QString &text),
```




index – позиція елемента, що видаляється (0).

```
cb->insertItem(0);
```



# Методи і слоти класу QComboBox




 4. `int count ()` – повертає кількість елементів в списку.

 `int n=cb->count (); //n=3`

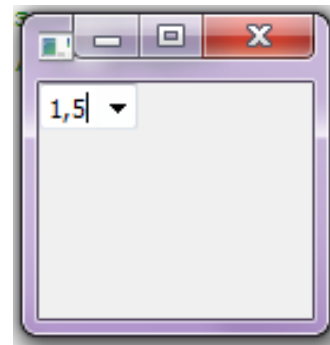
5. `void setEditable (bool editable)` – дозволяє / забороняє редагування тексту в редакторі комбінованого списку.

 `cb->setEditable (true); // дозволяє`

`cb->setEditable (false); // забороняє`

 Після редагування та натискання клавіші<Enter>, новий елемент

додається в список, а старий не змінюється



# Методи і слоти класу QComboBox



6. `void setMaxVisibleItems (int max)` – обмежує кількість елементів в випадяючому вікні списку - може з'явитися смуга прокрутки.

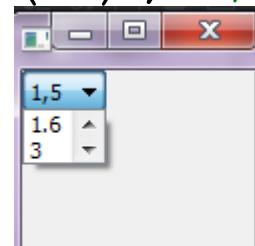
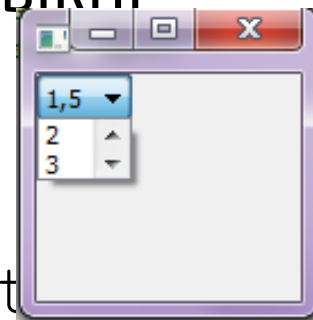
`cb->setMaxVisibleItems (2);`

7. `void setItemText (int index, const QString &text)` – зміна тексту елемента, де `index` - індекс елемента, `text` - новий текст.

8. `QString itemText (int index)` - зчитування тексту елемента, де `index` - індекс елемента.

`QString str1=cb->itemText (1); //str="2«`

`cb->setItemText (1, "1.6");`



# Методи і слоти класу QComboBox



9. `int currentIndex()` – повертає індекс вибраного елементу списку.

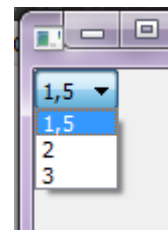
10. `QString currentText()` – повертає текст вибраного елемента.

```
QString str2=cb->currentText(); //str2="3"
```

```
int i=cb->currentIndex(); //i=2
```

11. `int findText(QString &text)` – шукає елемент з текстом `text` і в разі успіху повертає його індекс, інакше повертає -1.

```
int n=cb->findText("2"); //n=1
```



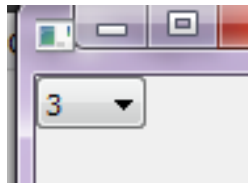


# Методи і слоти класу QComboBox



12. `void setCurrentIndex (int index)` – робить поточним (обраним) елемент списку

`cb->setCurrentIndex (0) ;`



13. `void setMaxCount (int max)` – задає максимально можливу кількість елементів у списку

`cb->setMaxCount (1) ;`



# Сигнали класу QComboBox



1. Висилаються, коли користувач вибрав новий елемент списку (`index`, `text` – індекс нового елементу)



```
void activated (int index)
```

```
void activated (const QString &text)
```



2. Висилаються, коли програмно або користувачем обраний новий елемент (`index`, `text` - індекс нового елементу)



```
void currentIndexChanged (int index)
```

```
void currentIndexChanged (const QString  
&text)
```



# Сигнали класу QComboBox



3. Висилаються, коли користувач «перебирає» елементи в списку (`index`, `text` – індекс нового елементу)




```
void highlighted(int index)
```


```
void highlighted(const QString &text)
```



# Клас QListWidget

A small icon of a network or tree structure.

Призначений для створення об'єкта «звичайного» списку, елементами списку можуть бути текстові рядки, іконки, прапорці та інші віджети.

A small icon of a checklist with a green checkmark.

Конструктор:

A small icon of a computer monitor.

`QListWidget (QWidget *parent=0)`,  
де `parent` – покажчик на батьківський об'єкт.



# Методи і слоти класу QListWidget

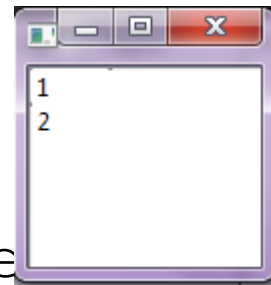


1. `void addItem (const QString &text)` – додає в список новий елемент з текстом `text`.

