

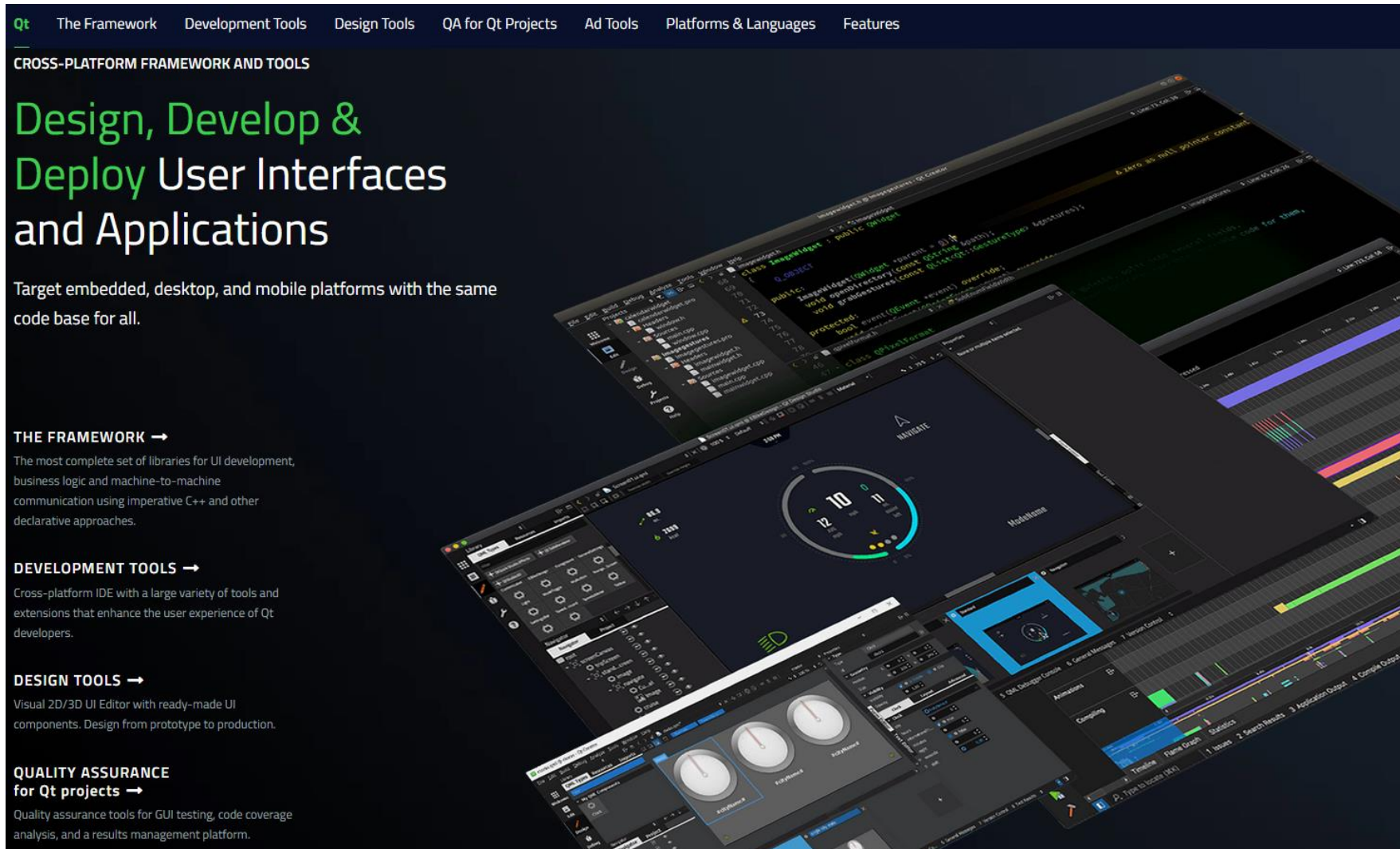
Qt

*Code less, create more, deploy everywhere.*

1.26

# Qt

biblioteki i narzędzia do budowy m.in. aplikacji desktoowych oraz mobilnych dla środowisk Windows, Linux , Mac OS, Android, iOS



**Qt** The Framework Development Tools Design Tools QA for Qt Projects Ad Tools Platforms & Languages Features

## CROSS-PLATFORM FRAMEWORK AND TOOLS

# Design, Develop & Deploy User Interfaces and Applications

Target embedded, desktop, and mobile platforms with the same code base for all.

**THE FRAMEWORK** →  
The most complete set of libraries for UI development, business logic and machine-to-machine communication using imperative C++ and other declarative approaches.

**DEVELOPMENT TOOLS** →  
Cross-platform IDE with a large variety of tools and extensions that enhance the user experience of Qt developers.

**DESIGN TOOLS** →  
Visual 2D/3D UI Editor with ready-made UI components. Design from prototype to production.

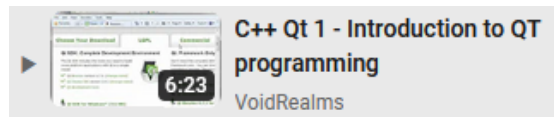
**QUALITY ASSURANCE for Qt projects** →  
Quality assurance tools for GUI testing, code coverage analysis, and a results management platform.

<https://www.qt.io/product>

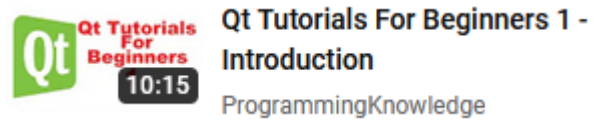
# tutorial



<https://forbot.pl/blog/kurs-qt-1-czym-jest-qt-pierwsza-aplikacja-w-praktyce-id35549>



<https://www.youtube.com/watch?v=6KtOzh0StTc&list=PL2D1942A4688E9D63&index=1>



<https://www.youtube.com/watch?v=EkjaiDsiM-Q&list=PLS1QulWo1RIZiBcTr5urECberTITj7gJA>

# documentation

The screenshot shows a web browser window with the address bar displaying `doc.qt.io/qt-6/classes.html`. The page content includes a navigation menu on the left with sections for 'Qt 6.4', 'Reference', 'Getting Started', and 'Overviews'. The 'Reference' section is expanded to show 'All Qt C++ Classes' and other links. The main content area features a breadcrumb trail 'Qt 6.4 > All Classes', a large heading 'All Classes', and a list of links: 'All Functions', 'All Namespaces', 'All Classes by Module', 'Obsolete Classes', and 'New Classes and Functions'. A partial sentence 'For more reference pages including !' is visible at the bottom of the main content area.

Qt All Classes | Qt 6.4

doc.qt.io/qt-6/classes.html

< Back to Qt.io

Qt DOCUMENTATION

Qt 6.4

Qt 6.4 > [All Classes](#)

## All Classes

This is a list of all Qt classes. The foll

- > [All Functions](#)
- > [All Namespaces](#)
- > [All Classes by Module](#)
- > [Obsolete Classes](#)
- > [New Classes and Functions](#)

For more reference pages including !

<https://doc.qt.io/qt-6/classes.html>

# download

## Download Qt for open source use

Find out how you can use Qt under the (L)GPL and contribute to the Qt project.

[Go open source](#)

[View Qt product map](#)

<https://www.qt.io/download-open-source>



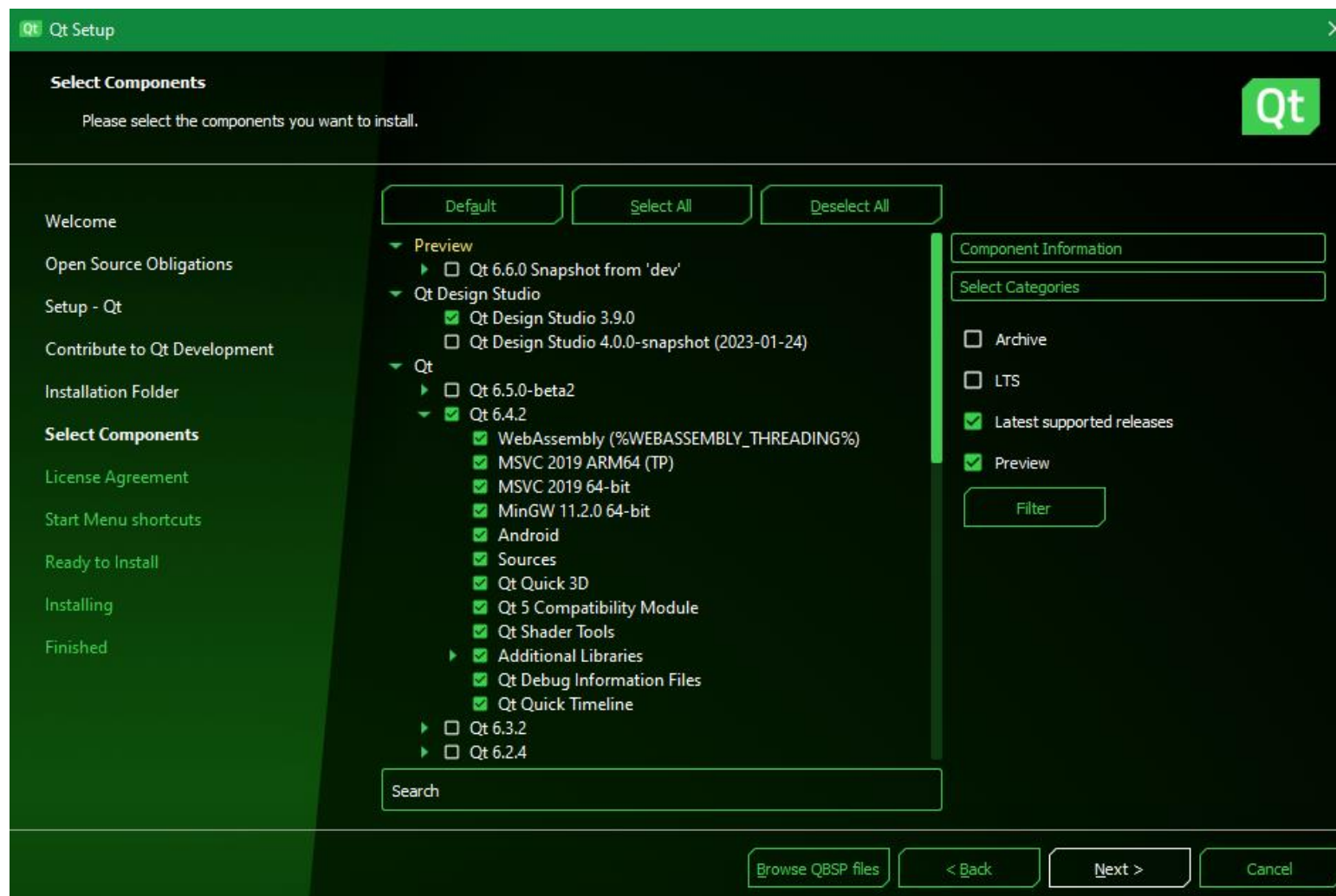
## Looking for Qt binaries?

Find them in the Qt Online Installer. It will steer you to the right download version and help you install tools and add-on components that are available for your open source license.

[Download the Qt Online Installer](#)



# download



# Qt Creator

mainwindow.ui @ helloWorld - Qt Creator

Plik Edycja View Budowanie Debugowanie Analiza Narzędzia Okno Pgmoc

**Powitanie**

## Welcome to Qt Creator

Poszukaj w samouczkach...

Create Project... Open Project... Nowicjusz? Get Started

Projekty Przykłady **Samouczki** Marketplace

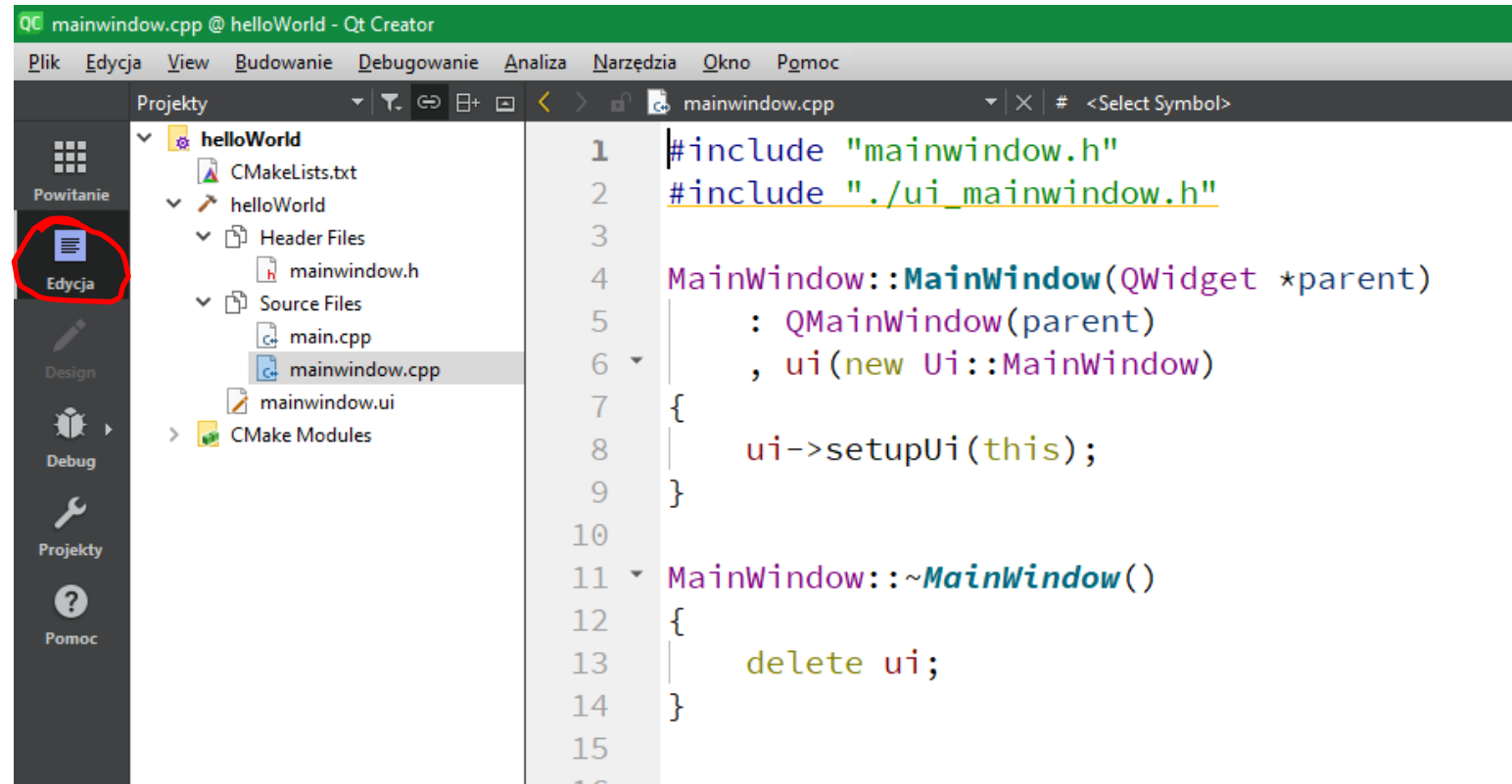
helloWorld Debug

Get Qt Konto Qt Społeczność online Blogi Przewodnik użytkownika

Tutorial Title	Duration	Tags
Help: Building and Running an Exa...		qt creator build compile help
Help: Creating a Mobile Qt Applica...		qt creator qml android ios controls help
Help: Creating a Qt Quick Applicati...		qt creator qt quick qml states transitions help
Help: Creating a Qt Widget-Based ...		qt creator qt designer widgets c++ text help
Help: Getting Started Programming...		qt quick controls tumblr help
Help: Getting Started Programming...		qt qt creator qt designer widgets c++ help
Online: Creating a simple Qt Quick ...	8:21	qt creator qt quick 2021
Online: Creating a simple widget app	6:08	qt creator widgets 2021
Online: Debugging inside Qt Creator	21:54	qt creator debugging 2021
Online: How to build your first 'Qt f...	21:54	qtformcus mcus qt video NXIP IMXRT1050-EVKB 2020
Online: How to create a simple app...	5:16	qtformcus mcus qt video 2021
Online: How to do translations with Qt Linguist		qt creator qt linguist translation 2021
Online: How to install and set up Q...	6:52	embedded installation device creation
Online: How to install and set up Q...	8:29	qt mcus video STM32H750B-DISCOVERY
Online: How to set up and deploy ...	5:48	qt creator embedded device creation
Online: Qt Creator - Examples	9:29	qt creator video 2018
Online: Qt Creator - Introduction to...	7:09	qt creator qt quick controls video 2018
Online: Qt Creator - Meet Qt Creator	2:06	qt creator video

