Об’єктно орієнтоване програмування

Лабораторна робота №2

Можливості графічних додатків

При створенні фреймворку QT широко використовується принципи ООП, а всі класи створюють певну ієрархію (рис. 2.1).

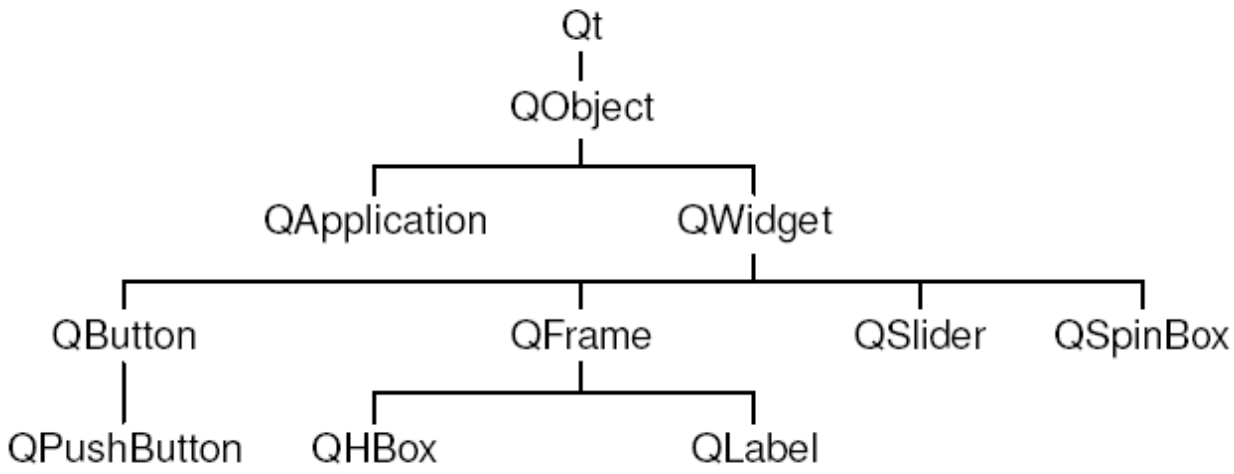

Рисунок 2.1 – Спрощена ієрархія класів QT

Клас **QWidget** є базовим для всіх класів віджетів Qt та має відповідний конструктор:

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

де parent – покажчик на батьківський віджет,

f – задає вигляд (тип) вікна.

якщо parent == 0 – буде створено вікно верхнього рівня, з заголовками і системним меню;

якщо parent != 0 та вказує на конкретний віджет, то нове вікно буде створено на його поверхні.

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

Метод `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
```

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()` і `int y()` повертають відповідно горизонтальну і вертикальну координати віджета;

Метод `void setX (int x), void setY (int y)` – задають нові координати віджета;

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);
```

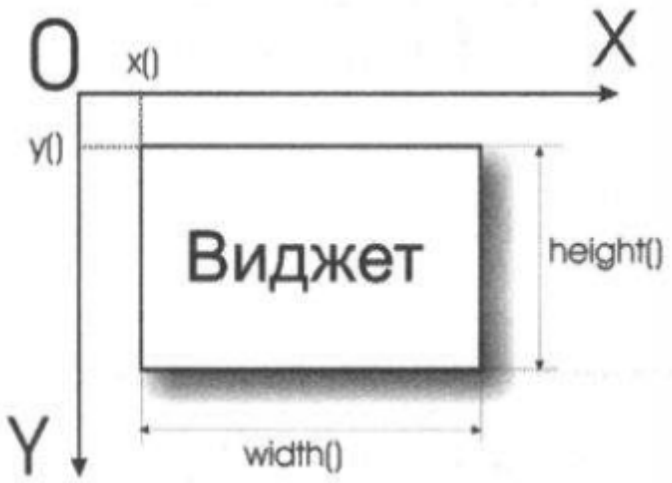

Рисунок 2.2 – Система координат віджету

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

Віджет QLabel служить для показу стану додатка або пояснення і являє собою текстове поле, текст якого не підлягає зміні з боку користувача, та має відповідний конструктор

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

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

Слот void setAligment (Qt :: Aligment) – управляє розташуванням тексту в поле мітки;

Qt :: AlignHCenter | Qt :: AlignVCenter – по центру по горизонталі і по вертикалі;

Qt :: AlignTop | Qt :: AlignLeft – у верхньому лівому кутку мітки.

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 **QCheckBox** використовується для введення булівих даних від користувача, та має відповідний конструктор:

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

де text - задає текст прапорця,

parent - покажчик на батьківський віджет.