```
QListWidget *lw=new QListWidget(this);
```

```
lw->addItem("1");
```

```
lw->addItem("2");
```

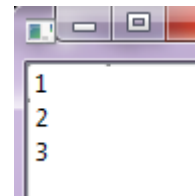


2. `void addItem (const QListWidgetItem *item)` – додає новий елемент в список, `item` – вказує на новий елемент.

Клас `QListWidgetItem` призначений для створення елемента списку. Може зберігати в собі як рядок тексту, так і елементи графічного інтерфейсу (прапорці, текстові редактори і т.д.)

```
QListWidgetItem *item = new QListWidgetItem  
("3", lw);
```

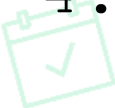
```
lw->addItem(item);
```



# Методи і слоти класу QListWidget



3. `int count()` – повертає кількість елементів в списку.




4. `int currentRow()` – повертає індекс поточного елемента списку.



5. `void setCurrentRow(int index)` – робить поточним новий елемент в списку з індексом `index`.




# Методи і слоти класу QListWidget



6. `QListWidgetItem *takeItem (int row)` – витягує (видаляє) елемент з індексом `row` зі списку. Значення, що повертається - покажчик на витягнутий елемент.



7. `void insertItem(int row, QListWidgetItem *item)` – вставляє елемент `item` на задану (`pos`) позицію.



```
int pos=0; // позиція елемента, текст
           // которого будем менять
```

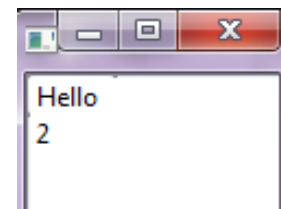
```
QListWidgetItem *item=lw->takeItem(pos);
```



```
item->setText ("Hello");
```

```
lw->insertItem(pos,item);
```

8. `void clear()` – видаляє всі елементи зі списку



# Сигнали класу QListWidget



1. `void itemClicked (QListWidgetItem *item)` – висилається при натисканні на елементі списку, `item` - вказує на «клацнути» елемент.



2. `void currentRowChanged(int currentRow)` – висилається, коли новий елемент став поточним (обраним), `currentRow` - індекс нового поточного елемента





# Клас QGroupBox



Призначений для створення об'єкта контейнера для розміщення в ньому інших віджетів.



Конструктор:

```
QGroupBox (const QString &title, QWidget  
*parent=0) ,
```



де `title` – задає текст заголовка контейнера, `parent` - показчик на батьківський об'єкт.



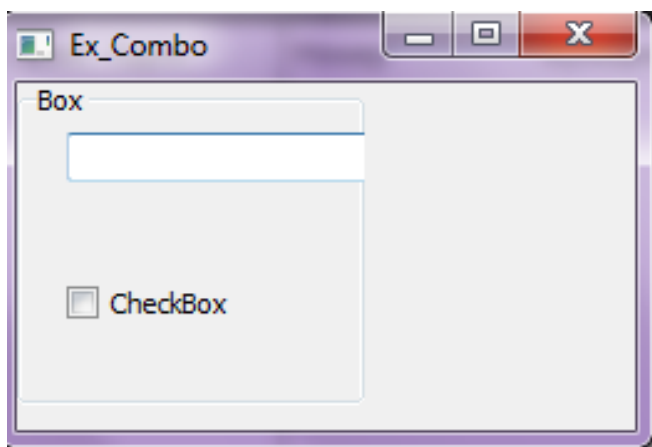
Для того, щоб віджети при створенні автоматично поміщалися в контейнер, необхідно в якості батьківського віджета для них вказувати даний контейнер.



# Клас QGroupBox



```
QGroupBox *box=new QGroupBox ("Box", this);  
QLineEdit *edit=new QLineEdit (box);  
edit->move (20, 20);  
QCheckBox *chbx=new QCheckBox  
("CheckBox", box);  
chbx->move (20, 80);
```



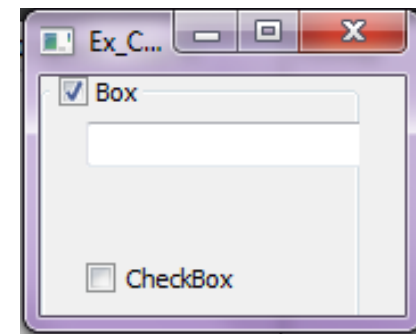
# Методи класу QGroupBox



1. `void setCheckable (bool checkable)` – розміщує / видаляє прапорець в заголовку контейнера



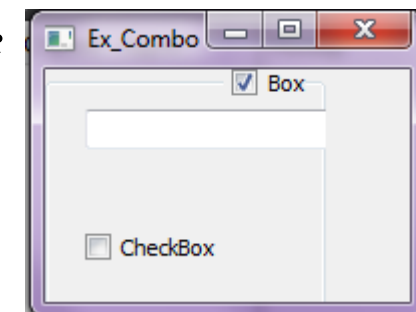
```
box->setCheckable (true) ;
```



2. `void setAlignment (Qt::Align align)` – управляє розміщенням (по горизонталі) прапорця і заголовка