strona z tutorialami, przykładami

# Qt Creator



edytor

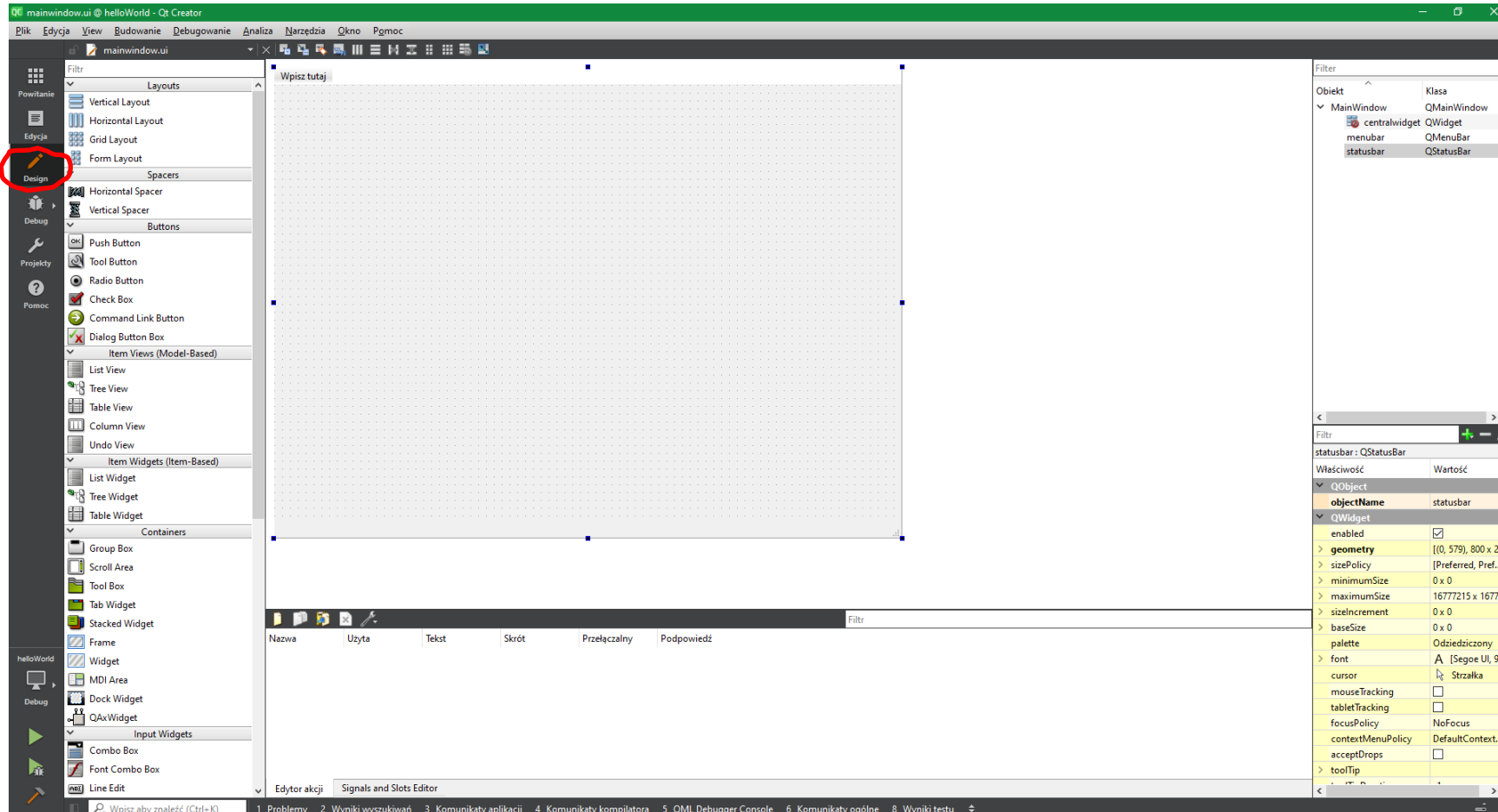


# Qt Creator

Projekty

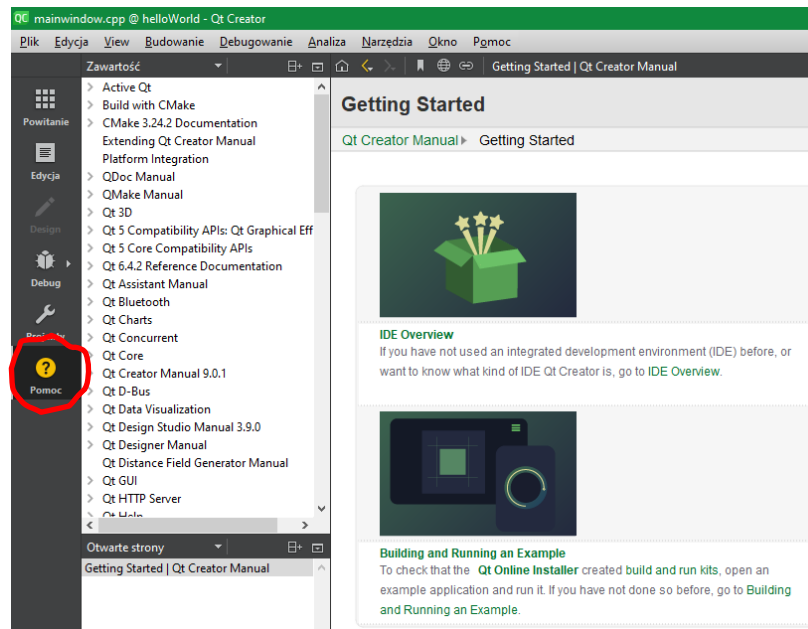
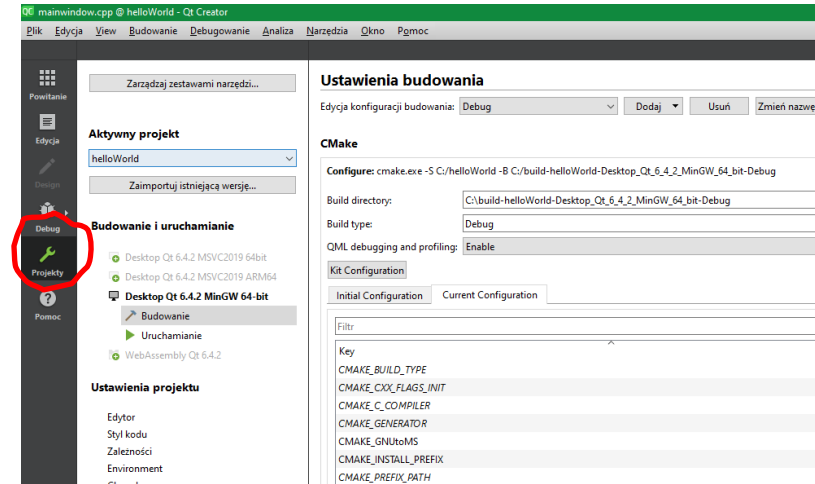
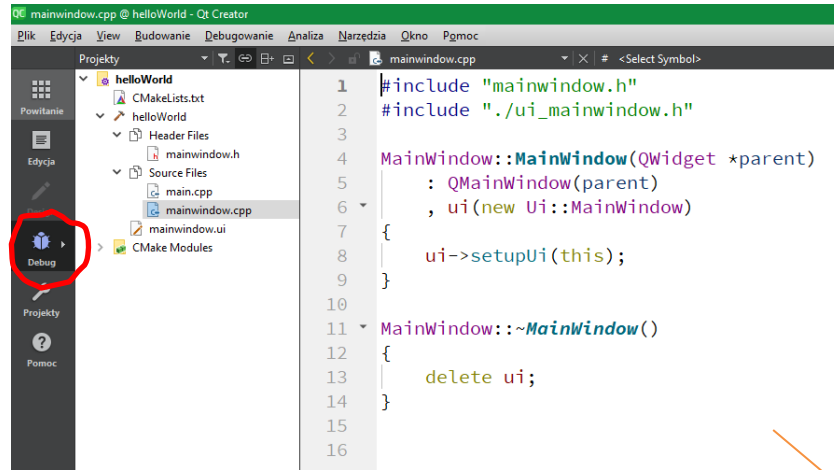
- ▼ **helloWorld**
  - ▼ **CMakeLists.txt** — plik konfiguracyjny programu CMake zarządzającego kompilacją programu
  - ▼ **helloWorld**
    - ▼ **Header Files** — plik nagłówkowy klasy MainWindow
      - mainwindow.h
    - ▼ **Source Files** — plik z funkcją main()
      - main.cpp
      - mainwindow.cpp — plik z klasą MainWindow głównego okna aplikacji
    - mainwindow.ui — plik XML interfejsu graficznego (podwójne kliknięcie na pliku powoduje przejście do zakładki Design)
  - > **CMake Modules**

# Qt Creator



narzędzie do projektowania interfejsu

# Qt Creator

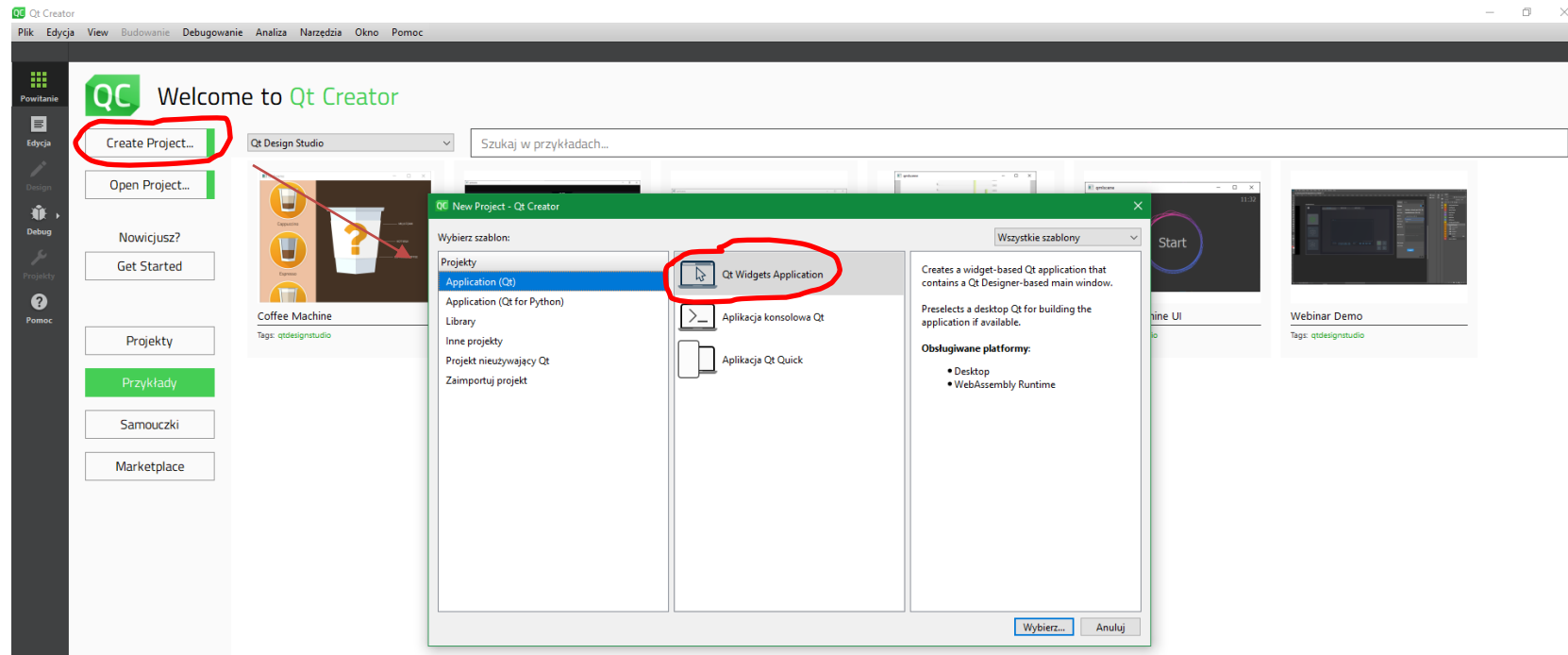


debugger  
(usuwanie błędów)

konfiguracja projektu

źródła pomocy

# new project



# new project

Qt Widgets Application

Położenie projektu

This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

Położenie

- System budowania
- Szczegóły
- Translation
- Zestawy narzędzi
- Podsumowanie

Nazwa:

Utwórz w:

Ustaw jako domyślne położenie projektów

Qt Widgets Application

Zdefiniuj system budowania

System budowania:

Położenie

- System budowania
- Szczegóły
- Translation
- Zestawy narzędzi
- Podsumowanie

Qt Widgets Application

Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Położenie

- System budowania
- Szczegóły
- Translation
- Zestawy narzędzi
- Podsumowanie

Nazwa klasy:

Klasa bazowa:

Plik nagłówkowy:

Plik źródłowy:

Generate form

Form file:

Qt Widgets Application

Translation File

If you plan to provide translations for your project's user interface via the Qt Linguist tool, please select a language here. A corresponding translation (.ts) file will be generated for you.

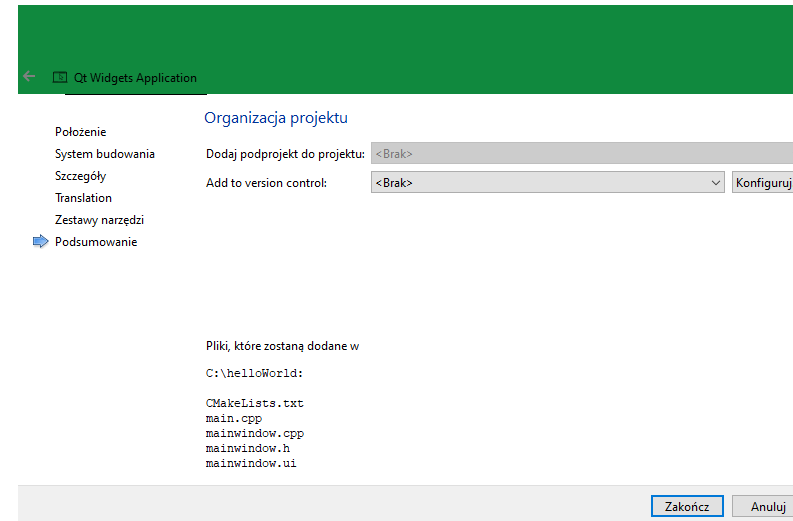
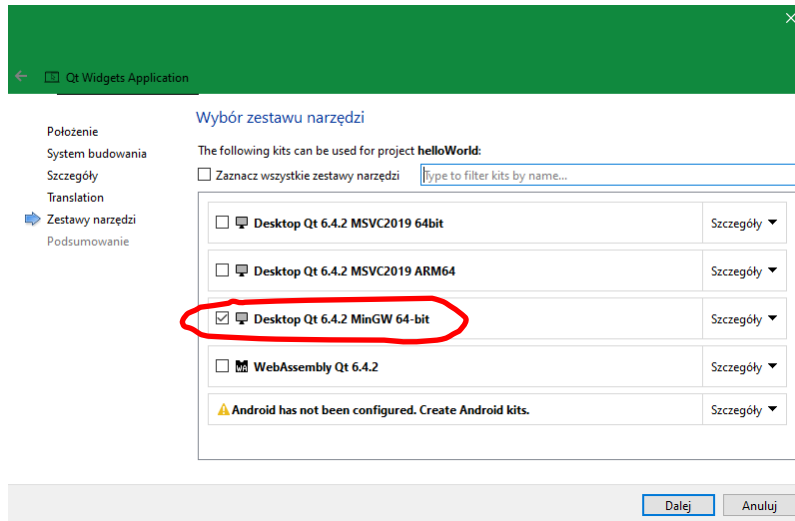
Położenie

- System budowania
- Szczegóły
- Translation
- Zestawy narzędzi
- Podsumowanie

Language:

Translation file:

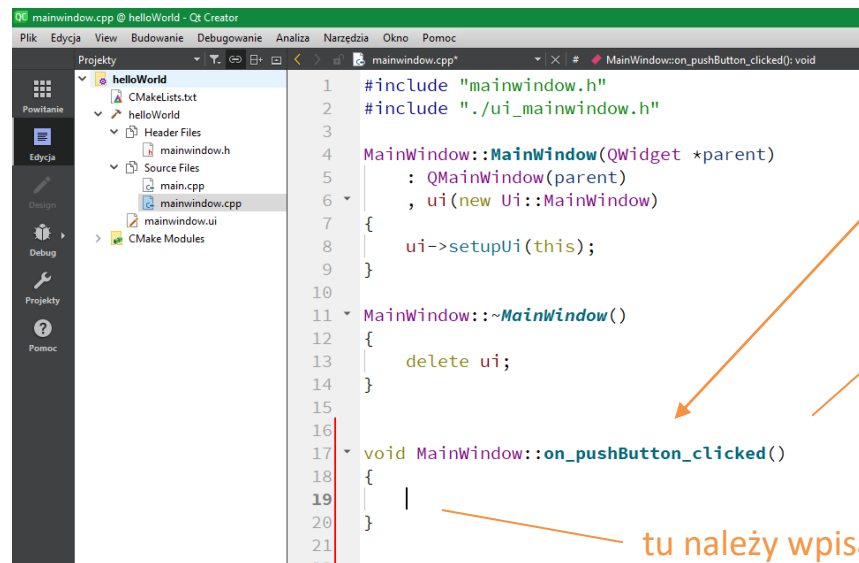
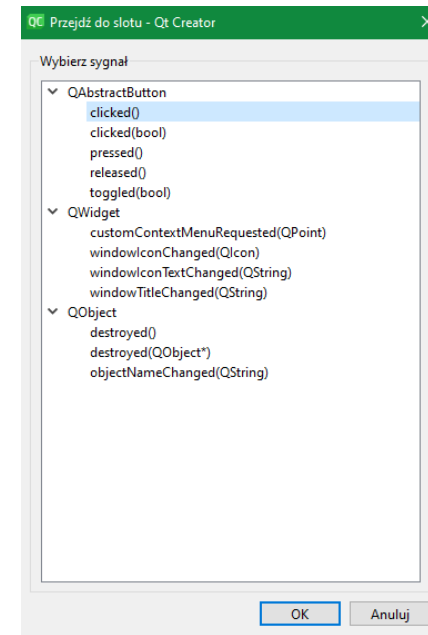
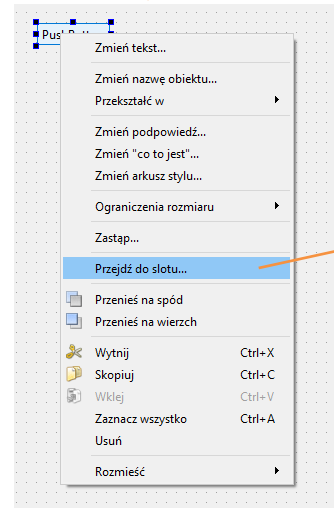
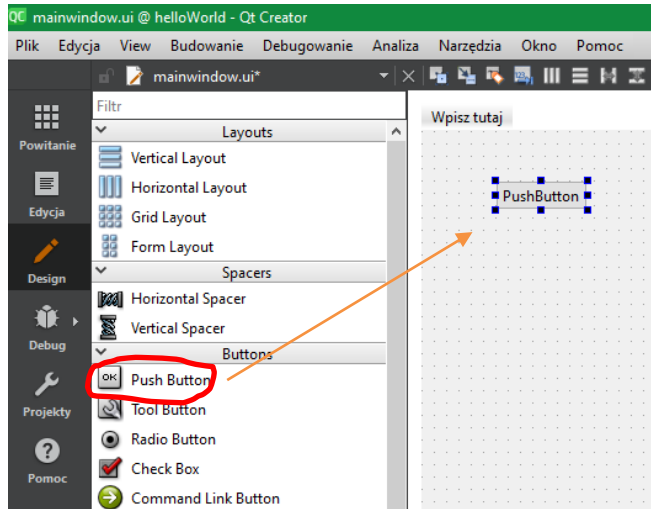
# new project



# helloWorld

łączymy sygnał przycisku `click()`  
ze slotem okna

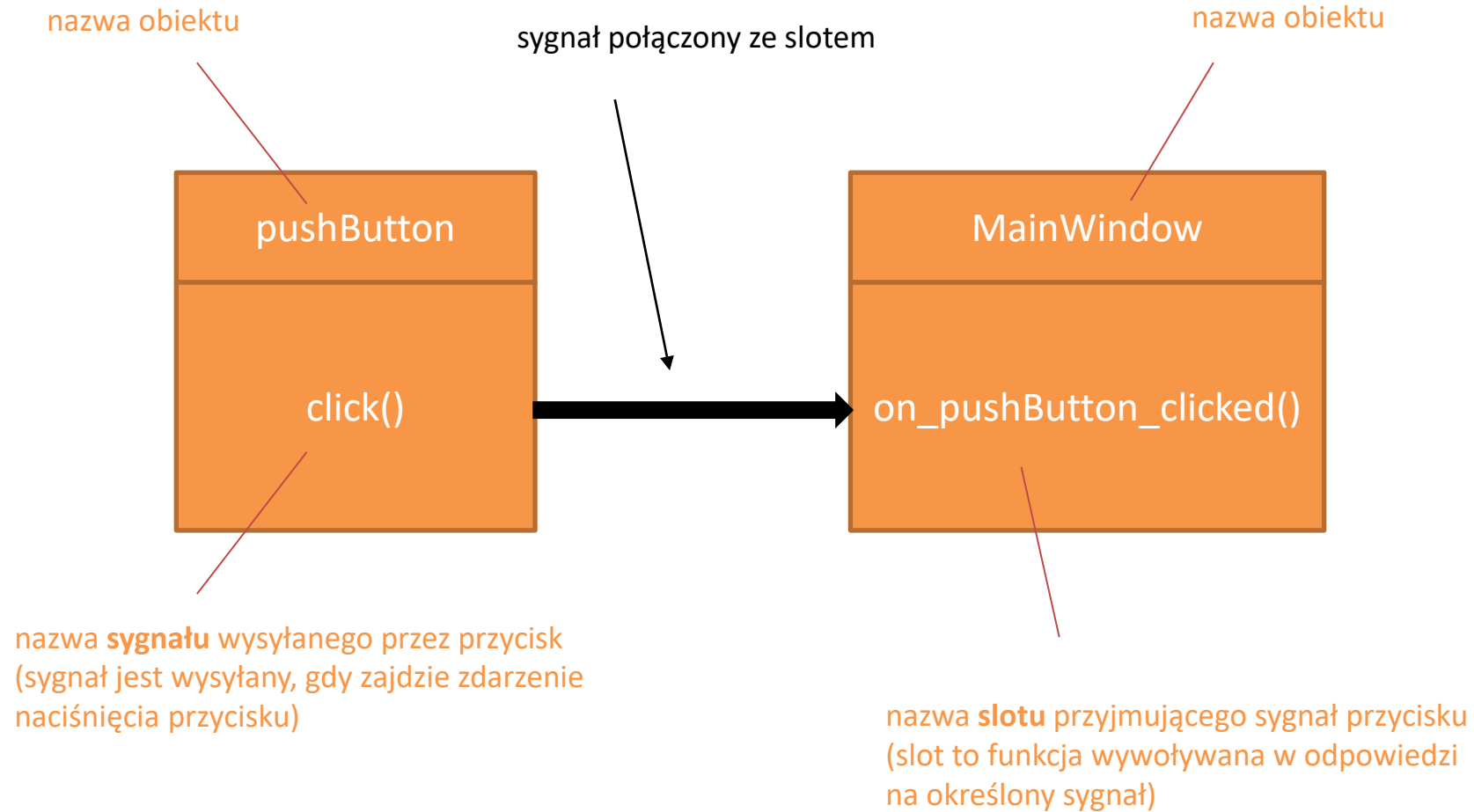
prawy przycisk myszy na buttonie



implementacja slotu okna  
(metoda obsługująca kliknięcie przycisku)

tu należy wpisać kod przycisku

# helloWorld





# helloWorld

```
mainwindow.cpp @ helloWorld - Qt Creator
Plik  Edycja  View  Budowanie  Debugowanie  Analiza  Narzędzia  Okno  Pomoc
Projekt  mainwindow.cpp*  MainWindow::on_pushButton_clicked(): void
helloWorld
  CMakeLists.txt
  helloWorld
    Header Files
      mainwindow.h
    Source Files
      main.cpp
      mainwindow.cpp
      mainwindow.ui
  CMake Modules
1  #include "mainwindow.h"
2  #include "../ui/mainwindow.h"
3  #include <QDebug>
4
5  MainWindow::MainWindow(QWidget *parent)
6      : QMainWindow(parent)
7      , ui(new Ui::MainWindow)
8  {
9      ui->setupUi(this);
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     qDebug() << "Wcisnąłeś przycisk!";
21 }
22
23
```

kod przycisku (slot)