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

Може мати 2 або 3 стану

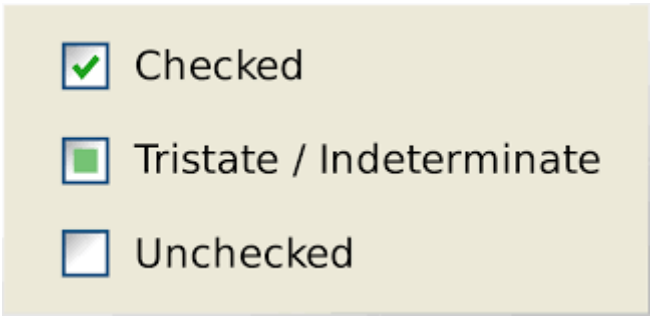

Рисунок 2.3 – Можливий вигляд QCheckBox

`void setChecked(bool)` – встановити стан прапорця

true – встановити

false – скинути

`bool isChecked()` – перевірити стан прапорця

true – встановлений
false – скинутий
`void toggled (bool)` – при зміні стану прапорець висилає сигнал
true – прапорець включений
false – прапорець вимкнений

Прапорець з 3 станами
`void setTristate (bool y = true)` – для перетворення звичайного прапорця (2 стану) в прапорець з 3 станами
y = true – три стану;
y = false – два стану;
`bool is Tristate ()` – 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 вимкнений;
Qt :: Checked – включений (установлений);

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);

Лабораторна робота №2 (3-й семестр)

Варіанти 1 – 9.

Написати програму, в якій:

1. Реалізовано клас, що описує функцію згідно варіанту (Таблиця 2.1), де *a*, *b*, *c* дійсні числа, які задає користувач.
2. Для вводу параметрів *a*, *b*, *c* використовуються компонент *Double spin box.*
3. Користувач має можливість обчислити значення функції для введеного значення *x*.
4. Після натискання на кнопку розрахувати, кнопка стає не активною (Disabled) і залишається такою поки користувач не натисне на кнопку «очистити результат». Таку ж поведінку демонструють поля для вводу вхідних параметрів.
5. На формі розміщені 3 checkbox, якщо вони відмічені то на формі при розрахунку з'являться (Visible) відповідні поля для виводу модулю значення, квадрату та кубу результату обчислення функції.
6. Поєднання сигналів і слотів відбувається вручну за допомогою функції connect().

Варіант * (Додаткове завдання за додаткові бали)

1. Модифікувати клас та форму таким чином, щоб можна було додати в кінець необмежену кількість проміжків значень функції виду $k\,x+d$ та відповідних проміжків їх обчислення.

Таблиця 2.1 – Функції для реалізації

1	$y=\begin{cases} a\sin(x^2), & x<0 \\ b\cos(x) & 0\le x\le 0,3 \\ c\ln(x) & x>0,3 \end{cases}$	6	$y=\begin{cases} a\lvert x\rvert, & x<-1 \\ b\sin(x)+5\cos(x) & -1\le x\le 3 \\ c\ln(x) & x> \end{cases}$
2	$y=\begin{cases} 7x^4-2x^2+4a, & x<-8 \\ \sin(x)+5b\cos(x) & -8\le x\le 6 \\ c\lg(x) & x>6 \end{cases}$	7	$y=\begin{cases} 7x^4+3x^2-a, & x<2 \\ 0.5\exp(bx)/x & 2\le x\le 4 \\ c\log_2(x) & x>4 \end{cases}$
3	$y=\begin{cases} a\exp(x), & x<-2 \\ bx+3 & -2\le x\le 2 \\ \sqrt[3]{cx} & x>2 \end{cases}$	8	$y=\begin{cases} a5,3^x, & x<2 \\ \exp(bx)+3 & 2\le x\le 4 \\ 2\mathrm{tg}^2(cx)-4 & x>4 \end{cases}$
4	$y=\begin{cases} \lvert ax\rvert/(8x^2+2), & x<4 \\ \sqrt[3]{bx+8} & 4\le x\le 5 \\ \exp(cx)-\pi & x>5 \end{cases}$	9	$y=\begin{cases} 0.5/\lvert ax\rvert, & x<-2 \\ x^2+12x+4b & -2\le x\le 0 \\ 3c\ln(x) & x>0 \end{cases}$
5	$y=\begin{cases} a\sin(2x), & x<-1,5 \\ 2b/\exp(x) & -1,5\le x\le 0 \\ \sqrt{c^2x^2+1} & x>0 \end{cases}$	10	$y=\begin{cases} a\cos(2x), & x<-1 \\ b\exp(x) & -1\le x\le 0 \\ x^2+c & x>0 \end{cases}$