```
box->setAlignment (Qt::AlignRight) ;
```



# Клас QSlider



Призначений для створення віджета бігунка.

Конструктор:

```
QSlider (Qt::Orientation orientation,  
QWidget *parent=0),
```

orientation – задає орієнтацію бігунка, parent –  
показчик на батьківський віджет.

orientation^:

Qt::Horizontal – по горизонталі

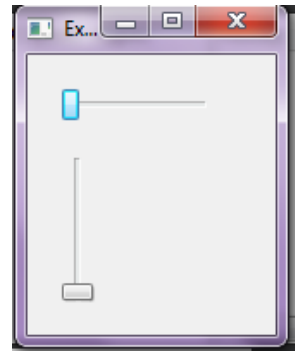
Qt::Vertical – по вертикалі

```
QSlider *s11=new QSlider(Qt::Horizontal,  
this);
```

```
s11->move(20, 20);
```

```
QSlider *s12=new QSlider(Qt::Vertical, this);
```

```
s12->move(20, 60);
```



# Методи та слоти класу QSlider



1. `void setTickPosition`  
(`QSlider::TickPosition position`) –  
управляє розміщенням «поділів» бігунка, за  
замовчуванням не відображаються.

`s1->setTickPosition(QSlider::TicksBothSides);`

Position:

`QSlider::NoTicks`

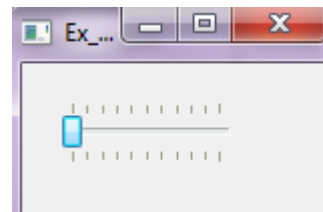
`QSlider::TicksBothSides`

`QSlider::TicksAbove`

`QSlider::TicksBelow`

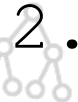



`QSlider::TicksLeft`

`QSlider::TicksRight`







# Методи та слоти класу QSlider

Qt

-  2. `void setSingleStep (int step)` – задає величину, на яку буде змінюватися значення (позиція) бігунка при натисканні на кнопки
-  3. `void setPageStep (int step)` – аналогічно при «сторінковому» перегляді
-  4. `void setRange (int min, int max)` – задає діапазон [min; max] значень бігунка
5. `void setValue (int )` – задає значення бігунка
-  6. `void setTracking (bool enable)` – якщо `enable = true`, сигнал бігунка при зміні його позиції буде надсилатися автоматично при переміщенні; якщо `enable=false`, сигнал бігунка про зміну його позиції буде висланий, коли користувач перемістить бігунок і відпустить його.

# Сигнали класу QSlider




-  1. `void valueChanged (int value)` – висилається при зміні позиції (значення) бігунка, `value` – містить нове значення.
-  2. `void sliderPressed ()` the user starts to drag the slider.
3. `void sliderMoved ()` the user drags the slider.
-  4. `void sliderReleased ()` the user releases the slider.
-  5. `void actionTriggered ()` a slider action was triggered.
6. `void rangeChanged ()` a the range has changed.



# Клас QLCDNumber



 Призначений для створення об'єкта семисегментний індикатора (як в електронному годиннику) для відображення числової інформації.



Конструктор:

 `QLCDNumber (uint numDigits, QWidget *parent=0),`

де `numDigits` – кількість відображуваних цифр, `parent` - покажчик на батьківський віджет.





# Методи та слоти класу QLCDNumber



1. `void display (const QString &num),`  
`void display (int num), void display`  
`(double num)` – задають відображене число.

```
QLCDNumber *lcd=new QLCDNumber(5,this);  
lcd->display(20.3);
```

2. `void setMode (Mode)` – встановлює систему  
числення (за замовчуванням - Dec).

Mode :

`QLCDNumber::Hex` - 16

`QLCDNumber::Dec` - 10

`QLCDNumber::Oct` - 8

`QLCDNumber::Bin` - 2



# Методи та слоти класу QLCDNumber



3. `void setSegmentStyle (Segment style)` – задає стиль цифр індикатора

style:



`QLCDNumber::Outline` – об'ємні кольору фону

`QLCDNumber::Filled` – об'ємні чорного (`windowText color`) кольору

`QLCDNumber::Flat` – плоскі чорного (`windowText color`) кольору



Сигнал:



`void overflow()` – висилається, коли для відображення числа не вистачає розрядів.