komunikat wypisze się na konsoli

# helloWorld

mainwindow.cpp @ helloWorld - Qt Creator

Plik Edycja View Budowanie Debugowanie Analiza Narzędzia Okno Pomoc

Projekt: helloWorld

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QDebug>
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     qDebug() << "Wcisnął";
21 }
22
23
```

Open Documents: CMakeLists.txt, main.cpp, mainwindow.cpp\*, mainwindow.h\*, mainwindow.ui\*

uruchomienie aplikacji (lub CTR + R)

Zachowaj zmiany - Qt Creator

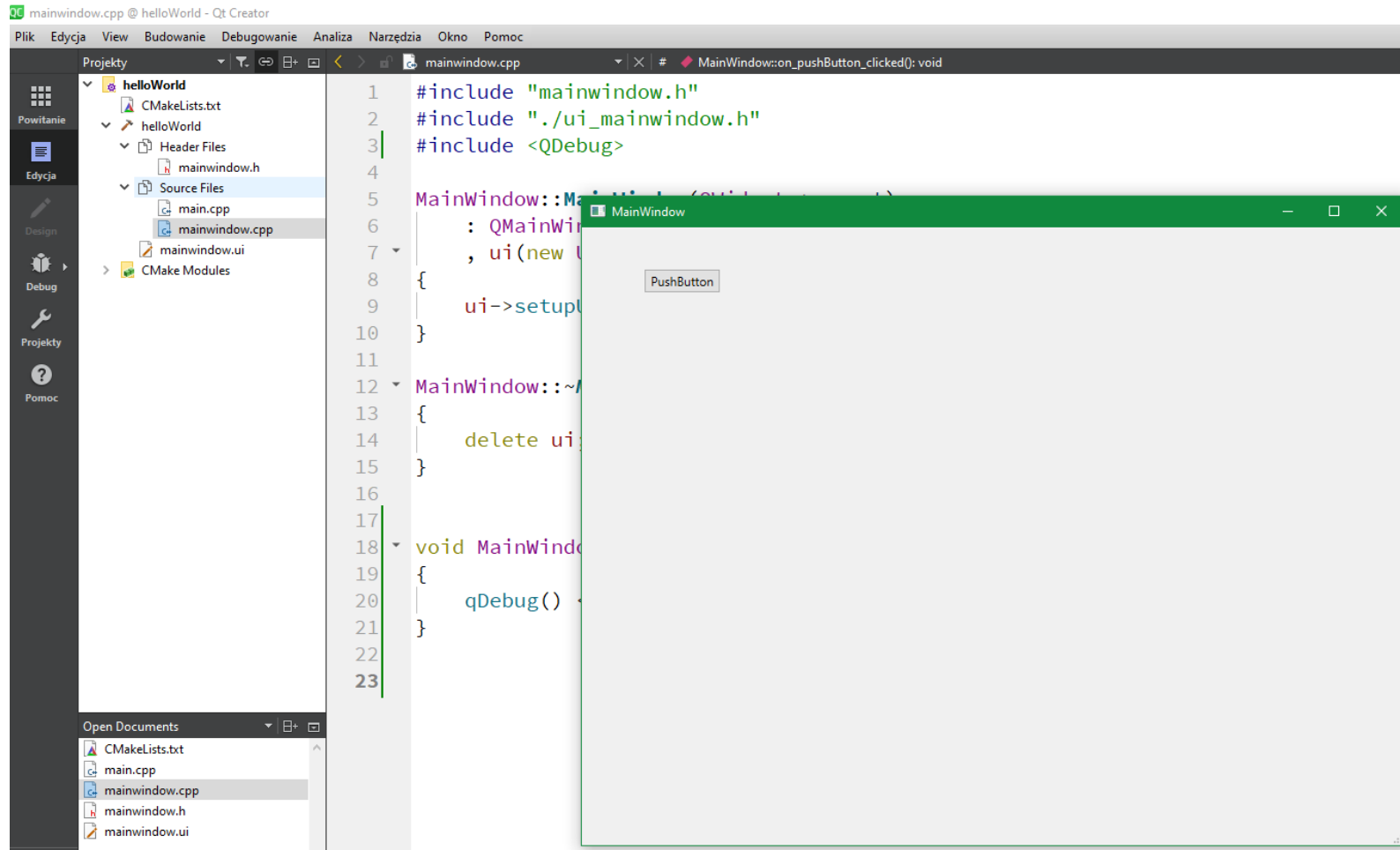
Następujące pliki posiadają niezachowane zmiany:

- mainwindow.cpp C:\helloWorld
- mainwindow.h C:\helloWorld
- mainwindow.ui C:\helloWorld

Zawsze zachowuj pliki przed budowaniem

Zachowaj wszystko Pokaż różnice we wszystkich i anuluj Nie zachowuj Anuluj

# helloWorld



okno aplikacji

# helloWorld

The image shows the Qt Creator IDE interface for a project named 'helloWorld'. The main window displays a 'PushButton' button. The console shows the execution of the application, with a red circle highlighting the message 'Wcisnales przycisk!' (You pressed the button!).

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QDebug>
4
5 MainWindow::MainWindow(QWidget *parent)
6 : QMainWindow(parent), ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     qDebug() << "Wcisnales przycisk!";
21 }
22
23
```

Komunikaty aplikacji

helloWorld x

15:36:06: Debugging C:\build-helloWorld-Desktop\_Qt\_6\_4\_2\_MinGW\_64\_bit-Debug\helloWorld.exe...

15:37:12: Debugging of C:\build-helloWorld-Desktop\_Qt\_6\_4\_2\_MinGW\_64\_bit-Debug\helloWorld.exe has finished with exit code 0.

15:55:03: Uruchamianie C:\build-helloWorld-Desktop\_Qt\_6\_4\_2\_MinGW\_64\_bit-Debug\helloWorld.exe...

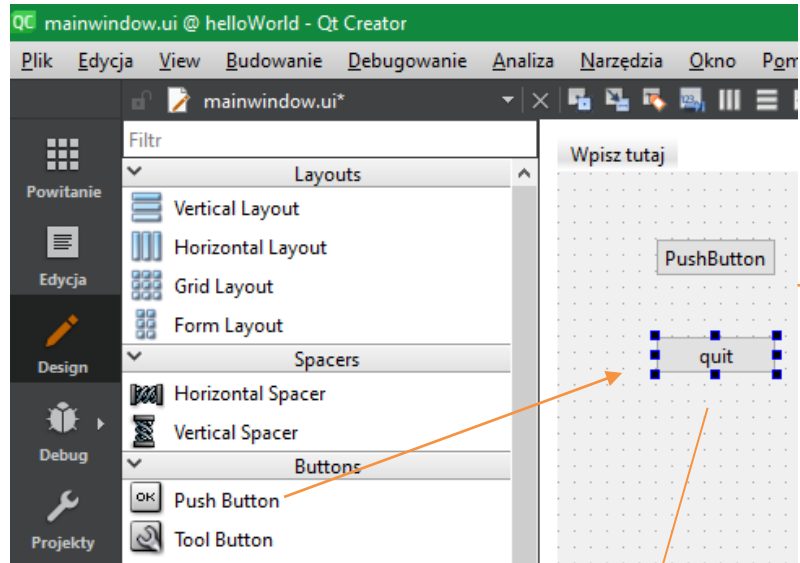
15:55:10: C:\build-helloWorld-Desktop\_Qt\_6\_4\_2\_MinGW\_64\_bit-Debug\helloWorld.exe exited with code 0

16:18:45: **Uruchamianie C:\build-helloWorld-Desktop\_Qt\_6\_4\_2\_MinGW\_64\_bit-Debug\helloWorld.exe...**  
**Wcisnales przycisk!**

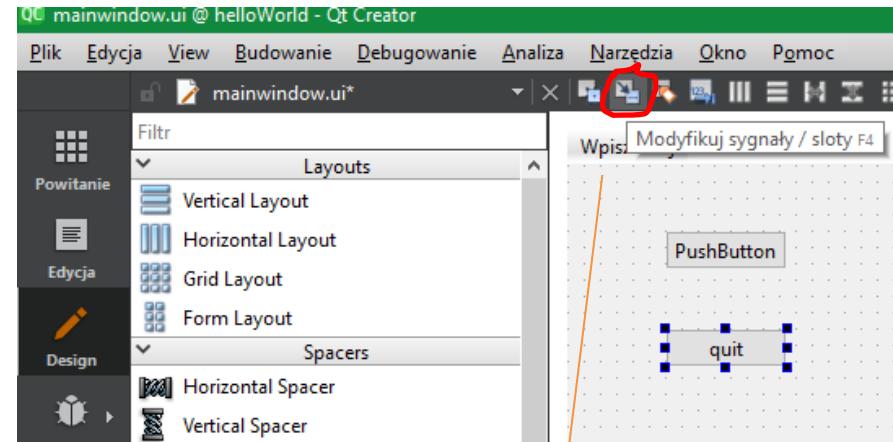
Wpisz aby znaleźć (Ctrl+K) 1 Problemy 2 Wyniki wyszukiwań 3 Komunikaty aplikacji 4 Komunikaty kompilatora 5 QML Debugger Console 6 Komunikaty ogólne 8 Wyniki testu

# helloWorld

dodajemy nowy przycisk



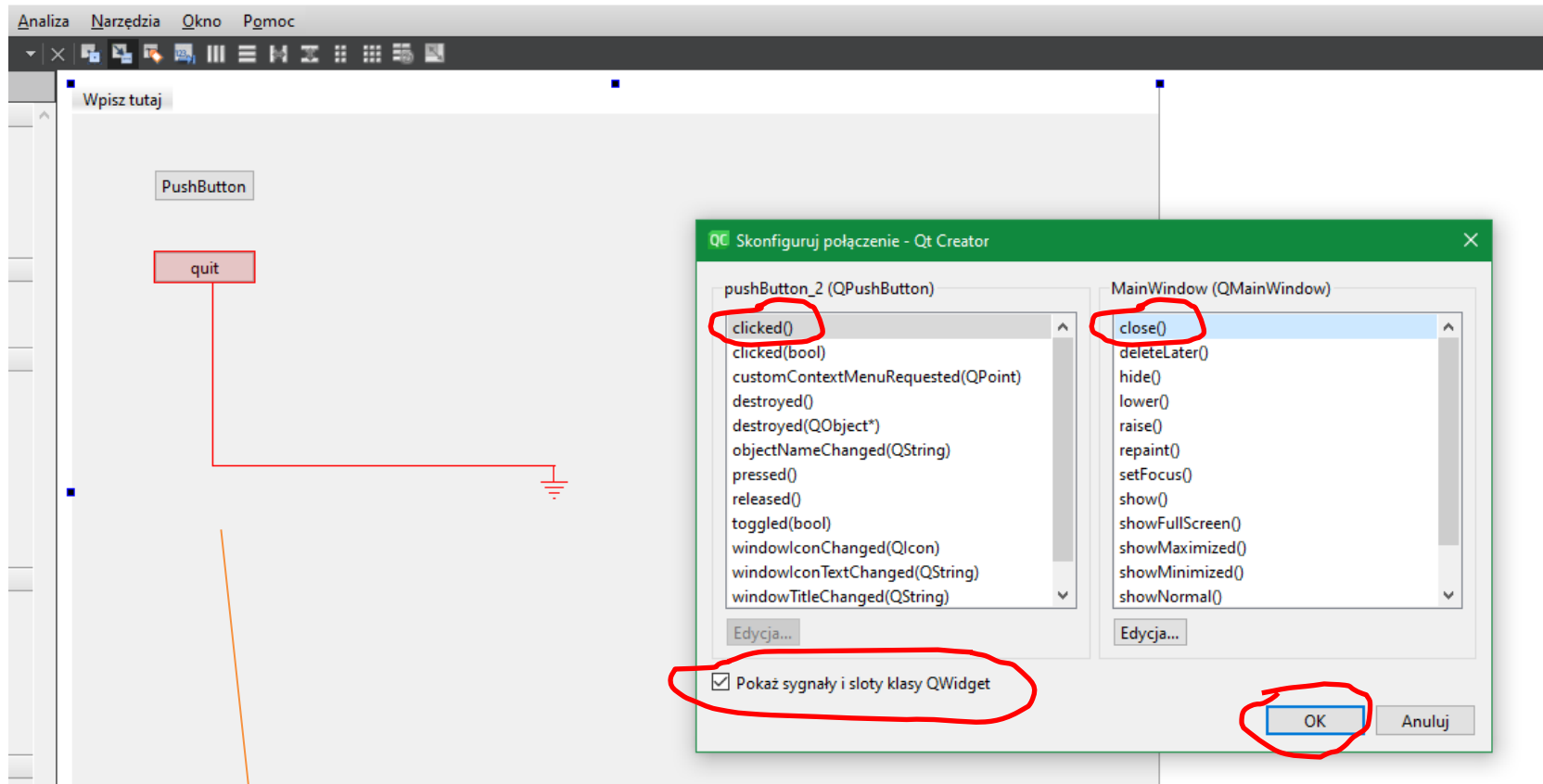
zmiana nazwy przycisku na quit



włączamy przycisk

# helloWorld

przypisanie do sygnału clicked() slotu close()



przeciągamy lewym klawiszem myszy w dół zaczynając od przycisku

# helloWorld

QC mainwindow.ui @ helloWorld - Qt Creator

Plik Edycja View Budowanie Debugowanie Analiza Narzędzia Okno Pomoc

mainwindow.ui\*

Filtr

- Layouts
  - Vertical Layout
  - Horizontal Layout
  - Grid Layout
  - Form Layout
- Spacers
  - Horizontal Spacer
  - Vertical Spacer
- Buttons
  - Push Button
  - Tool Button
  - Radio Button
  - Check Box
  - Command Link Button
  - Dialog Button Box
- Item Views (Model-Based)
  - List View
  - Tree View
  - Table View
  - Column View
  - Undo View
- Item Widgets (Item-Based)
  - List Widget
  - Tree Widget
  - Table Widget
- Containers

Wpisz tutaj

PushButton

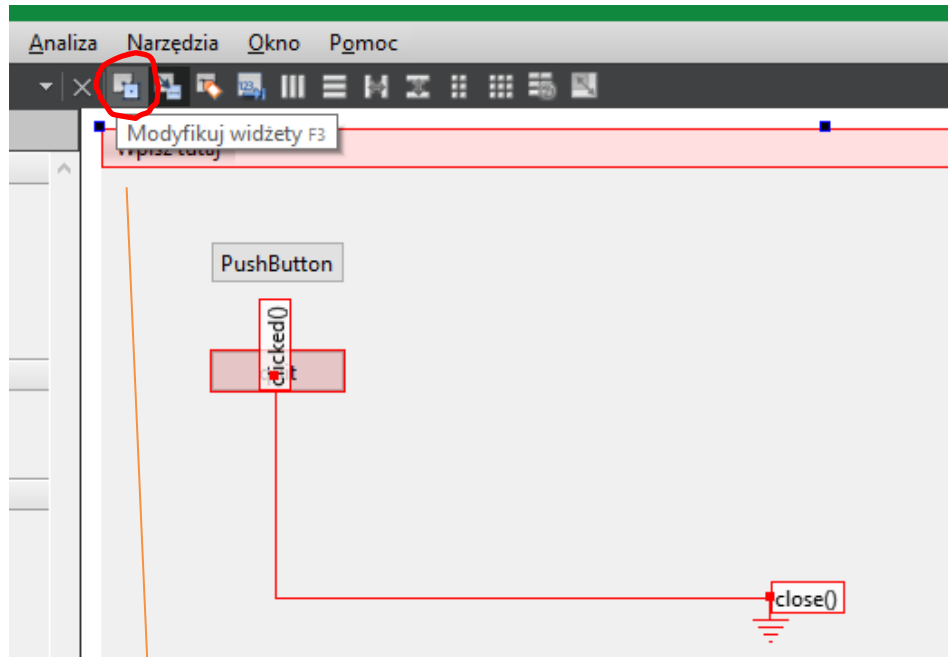
clicked()

close()

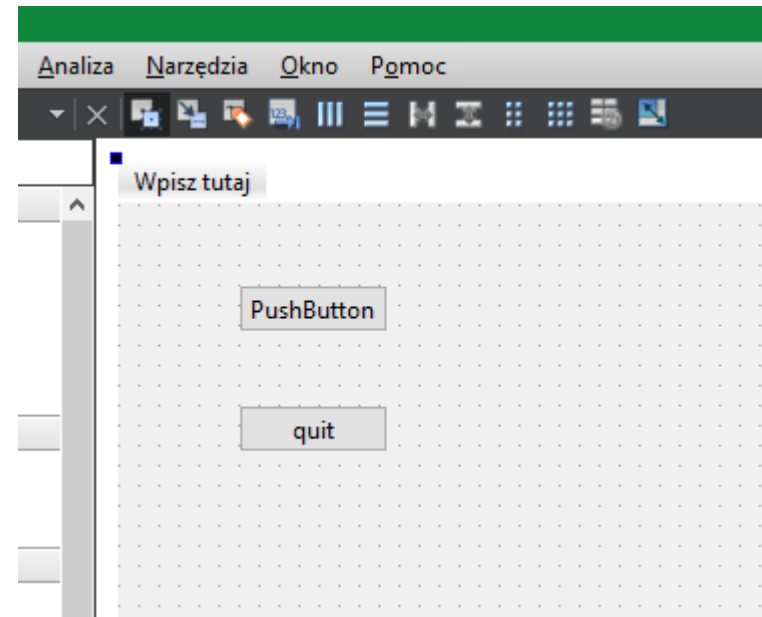
graficzne przedstawienie połączenia sygnału clicked() i slotu close()

The image shows the Qt Creator IDE interface. On the left is the 'Toolbox' with various widget categories. The main area displays a window titled 'mainwindow.ui' with a text input field 'Wpisz tutaj' and a 'PushButton' widget. A red line connects the 'clicked()' signal from the button to the 'close()' slot. An orange arrow points to this connection with the text 'graficzne przedstawienie połączenia sygnału clicked() i slotu close()'. The bottom of the window has a red bar.

# helloWorld

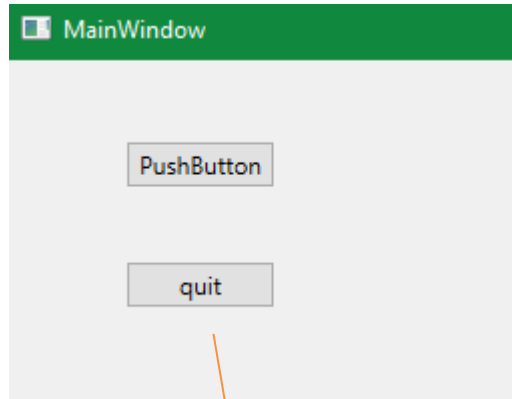


powrót do trybu modyfikowania widżeta



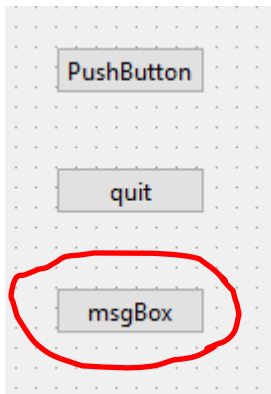


# helloWorld



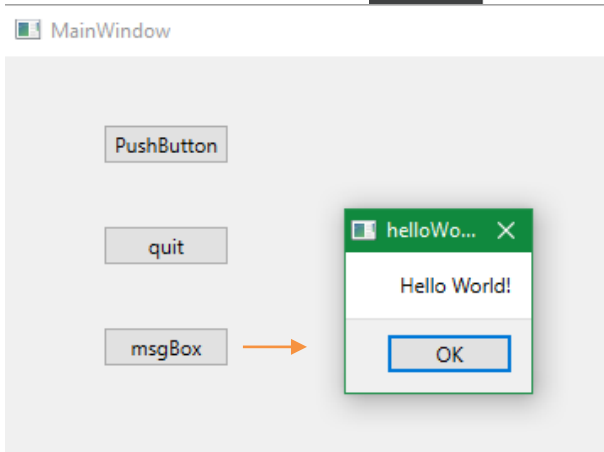
po naciśnięciu na przycisk *quit* następuje zamknięcie aplikacji

# helloWorld



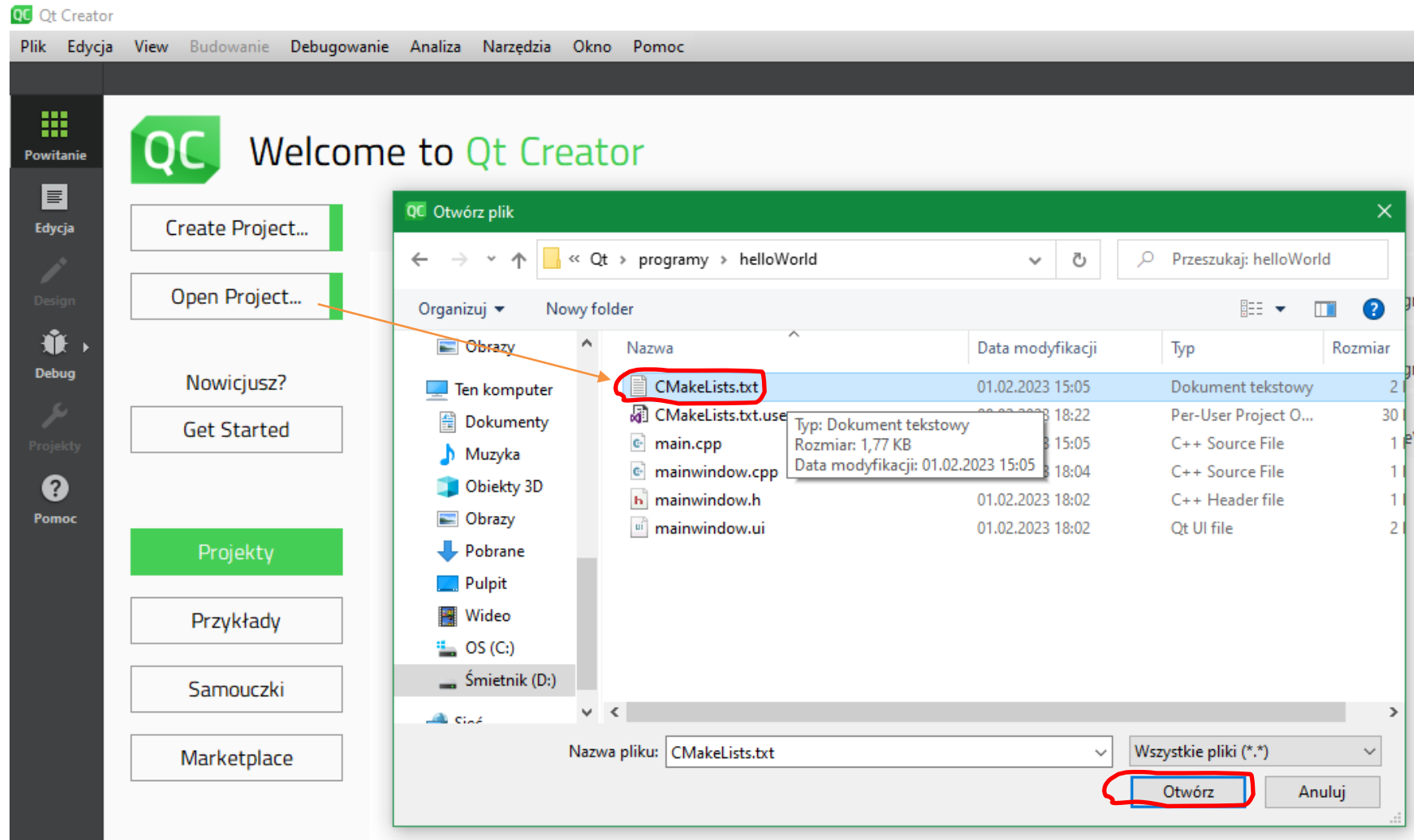
A screenshot of the Qt Creator IDE. The top bar shows the menu: Plik, Edycja, View, Budowanie, Debugowanie, Analiza, Narzędzia, Okno, Pomoc. The left sidebar shows the project structure for 'helloWorld' with files: CMakeLists.txt, mainwindow.h, main.cpp, mainwindow.ui, and CMake Modules. The main editor shows the source code for mainwindow.cpp. The code includes the following lines:

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QDebug>
4 #include <QMessageBox>
5
6 MainWindow::MainWindow(QWidget *parent)
7     : QMainWindow(parent)
8     , ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11 }
12
13 MainWindow::~MainWindow()
14 {
15     delete ui;
16 }
17
18
19 void MainWindow::on_pushButton_clicked()
20 {
21     qDebug() << "Wcisnąłeś przycisk!";
22 }
23
24
25
26 void MainWindow::on_pushButton_3_clicked()
27 {
28     QMessageBox msgBox;
29     msgBox.setText("Hello World!");
30     msgBox.exec();
31 }
```



# helloWorld

## otwarcie zapisanego projektu



aby otworzyć zapisany projekt wybieramy plik CMakeLists.txt

# deployment

1

2

3

4

po uruchomieniu aplikacji w folderze *Build directory* wygenerowany zostanie plik exe

Additional CMake options:

Key	Wartość
CMAKE_BUILD_TYPE	Release
CMAKE_CXX_FLAGS_INIT	
CMAKE_C_COMPILER	D:/Qt/Tools/mingw1120_64/bin/gcc.exe
CMAKE_GENERATOR	Ninja
CMAKE_GNUtoMS	<input type="checkbox"/> OFF
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/helloWorld
CMAKE_PREFIX_PATH	D:/Qt/6.4.2/mingw_64
QT_ADDITIONAL_HOST_PACKAGES_PREFIX_PATH	
QT_ADDITIONAL_PACKAGES_PREFIX_PATH	
QT_DIR	D:/Qt/6.4.2/mingw_64/lib/cmake/Qt6
QT_QMAKE_EXECUTABLE	D:/Qt/6.4.2/mingw_64/bin/qmake.exe
Qt6CoreTools_DIR	D:/Qt/6.4.2/mingw_64/lib/cmake/Qt6CoreTools
Qt6CoreTools_EXECUTABLE	D:/Qt/6.4.2/mingw_64/bin/Qt6CoreTools.exe

# deployment

## Initial deployment (Quick and dirty)

1. Close Qt Creator.
2. Copy the following into C:\Deployment\
  1. The *release version* of **MyApp.exe**
  2. All the **.dll files** from C:\Qt\5.2.1\mingw48\_32\bin\
    3. All the **folders** from C:\Qt\5.2.1\mingw48\_32\plugins\
      4. (If you used QML) All the **folders** from C:\Qt\5.2.1\mingw48\_32\qml\
  3. Rename C:\Qt\ to C:\QtHidden\ (This turns your PC into a clean environment, just like one that doesn't have Qt installed).
  4. Launch C:\Deployment\MyApp.exe.

najpierw przenosimy plik helloWorld.exe, pliki dll oraz pluginy do jednego folderu *Deployment*

If your app worked correctly, congratulations! You are almost ready for deployment. You don't want to ship a 1.5GB package though, so it's time to clean up unused files.

If it didn't work correctly, ask for help (see the Appendix)

## Final deployment (Cleaned up)

Do the deletion steps below in C:\Deployment\ and all of its subdirectories. After each deletion, launch C:\Deployment\MyApp.exe and test it. If it stops working, restore the files you just deleted.

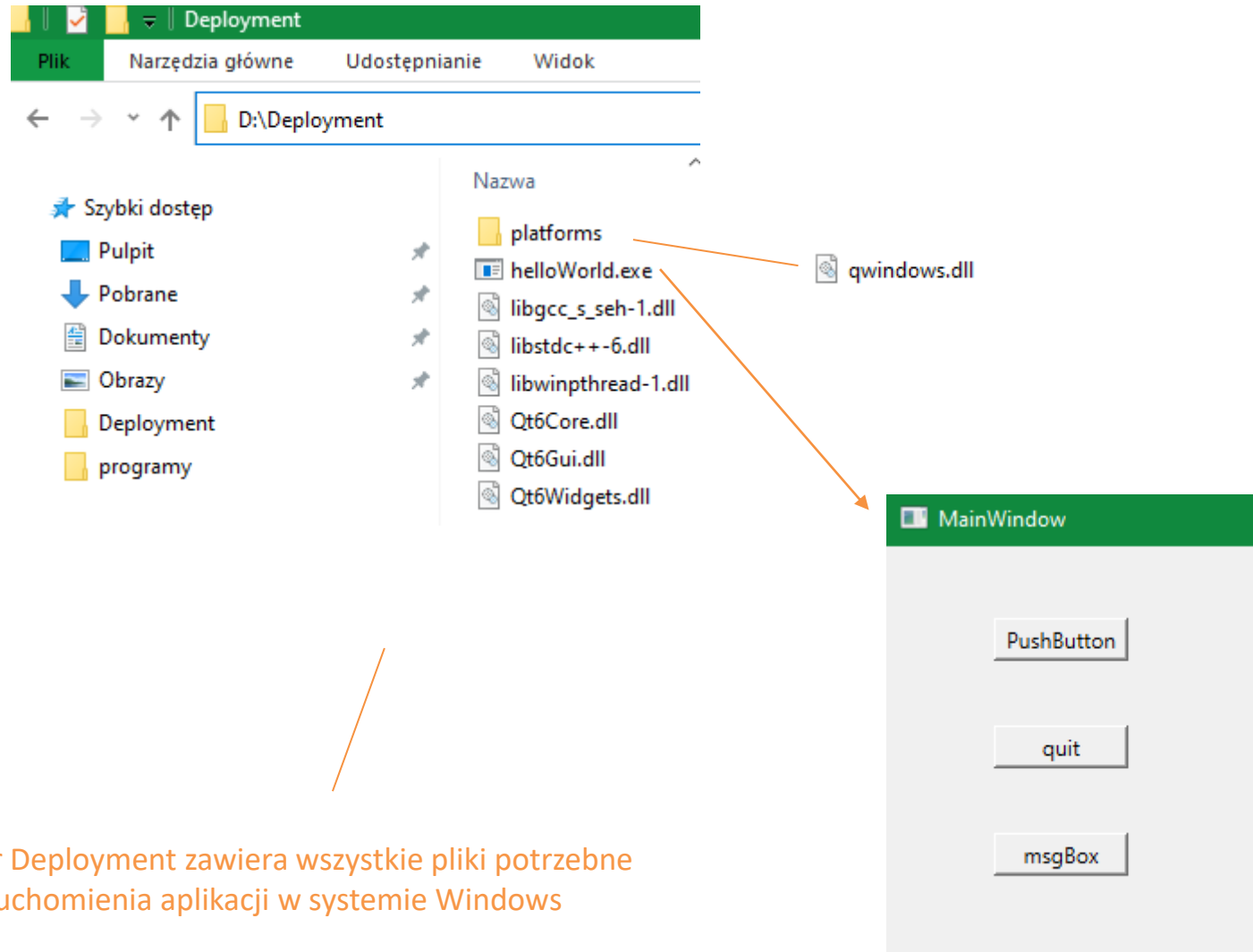
1. Launch MyApp.exe. While it is running, try to delete all DLLs. The DLLs that aren't used will go to the recycle bin, leaving behind only the DLLs that you need. (This trick doesn't work for .qml and qmldir files, however).
2. (If you used QML) Delete a few .qml files and try relaunching MyApp.exe. Repeat until you try all .qml files.
3. (If you used QML) Delete *qmldir* files from the folders that have no more DLLs or .qml files

When you have removed all the files that you don't need,

1. Rename C:\QtHidden\ back to C:\Qt\ to restore your installation.
2. Distribute your app.

następnie uruchamiamy plik exe i usuwamy wszystkie pliki, które są niepotrzebne (system nie pozwoli usunąć plików, które są potrzebne)

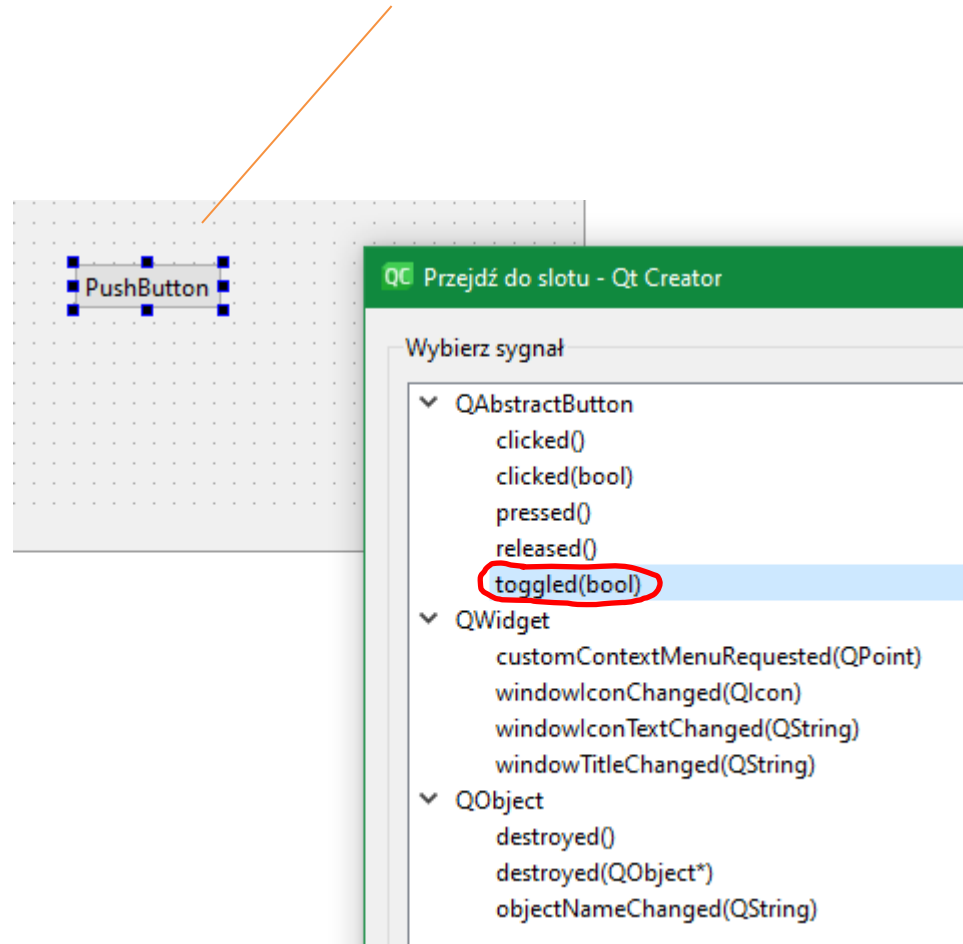
# deployment



folder Deployment zawiera wszystkie pliki potrzebne do uruchomienia aplikacji w systemie Windows

# toggle button

Push Button



# toggle button

The image shows a Qt Creator IDE window with a project named 'tmp'. The file explorer on the left shows the project structure: CMakeLists.txt, tmp, Header Files (mainwindow.h), Source Files (main.cpp, mainwindow.cpp), mainwindow.ui, and CMake Modules.

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     // https://doc.qt.io/qt-6/qpushbutton.html
10    // ustawienie toggle behavior przycisku (2 stany: przycisk włączony, wyłączony)
11    ui -> pushButton -> setCheckable(true);
12    // sprawdzenie toggle behavior
13    qDebug() << ui -> pushButton -> isCheckable();
14 }
15
16 MainWindow::~MainWindow()
17 {
18     delete ui;
19 }
20
21 // przełączanie przycisku
22 void MainWindow::on_pushButton_toggled(bool checked)
23 {
24     if(checked){
25         qDebug() << "przycisk włączony";
26     }else{
27         qDebug() << "przycisk wyłączony";
28     }
29 }
30
```

Below the code editor, a small window titled 'tmp' displays a button labeled 'PushButton'. A red arrow points from the button in the UI to the corresponding code in the editor.

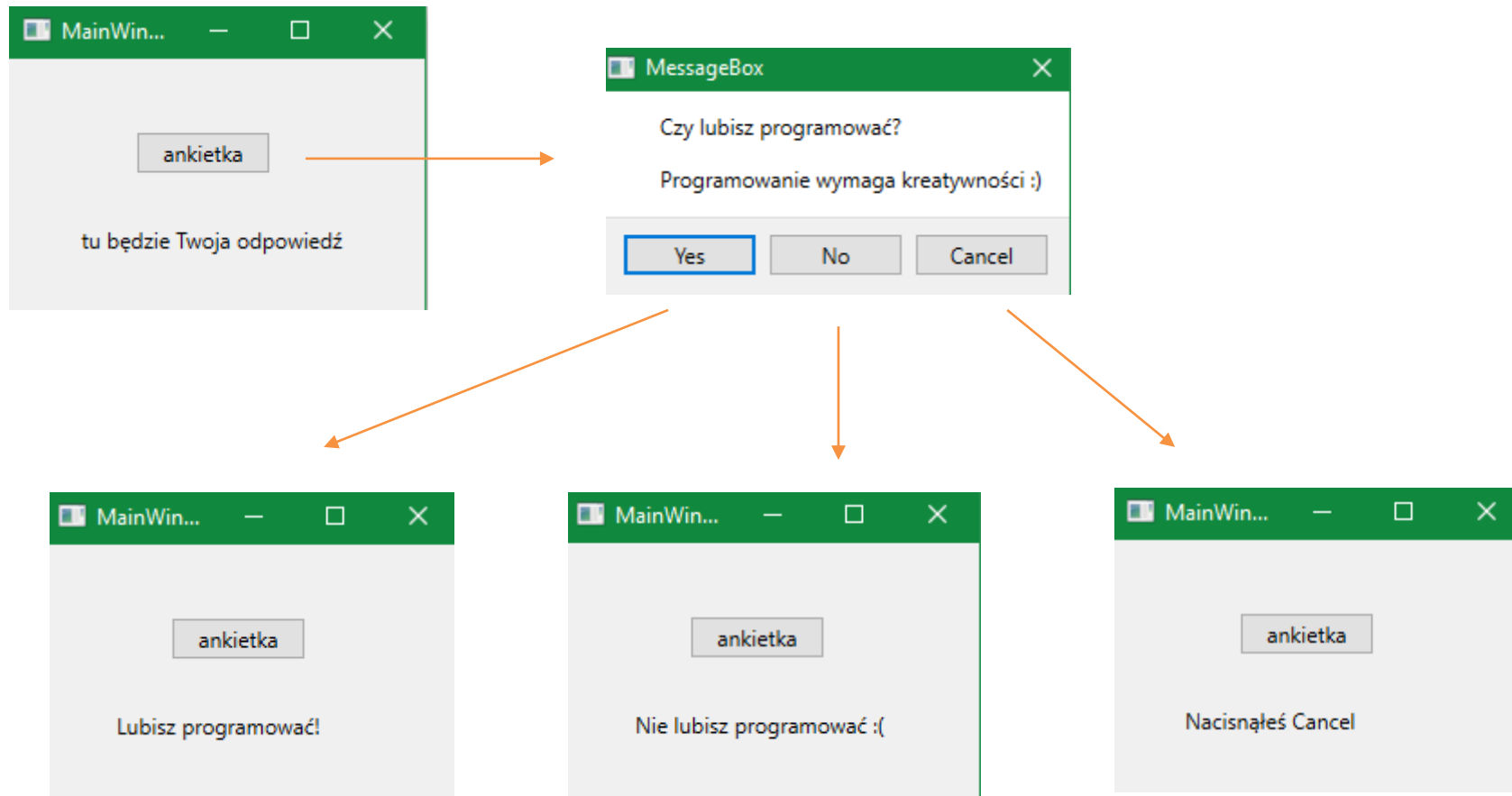
At the bottom, the 'Komunikaty aplikacji' (Application Messages) pane shows the following output:

```
17:39:20: D:\arch\TechnikProgramista\_programowanieAplikacjiDesktopowych\Qt\programy\build-tmp-Desktop_Qt_6_4_2_Mi
18:01:26: Uruchamianie D:\arch\TechnikProgramista\_programowanieAplikacjiDesktopowych\Qt\programy\build-tmp-Deskte
true
przycisk włączony
przycisk wyłączony
przycisk włączony
przycisk wyłączony
przycisk włączony
```

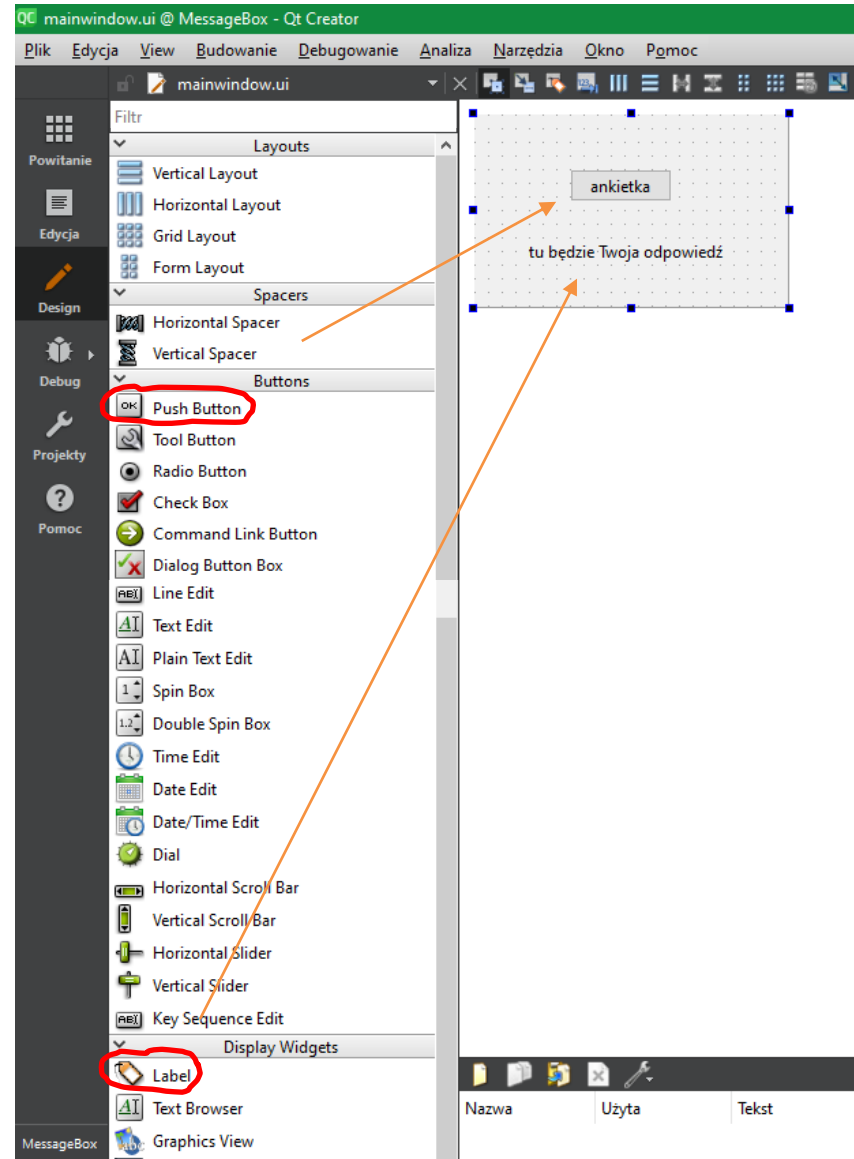
Red arrows indicate the flow of information: from the code editor to the UI window, and from the UI window to the application messages pane.



# MessageBox



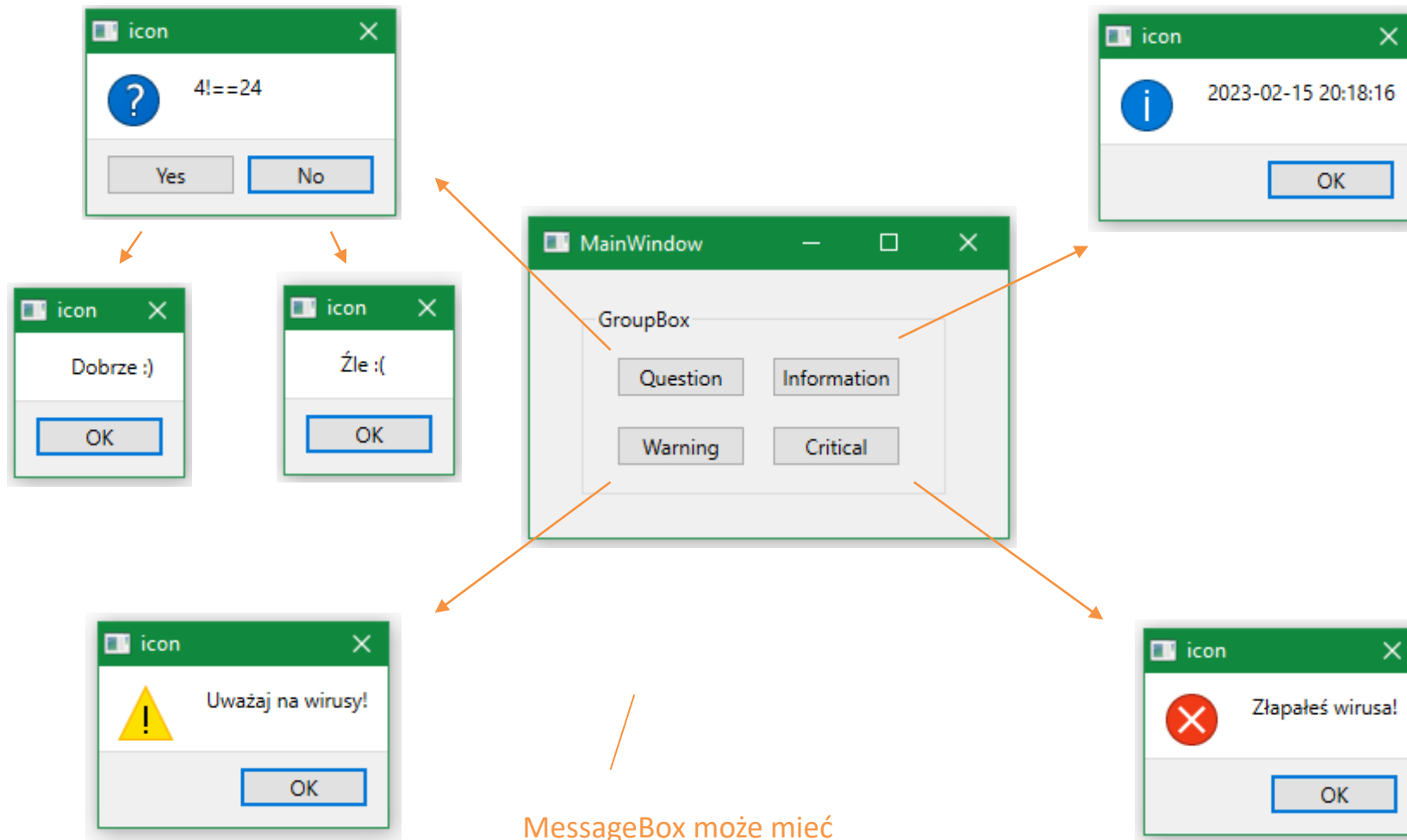
# MessageBox



# MessageBox

```
void MainWindow::on_pushButton_clicked()
{
    QMessageBox msgBox;
    msgBox.setText("Czy lubisz programować?");
    msgBox.setInformativeText("Programowanie wymaga kreatywności :)");
    msgBox.setStandardButtons(QMessageBox::Yes | QMessageBox::No | QMessageBox::Cancel);
    msgBox.setDefaultButton(QMessageBox::Yes);
    int ret = msgBox.exec();
    switch (ret) {
        case QMessageBox::Yes:
            ui->label->setText("Lubisz programować!");
            break;
        case QMessageBox::No:
            ui->label->setText("Nie lubisz programować :(");
            break;
        case QMessageBox::Cancel:
            ui->label->setText("Nacisnąłeś Cancel");
            break;
        default:
            // should never be reached
            break;
    }
}
```

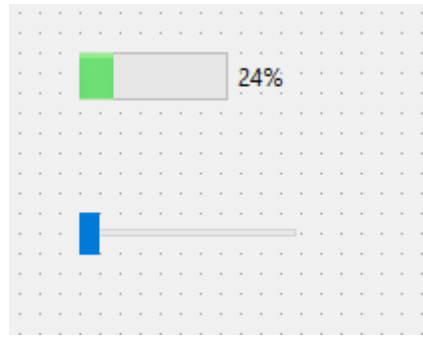
# MessageBox icon



MessageBox może mieć  
jedną z 4 ikonek:  
*Question, Information,  
Warning, Critical*

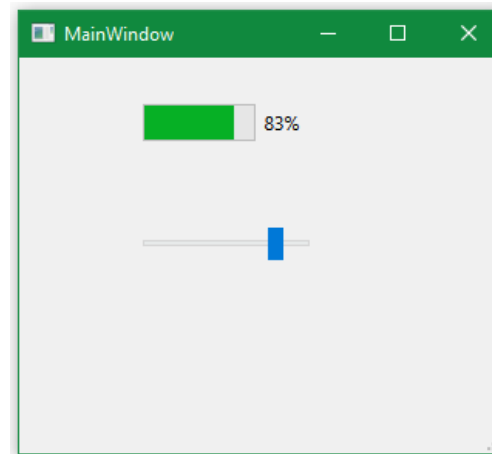
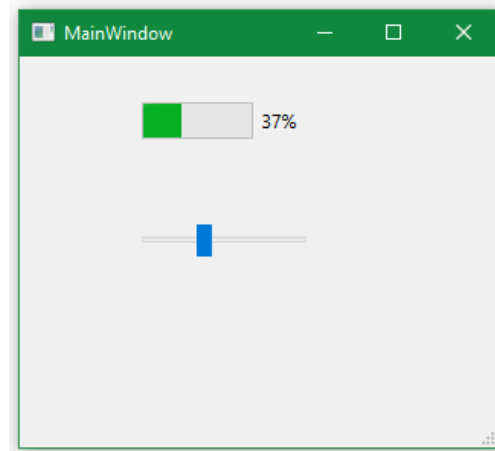
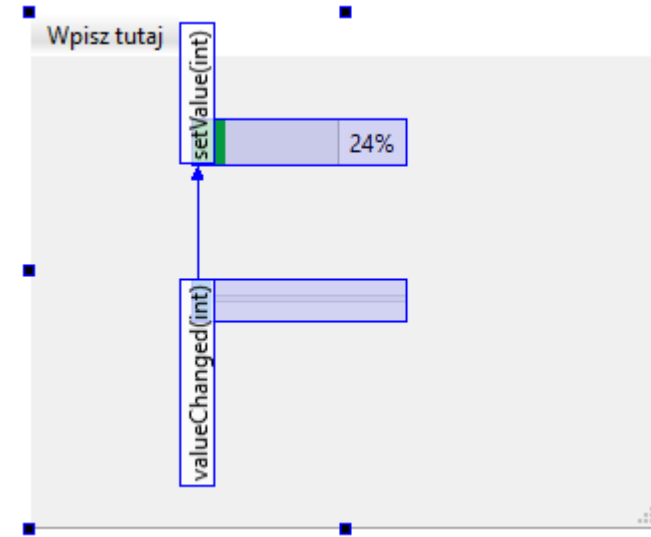
# Progress Bar

Progress bar



Horizontal Slider

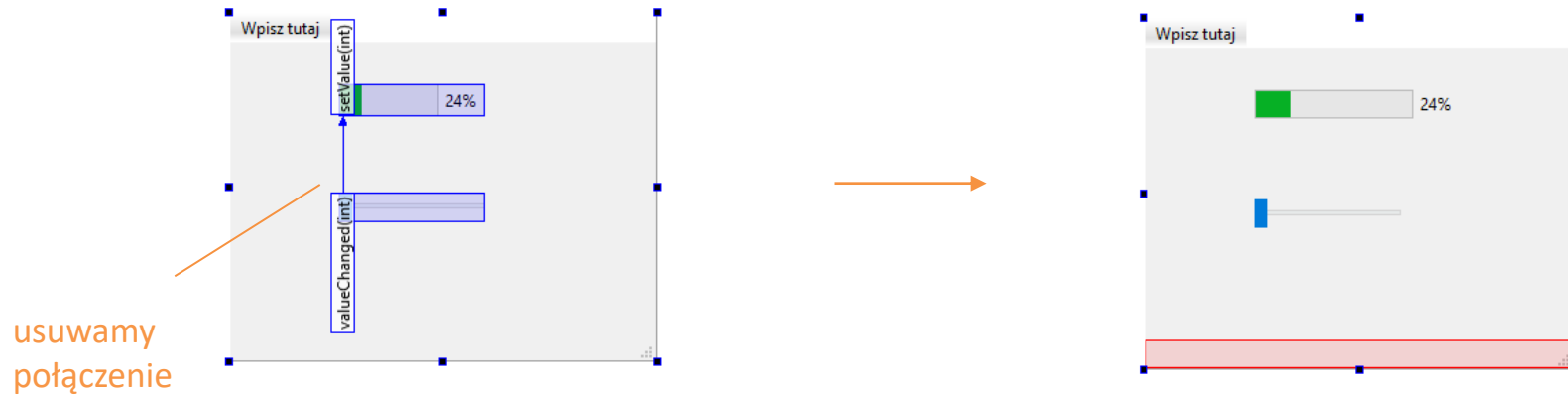
łączymy sygnał ze Slidera ze słotem w Progress Bar



[youtube](#)

wielkość przesunięcia suwaka jest odwzorowana w pasku postępu

# Progress Bar



```
Projekty | mainWindow.cpp* | MainWindow::~MainWindow()
> icon
  > progressBar
    > CMakeLists.txt
    > progressBar
      > Header Files
      > Source Files
        > main.cpp
        > mainWindow.cpp
        > mainWindow.ui
    > CMake Modules
```

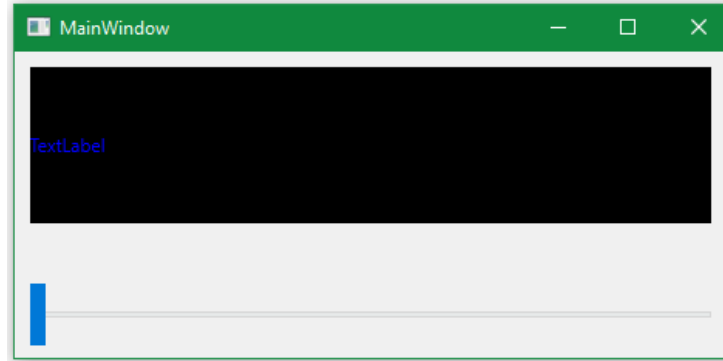
```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     connect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
10           ui->progressBar, SLOT(setValue(int)));
11     //disconnect(ui->horizontalSlider, SIGNAL(valueChanged(int)),
12               // ui->progressBar, SLOT(setValue(int)));
13 }
14
15 MainWindow::~MainWindow()
16 {
17     delete ui;
18 }
```

wpisujemy połączenie za pomocą kodu (działa tak samo)

rozłączenie sygnału ze slotem za pomocą kodu

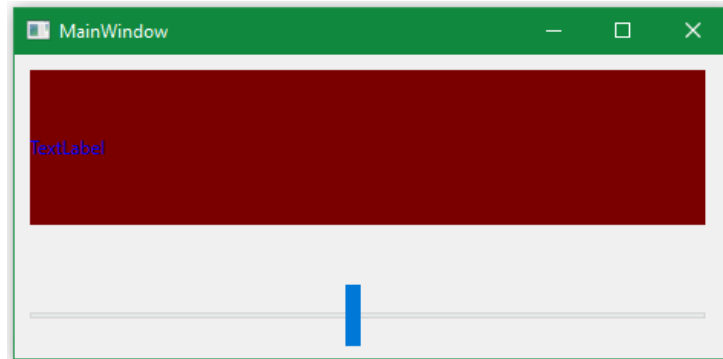
[youtube](https://www.youtube.com)

# Slider



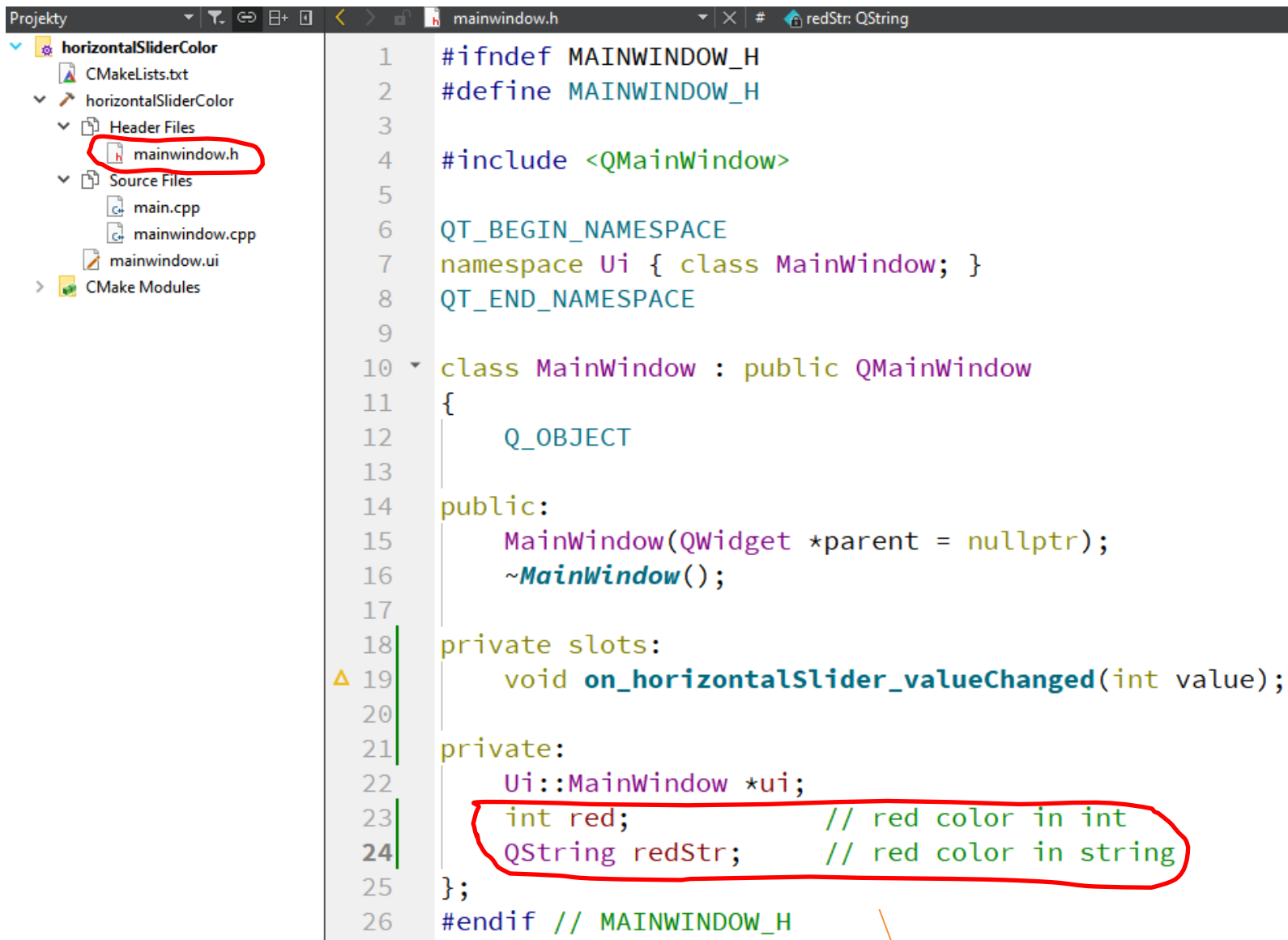
Label

Horizontal Slider (red)



kolor labela jest uzależniony od położenia suwaka (tylko składnik czerwony koloru ulega zmianie)

# Slider



```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class MainWindow; }
8  QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     MainWindow(QWidget *parent = nullptr);
16     ~MainWindow();
17
18 private slots:
19     void on_horizontalSlider_valueChanged(int value);
20
21 private:
22     Ui::MainWindow *ui;
23     int red;           // red color in int
24     QString redStr;   // red color in string
25 };
26 #endif // MAINWINDOW_H
```

polo klasy MainWindow przeznaczone na wartości koloru czerwonego



# Slider

ustawienie zakresu czerwonego slidera  
oraz wartości początkowej koloru czerwonego  
w konstruktorze okna

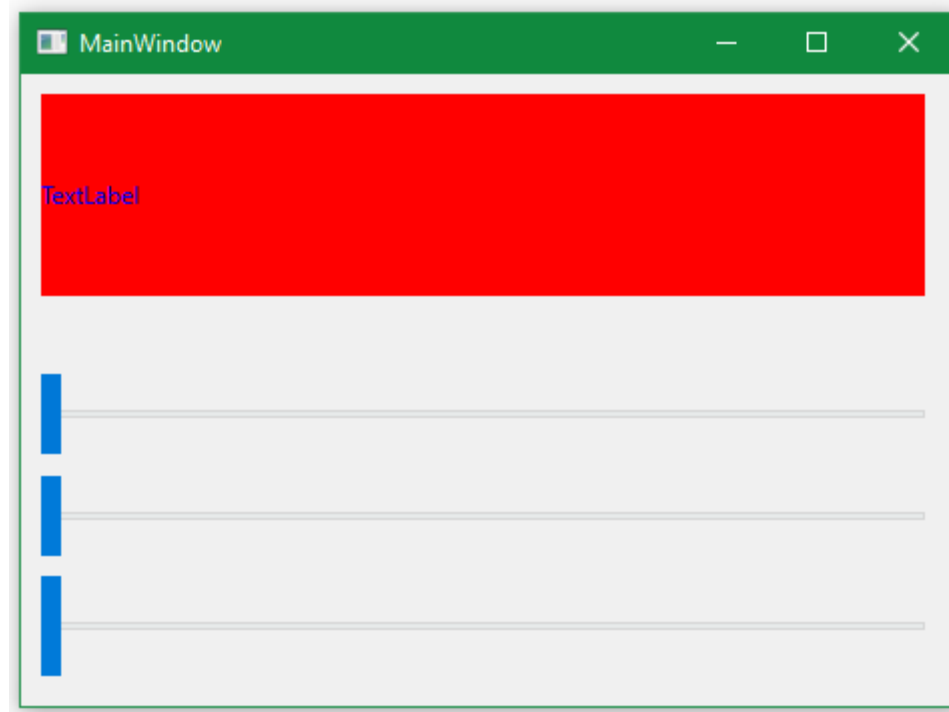


```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) // constructor
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     ui->horizontalSlider->setMinimum(0); // sets minimal value of slider
10    ui->horizontalSlider->setMaximum(255); // sets maximal value of slider
11    red = 0; // sets initial color of label (when program starts)
12    redStr = QString::number(red); // change red to string
13    // sets initial red color of label
14    ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + ",0,0); color : blue; }");
15 }
16
17 MainWindow::~MainWindow() // destructor
18 {
19     delete ui;
20 }
21
22 // slider red
23 void MainWindow::on_horizontalSlider_valueChanged(int value)
24 {
25     red=value; // assign value of slider to red
26     redStr = QString::number(red); // change red to string
27     // set red color of label
28     ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + ",0,0); color : blue; }");
29 }
```

zmiana wartości koloru czerwonego za pomocą slidera  
(w słocie on\_horizontalSlider\_valueChanged()  
połączonym z sygnałem valueChanged()  
generowanym przez slider)

<https://stackoverflow.com/questions/2749798/qlabel-set-color-of-text-and-background>

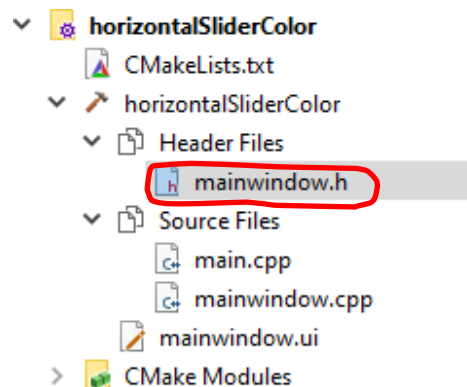
# Slidery RGB



red  
green  
blue

Dołącz slidery zmieniające kolory green oraz blue

# Slidery RGB

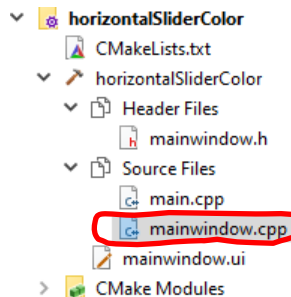


poła klasy MainWindow przeznaczone na wartości kolorów RGB

```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  QT_BEGIN_NAMESPACE
7  namespace Ui { class MainWindow; }
8  QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     MainWindow(QWidget *parent = nullptr);
16     ~MainWindow();
17
18 private slots:
19     void on_horizontalSlider_valueChanged(int value);
20
21     void on_horizontalSlider_2_valueChanged(int value);
22
23     void on_horizontalSlider_3_valueChanged(int value);
24
25 private:
26     Ui::MainWindow *ui;
27     int red;           // red color in int
28     QString redStr;   // red color in string
29     int green;        // green color in int
30     QString greenStr; // green color in string
31     int blue;         // blue color in int
32     QString blueStr;  // blue color in string
33 };
34 #endif // MAINWINDOW_H
```

# Slidery RGB

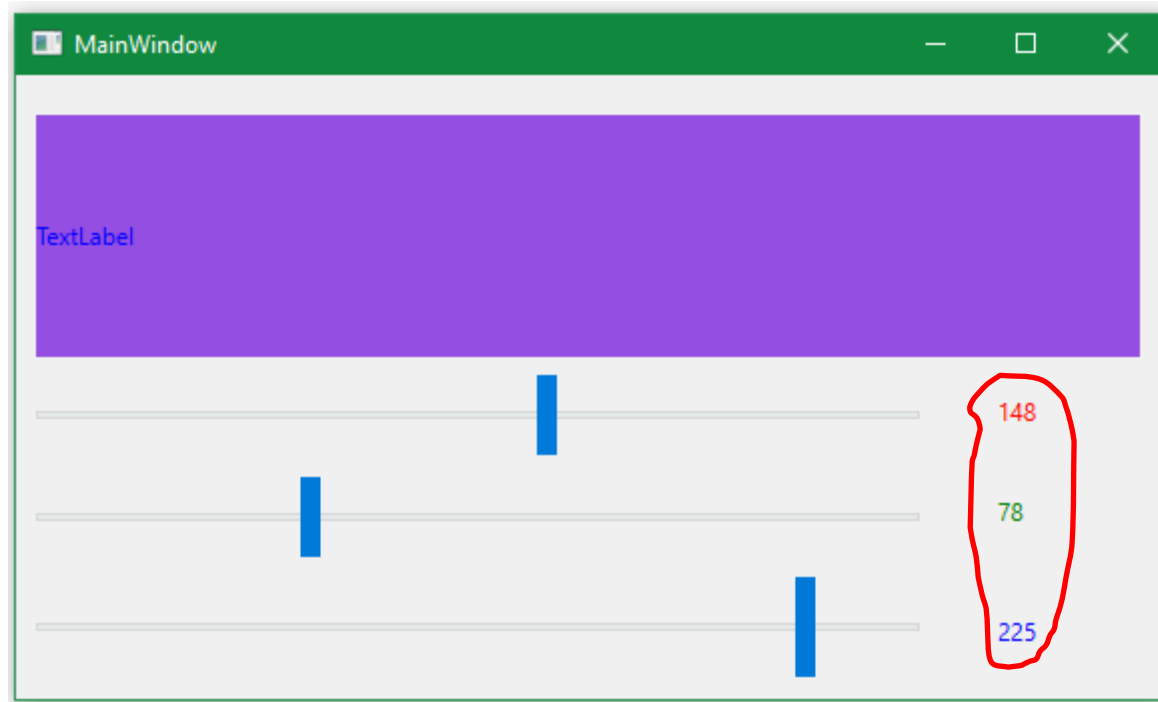
ustawienie zakresów sliderów  
oraz wartości początkowej kolorów  
w konstruktorze okna



zmiana wartości  
kolorów RGB  
za pomocą sliderów

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) // constructor
5 : QMainWindow(parent)
6 , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     ui->horizontalSlider->setMinimum(0); // sets minimal value of slider
10    ui->horizontalSlider->setMaximum(255); // sets maximal value of slider
11    ui->horizontalSlider_2->setMinimum(0);
12    ui->horizontalSlider_2->setMaximum(255);
13    ui->horizontalSlider_3->setMinimum(0);
14    ui->horizontalSlider_3->setMaximum(255);
15    red = 0; // sets initial color of label (when program starts)
16    redStr = QString::number(red); // change red to string
17    green = 0;
18    greenStr = QString::number(green);
19    blue = 0;
20    blueStr = QString::number(blue);
21    // sets initial red color of label
22    ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + "," + greenStr + "," + blueStr + "); color : blue; }");
23 }
24
25 MainWindow::~MainWindow() // destructor
26 {
27     delete ui;
28 }
29
30 // slider red
31 void MainWindow::on_horizontalSlider_valueChanged(int value)
32 {
33     red=value; // assign value of slider to red
34     redStr = QString::number(red); // change red to string
35     // set red color of label
36     ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + "," + greenStr + "," + blueStr + "); color : blue; }");
37 }
38
39
40 // slider green
41 void MainWindow::on_horizontalSlider_2_valueChanged(int value)
42 {
43     green=value; // assign value of slider to green
44     greenStr = QString::number(green); // change green to string
45     // set green color of label
46     ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + "," + greenStr + "," + blueStr + "); color : blue; }");
47 }
48
49
50 // slider blue
51 void MainWindow::on_horizontalSlider_3_valueChanged(int value)
52 {
53     blue=value; // assign value of slider to blue
54     blueStr = QString::number(blue); // change blue to string
55     // set blue color of label
56     ui->label->setStyleSheet("QLabel { background-color : rgb(" + redStr + "," + greenStr + "," + blueStr + "); color : blue; }");
57 }
58
59 }
```

# Slidery RGB

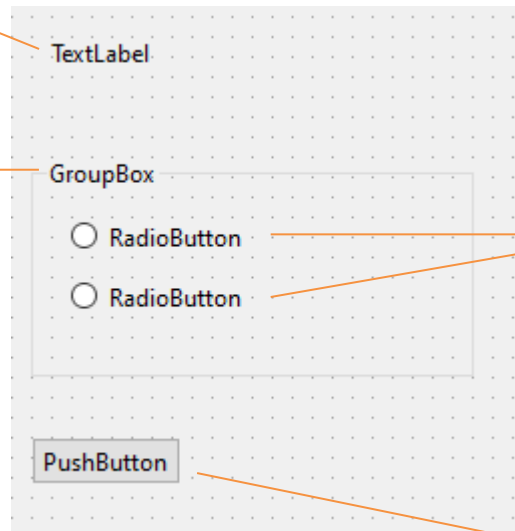


wstaw labelę z wartościami kolorów RGB

# QRadioButton

wstaw zagadkę,  
zwiększ rozmiar czcionki

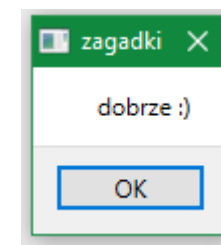
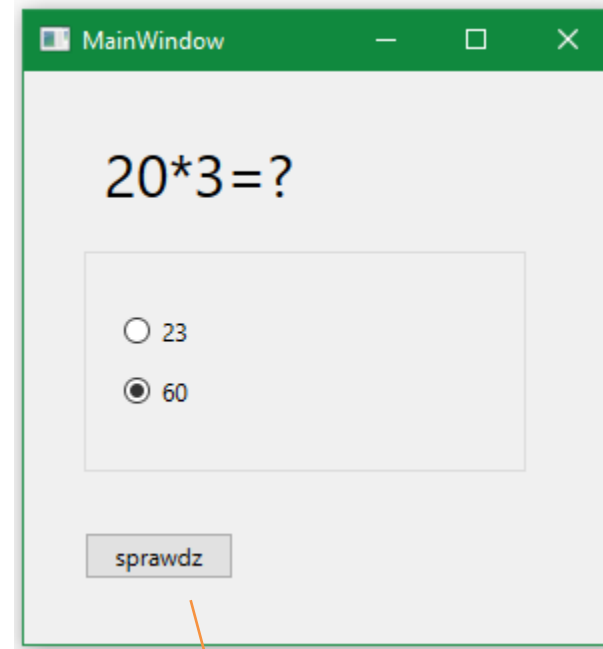
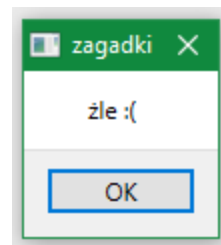
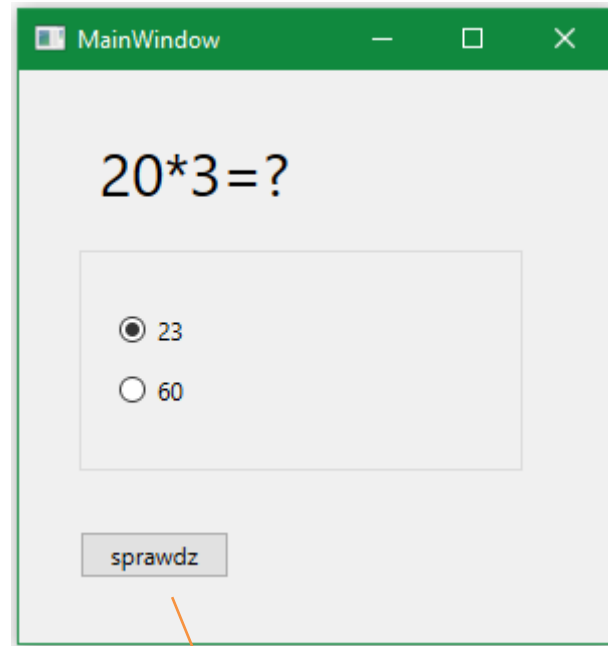
usuń napis



wstaw odpowiedzi

zmień napis, sprawdź odpowiedź,  
wyświetl QMessageBox z wynikiem

# QRadioButton



# QRadioButton

mainwindow.cpp

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"

#include <QMessageBox>

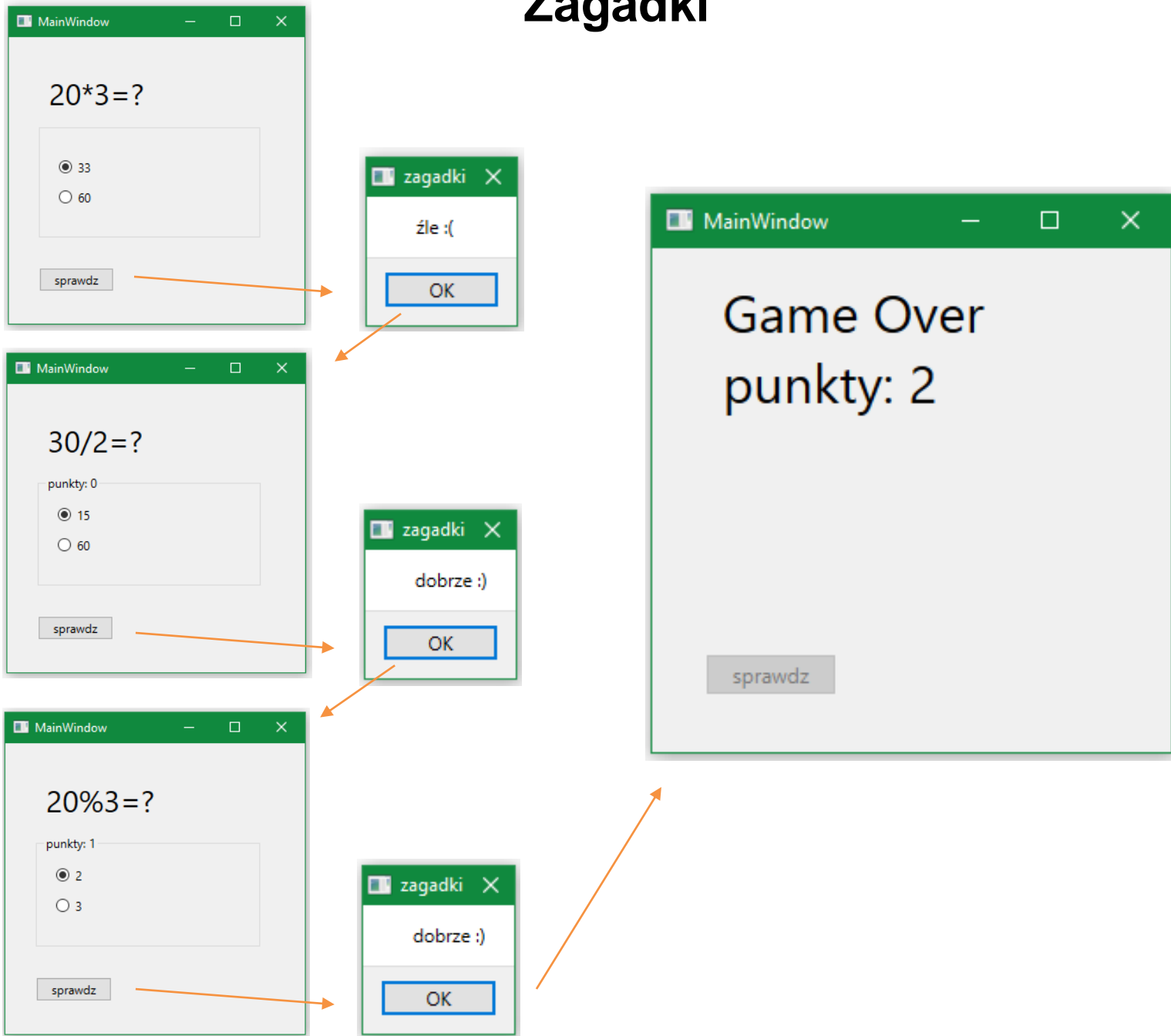
MainWindow::MainWindow(QWidget *parent) // konstruktor
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->radioButton->setChecked(false); // zaznacza pierwszy radiobutton
    ui->pushButton->setText("sprawdz"); // ustawia tekst na przycisku
    ui->radioButton->setText("23"); // odpowiedź 1
    ui->radioButton_2->setText("60"); // odpowiedź 2
    ui->groupBox->setTitle(""); // usuwa tytuł z groupBox
    ui->label->setText("20*3=?"); // ustawia text zagadki
    ui->label->setStyleSheet("QLabel {font-size: 30px}"); // ustawia styl zagadki
}

MainWindow::~MainWindow()
{
    delete ui;
}

// sprawdzenie odpowiedzi użytkownika
void MainWindow::on_pushButton_clicked()
{
    QMessageBox msgBox;
    if (ui->radioButton->isChecked()){
        msgBox.setText("źle :(");
        msgBox.exec();
    }else if (ui->radioButton_2->isChecked()){
        msgBox.setText("dobrze :)");
        msgBox.exec();
    }
}
```



# Zagadki



# Zagadki

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

private:
    Ui::MainWindow *ui;
    const static int iloscZagadek = 3;
    QString zagadki[iloscZagadek] = {"20*3=?", "30/2=?", "20%3=?"}; // tablica zagadek
    int i = 0; // index tablic
    int punkty = 0; // punkty
    QString rbText[iloscZagadek] = {"33", "15", "2"}; // radioButton text
    QString rb2Text[iloscZagadek] = {"60", "60", "3"}; // radioButton_2 text
    QString msgBoxRbText[iloscZagadek] = {"źle :(", "dobrze :)", "dobrze :)}"; // tekst wyświetlany w msgBox przy radioButton
    QString msgBoxRb2Text[iloscZagadek] = {"dobrze :)", "źle :(", "źle :("}; // tekst wyświetlany w msgBox przy radioButton_2
};
#endif // MAINWINDOW_H
```

tablice z informacjami tekstowymi  
jako pola prywatne klasy MainWindow

# Zagadki

mainwindow.cpp 1/2

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent) // konstruktor (uruchamia się na początku programu)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->radioButton->setChecked(true); // zaznacza pierwszy radiobutton
    ui->pushButton->setText("sprawdz"); // ustawia tekst na przycisku
    ui->radioButton->setText(rbText[i]); // ustawia początkową odpowiedź 1 (pierwszej zagadki)
    ui->radioButton_2->setText(rb2Text[i]); // ustawia początkową odpowiedź 2 (pierwszej zagadki)
    ui->groupBox->setTitle(""); // usuwa tytuł z groupBox
    ui->label->setText(zagadki[i]); // ustawia text pierwszej zagadki
    ui->label->setStyleSheet("QLabel {font-size: 30px}"); // ustawia styl zagadki - wielkość fontów
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

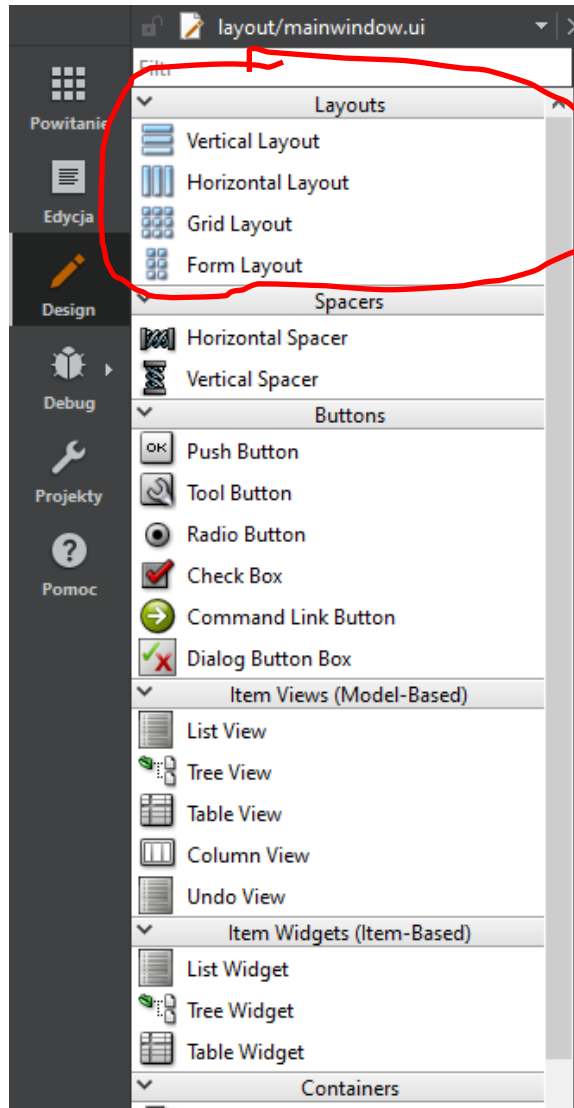
ustawienia początkowe programu

# Zagadki

mainwindow.cpp 2/2

```
// sprawdza odpowiedzi użytkownika po kliknięciu na przycisk
void MainWindow::on_pushButton_clicked()
{
    QMessageBox msgBox;
    if (ui->radioButton->isChecked()){
        msgBox.setText(msgBoxRbText[i]);          // wyświetla w msgBox napis z tablicy msgBoxRbText[]
        msgBox.exec();
        if (msgBoxRbText[i] == "dobrze :"){
            punkty++;
        }
    }else if (ui->radioButton_2->isChecked()){
        msgBox.setText(msgBoxRb2Text[i]);        // wyświetla w msgBox napis z tablicy msgBoxRb2Text[]
        msgBox.exec();
        if (msgBoxRb2Text[i] == "dobrze :"){
            punkty++;
        }
    }
    i++;                                         // inkrementuje numer zagadki
    if (i >= iloscZagadek){                    // sprawdza czy zagadki się skończyły
        ui->label->setText("Game Over\npunkty: " + QString::number(punkty)); // wyświetla informację o zakończeniu gry
        ui->groupBox->setVisible(false);        // chowa groupBox (radioButtony)
        ui->pushButton->setEnabled(false);     // unieruchamia przycisk
    }else{
        ui->label->setText(zagadki[i]);         // ustawia text następnej zagadki
        ui->radioButton->setText(rbText[i]);   // ustawia odpowiedź 1 następnej zagadki
        ui->radioButton_2->setText(rb2Text[i]); // ustawia odpowiedź 2 następnej zagadki
        ui->groupBox->setTitle("punkty: " + QString::number(punkty)); // wyświetla punktację
    }
}
```

# Layout



layouty do wyboru

czerwony znaczek (kółeczko)  
oznacza brak layoutu

Obiekt	Klasa
MainWindow	QMainWindow
centralwidget	QWidget
groupBox	QGroupBox
radioButton	QRadioButton
radioButton_2	QRadioButton
label	QLabel
pushButton	QPushButton

# Vertical Layout

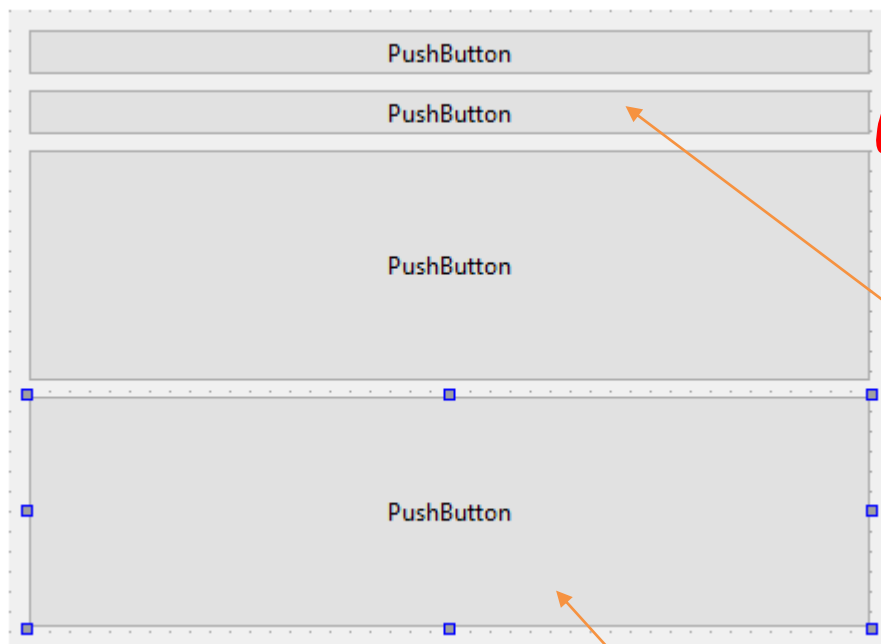
The diagram illustrates the process of converting a vertical stack of four buttons into a vertical layout of four buttons. On the left, four buttons are stacked vertically. A context menu is opened over them, and the 'Rozmieść' (Align) option is selected. A sub-menu is then displayed, showing 'Rozmieść w pionie' (Align vertically) as the chosen option. An arrow points from the sub-menu to the resulting layout on the right, where the buttons are now arranged in a vertical layout.

prawy przycisk myszy

- Utwórz pasek menu
- Dodaj pasek narzędzi
- Add Tool Bar to Other Area
- Utwórz pasek stanu
- Zmień nazwę obiektu...
- Zmień podpowiedź...
- Zmień "co to jest"...
- Zmień arkusz stylu...
- Ograniczenia rozmiaru
- Zastępcze widżety...
- Zmień sygnały/sloty...
- Przejdź do slotu...
- Wytnij Ctrl+X
- Skopiuj Ctrl+C
- Wklej Ctrl+V
- Zaznacz wszystko Ctrl+A
- Usuń
- Rozmieść**
- Dopasuj wielkość Ctrl+J
- Rozmieść w poziomie Ctrl+H
- Rozmieść w pionie Ctrl+L**
- Rozmieść poziomo w splitterze
- Rozmieść pionowo w splitterze
- Rozmieść w siatce Ctrl+G
- Rozmieść w formularzu
- Usuń rozmieszczenie
- Uprość rozmieszczenie w siatce

Nazwa	Użyta	Tekst	Skrót	Przełączalny	Podpowiedź

# Vertical Layout



sizePolicy

pushButton\_3 : QPushButton

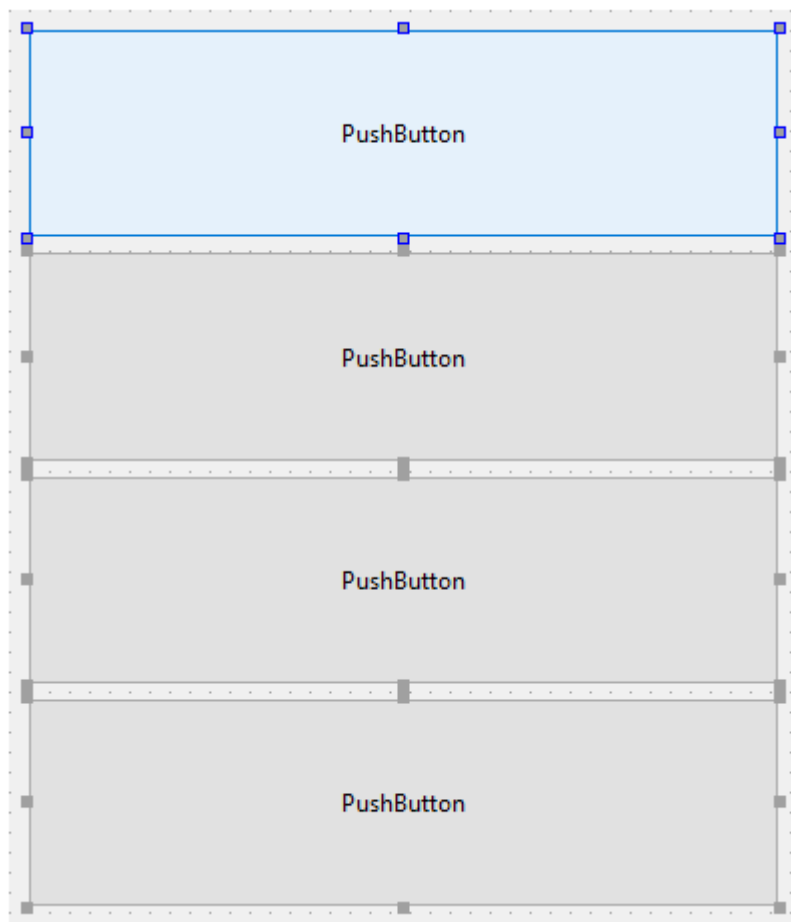
Właściwość	Wartość
QWidget	
sizePolicy	[Minimum, Fixed, 0, 0]
Strategia pozioma	Minimum
Strategia pionowa	Fixed
Rozciąganie w po...	0
Rozciąganie w pi...	0

sizePolicy

pushButton : QPushButton

Właściwość	Wartość
QWidget	
sizePolicy	[Minimum, Expanding, 0, 0]
Strategia pozioma	Minimum
Strategia pionowa	Expanding
Rozciąganie w po...	0
Rozciąganie w pi...	0

# Vertical Layout



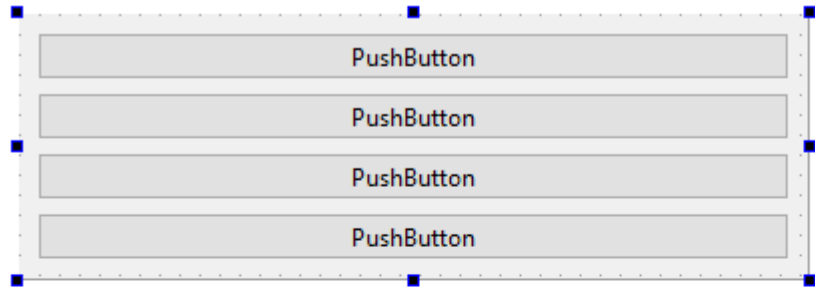
wszystkie przyciski mają Strategię pionową ustawioną na Preferred

A screenshot of the Qt Designer Properties panel for a QPushButton widget. The panel shows various properties and their values. The 'sizePolicy' property is expanded, and the 'Strategia pionowa' (Vertical Strategy) is set to 'Preferred'. This setting is circled in red, and an orange arrow points from the text above to it.

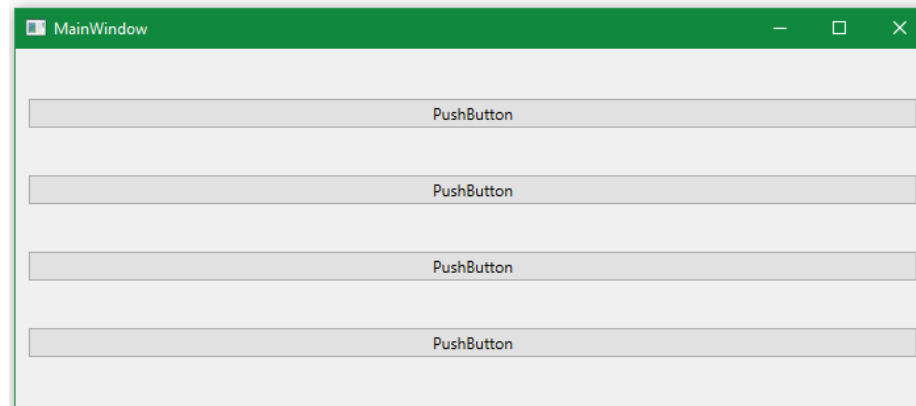
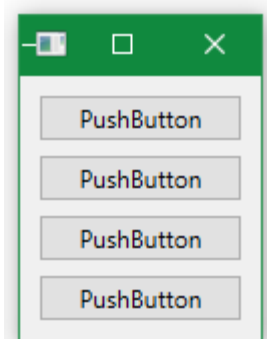
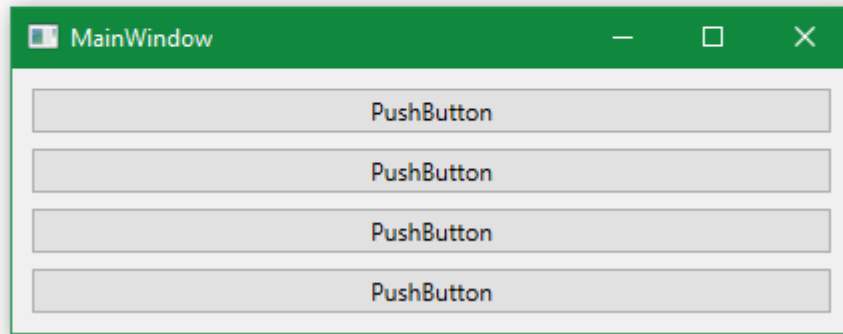
Właściwość	Wartość
▼ QObject	
objectName	pushButton_2
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
> geometry	[(9, 74), 376 x 24]
▼ sizePolicy	
Strategia pozioma	Minimum
Strategia pionowa	Preferred
Rozciąganie w po...	0
Rozciąganie w pi...	0
> minimumSize	0 x 0



# Vertical Layout



dzięki temu, że przyciski umieściliśmy w layoucie, ich wymiary dopasowują się do okna programu



# Form Layout

## Grid Layout

A diagram illustrating a form layout. It features a light gray rectangular container with a dotted grid background. Inside, there are four horizontal elements stacked vertically: a label 'imię', a line edit field, a label 'nazwisko', another line edit field, a label 'klasa', and a third line edit field. At the bottom of the container is a gray button labeled 'submit'. Two orange arrows point from the text 'Label' to the 'imię' and 'nazwisko' labels, and another orange arrow points from 'Line Edit' to the first line edit field.

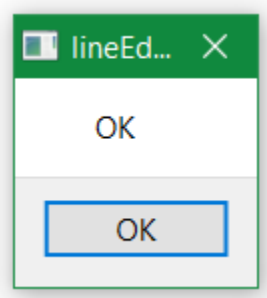
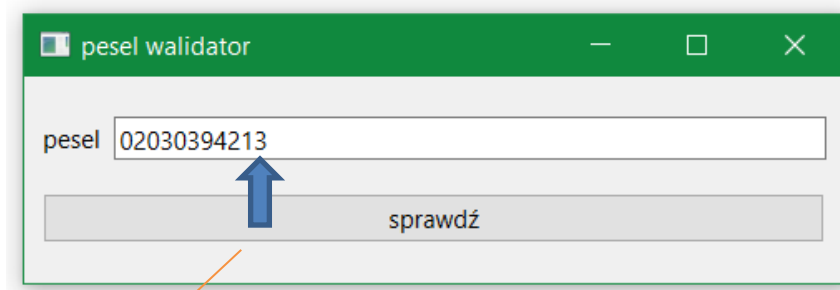
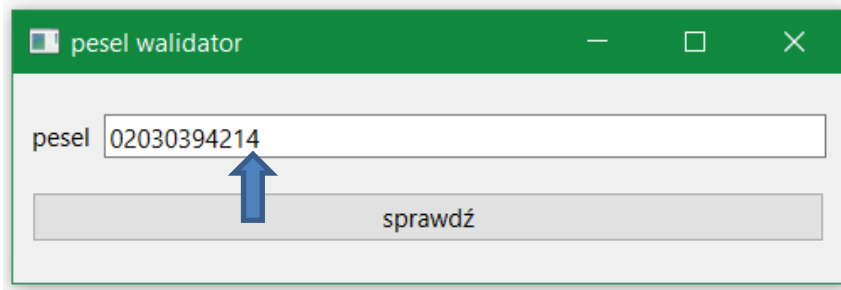
layout Formularz

A diagram illustrating a grid layout. It features a light gray rectangular container with a dotted grid background. The layout is organized into four rows and three columns. The first row contains three gray buttons, each labeled 'PushButton'. The second row contains three radio buttons, each labeled 'RadioButton' followed by a small circle icon. The third row contains two checkboxes, each labeled 'CheckBox' followed by a small square icon, and a gray button labeled 'PushButton' in the third column. The fourth row contains two checkboxes, each labeled 'CheckBox' followed by a small square icon, and a gray button labeled 'PushButton' in the third column.

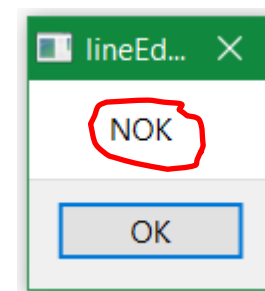
layout Siatka

# pesel walidator

Zaprojektuj aplikację, która będzie weryfikować poprawność numeru pesel.



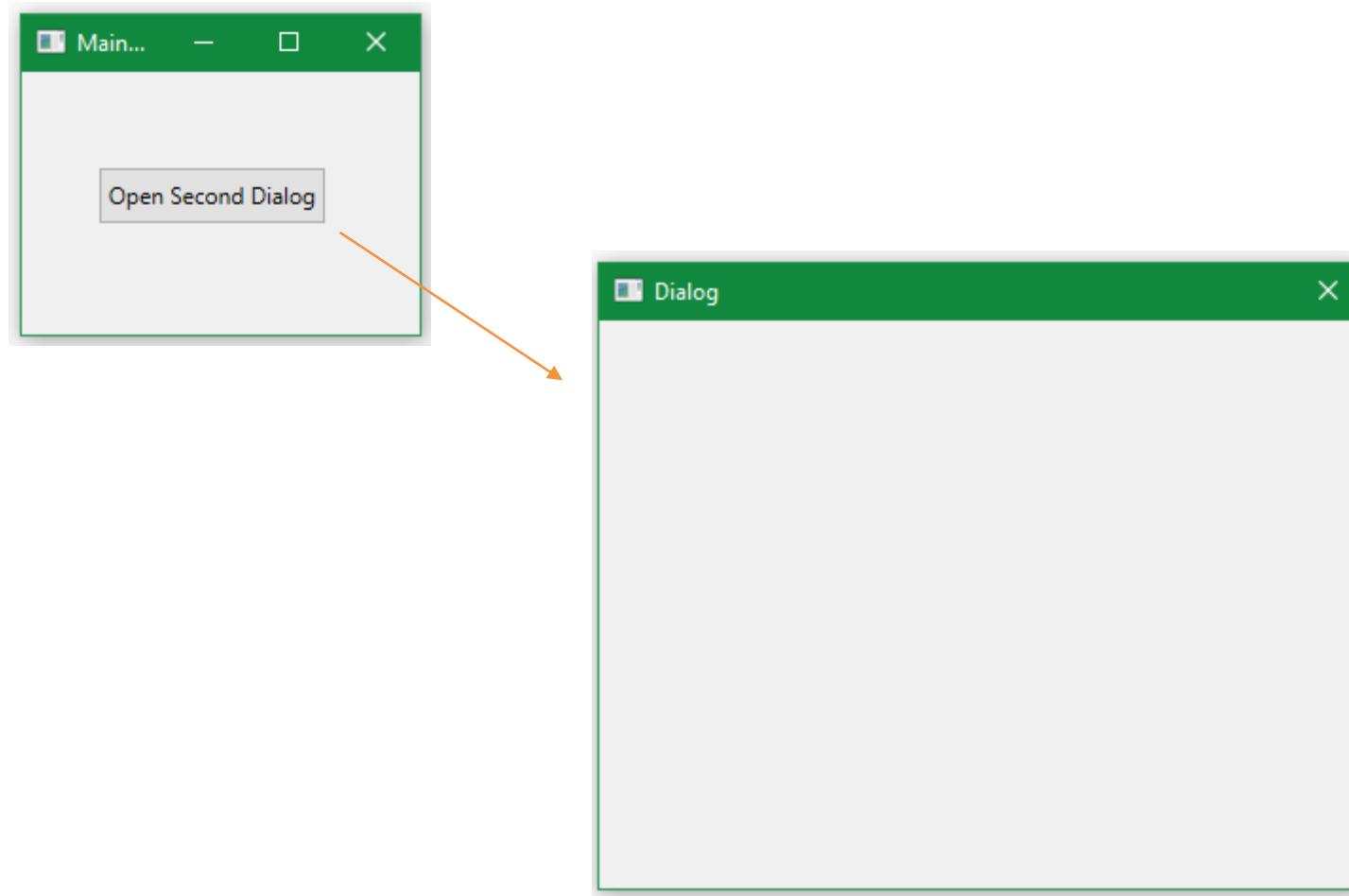
zła wartość cyfry kontrolnej  
powoduje błąd weryfikacji



budowa numeru pesel

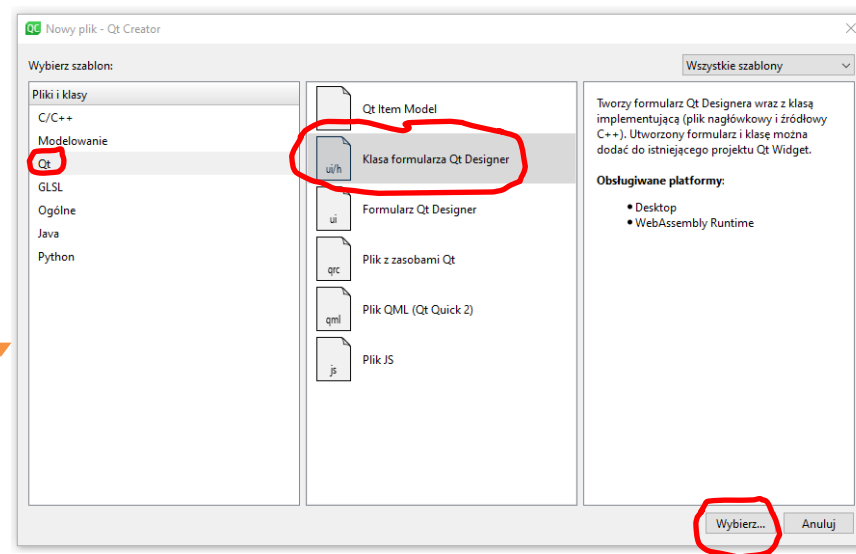
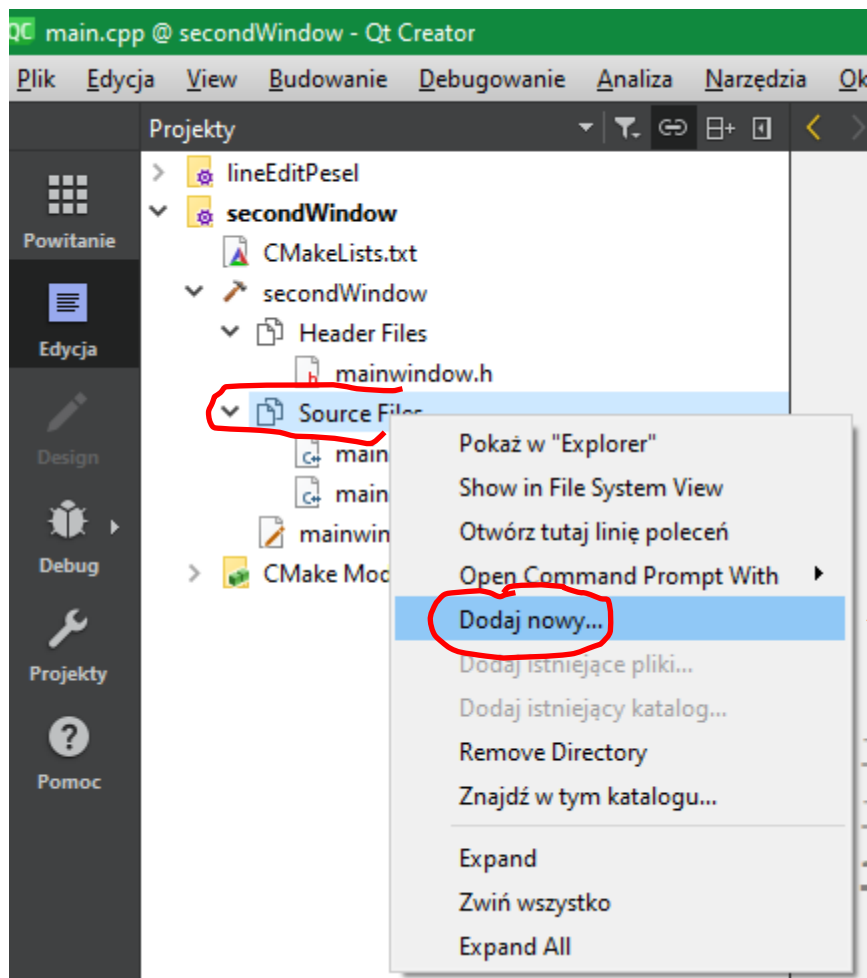


# second window

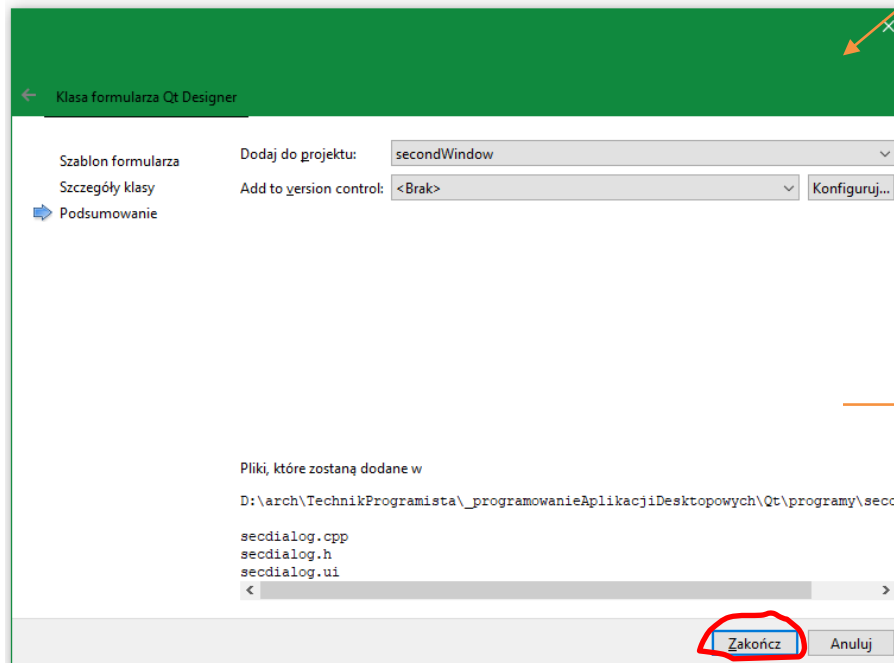
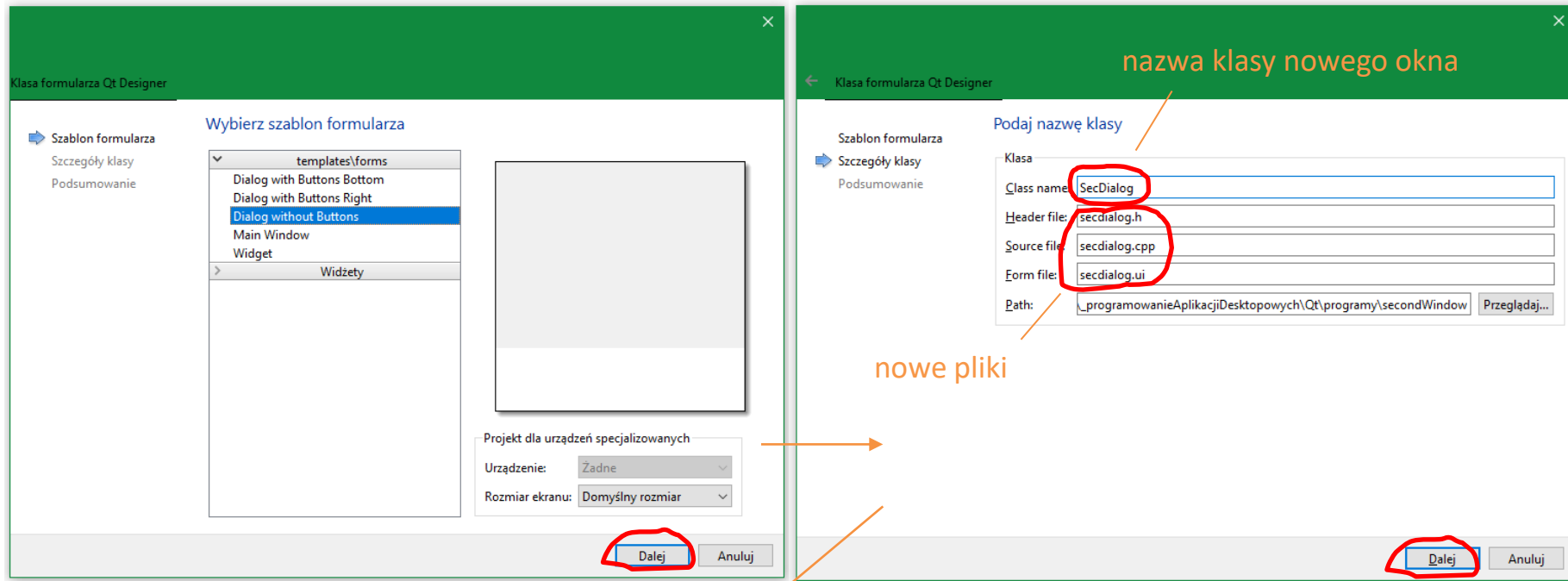


# second window

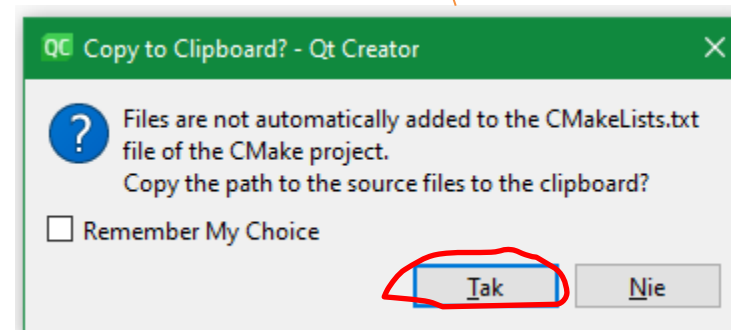
aby dodać nowe okno należy wygenerować nowe pliki



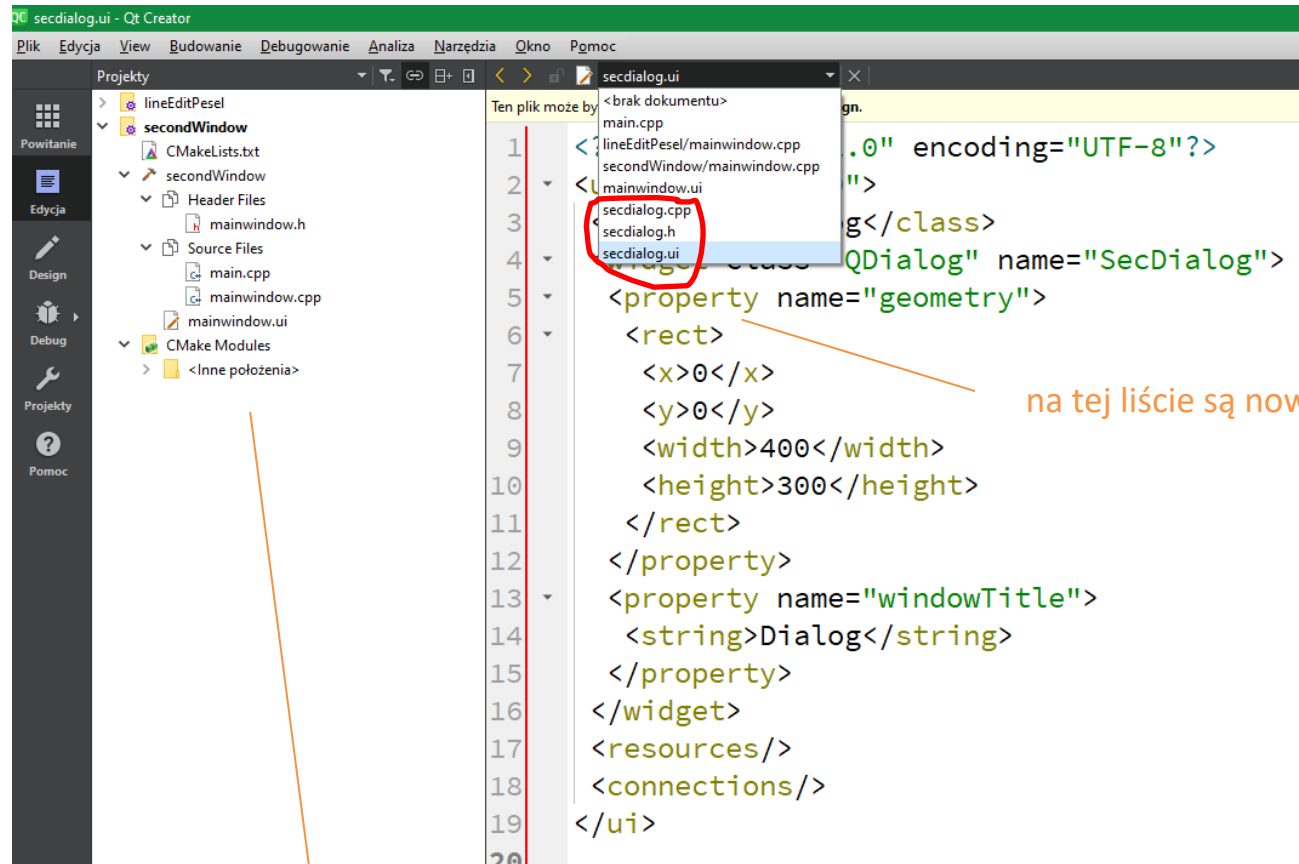
# second window



nazwy nowych plików nie zostały dodane do CMakeLists.txt (aby stworzone pliki pojawiły się w projekcie należy wpisać ich nazwy do pliku CMakeLists.txt)



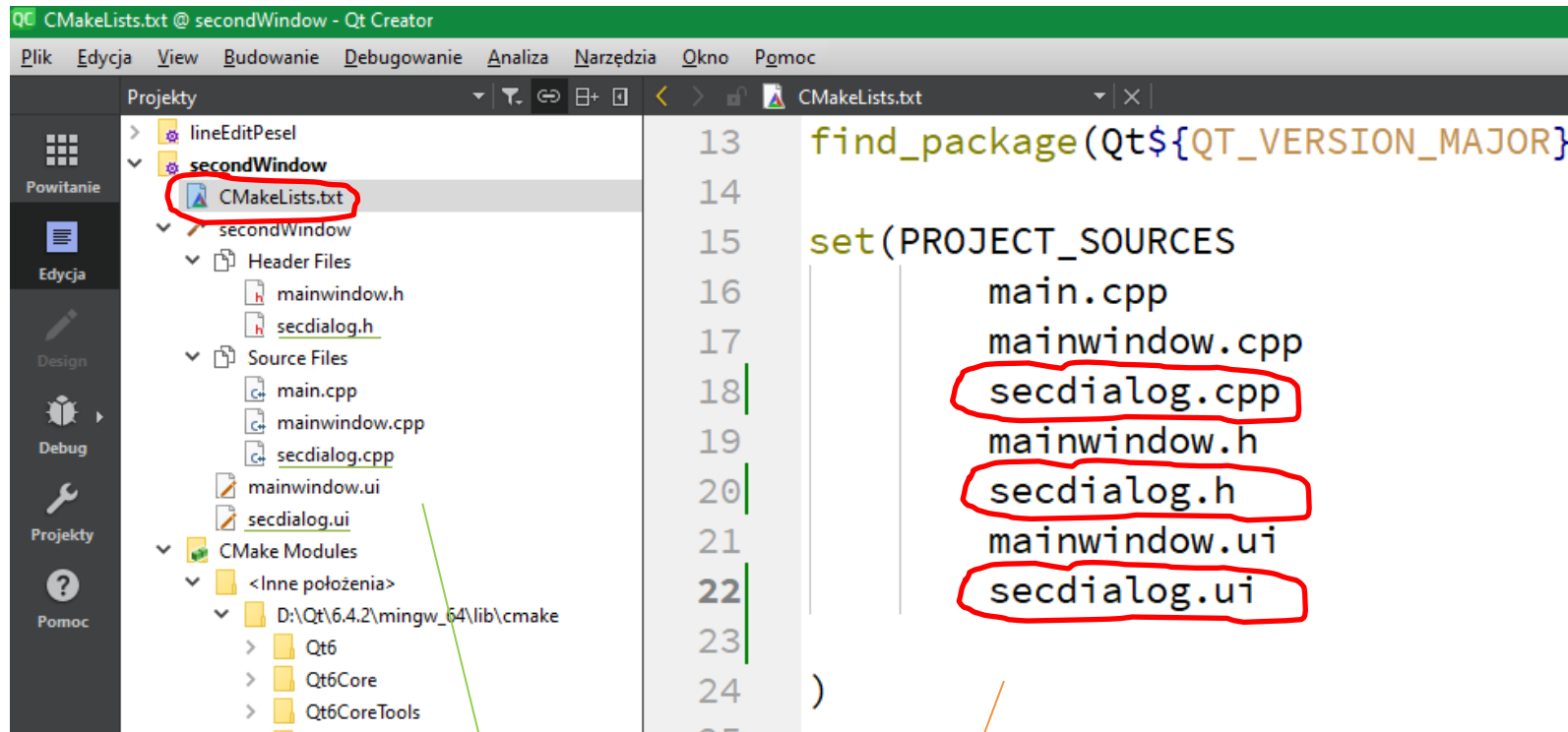
# second window



na tej liście są nowe pliki

brak nowych plików w projekcie !?

# second window



aby stworzone pliki pojawiły się w projekcie należy je wpisać do pliku CMakeLists.txt

nowe pliki pojawiły się w projekcie



# second window

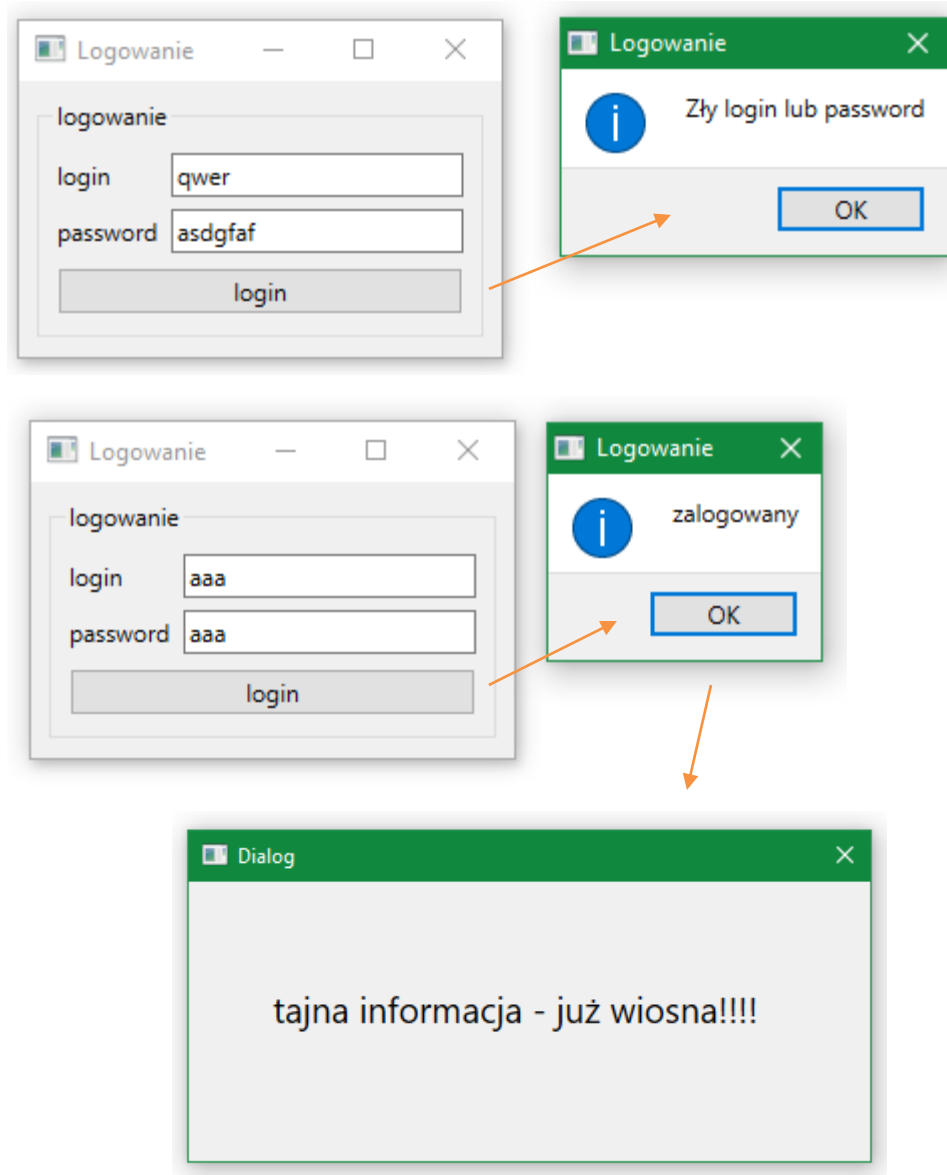
na głównym oknie (mainwindow) wstawiamy przycisk otwierający drugie okno. Generujemy slot połączony z sygnałem kliknięcia przycisku (prawy/przejdź do slotu)

```
1 #include "mainwindow.h"
2 #include "../ui_mainwindow.h"
3 #include "secdialog.h" // wstawiamy plik nagłówkowy nowego okna
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     // tworzymy obiekt nowego okna
21     SecDialog secDialog;
22     // ustawiamy nowe okna jako modalne
23     secDialog.setModal(true);
24     // uruchamiamy nowe okno
25     secDialog.exec();
26 }
```

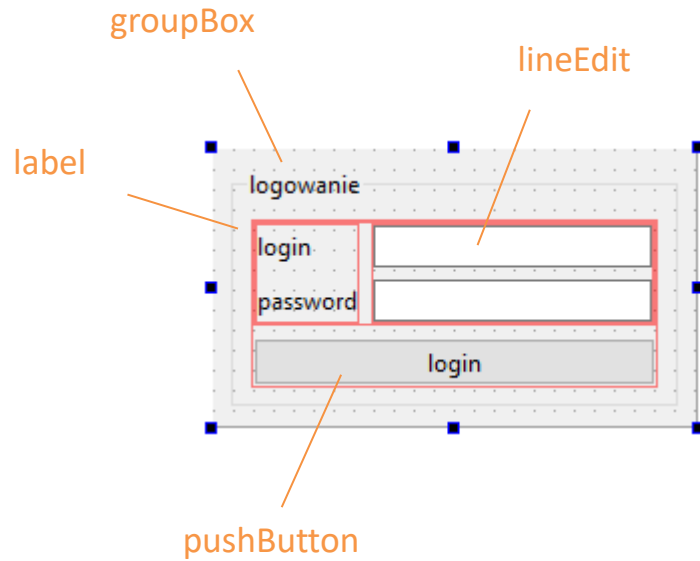
slot połączony z sygnałem kliknięcia przycisku

okno modalne blokuje okno główne programu (tak jak okno wyłączenia systemu operacyjnego)

# login

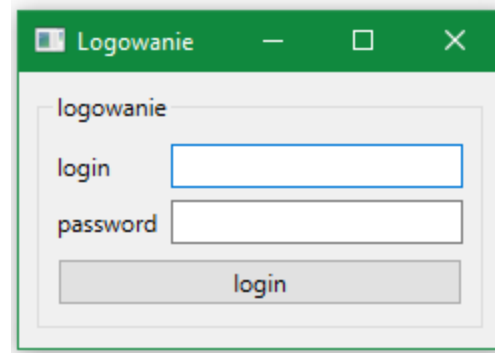


# login



Obiekt	Klasa
MainWindow	QMainWindow
centralwidget	QWidget
groupBox	QGroupBox
verticalLayout_3	QVBoxLayout
horizontalLayout	QHBoxLayout
verticalLayout	QVBoxLayout
label	QLabel
label_2	QLabel
verticalLayout_2	QVBoxLayout
lineEditLogin	QLineEdit
lineEditPassword	QLineEdit
pushButton	QPushButton

zmiana nazwy pól lineEdit



# login

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QMessageBox>
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     QString login = ui->lineEditLogin->text();
21     QString password = ui->lineEditPassword->text();
22
23     if (login == "aaa" && password == "aaa"){
24         QMessageBox::information(this, "Logowanie", "zalogowany");
25     }else{
26         QMessageBox::information(this, "Logowanie", "Zły login lub password");
27     }
28 }
```

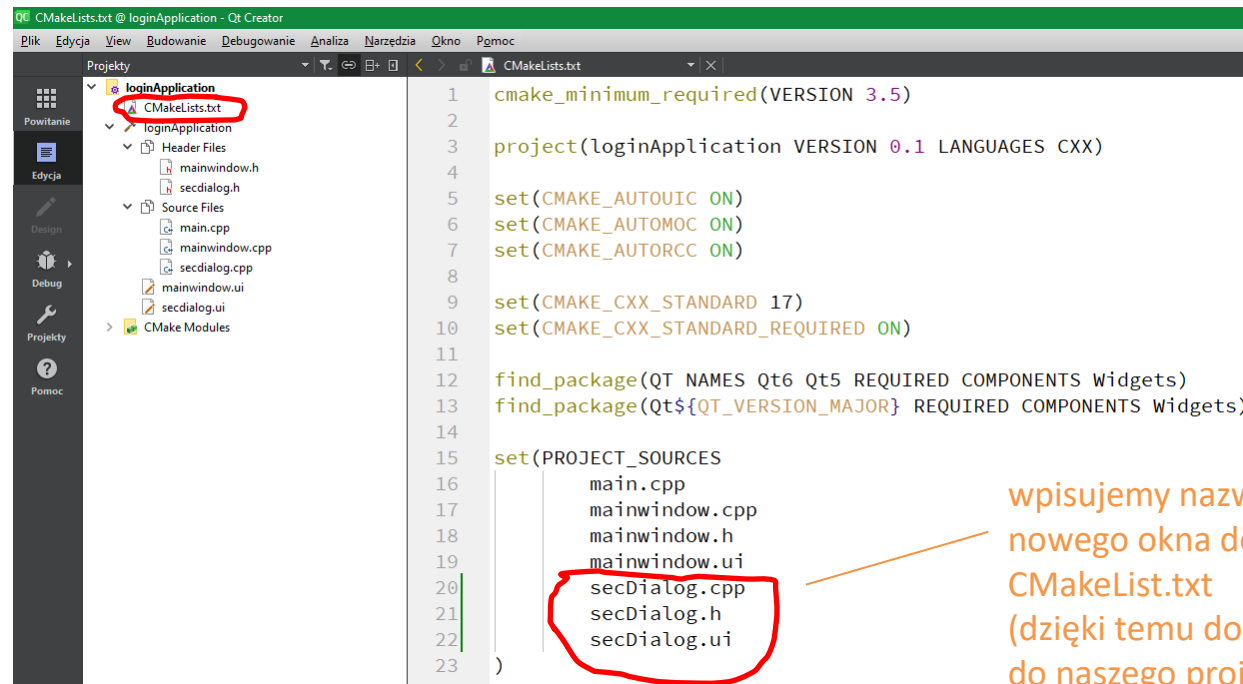
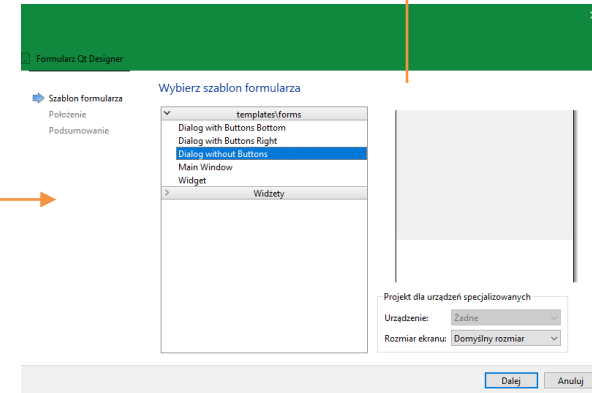
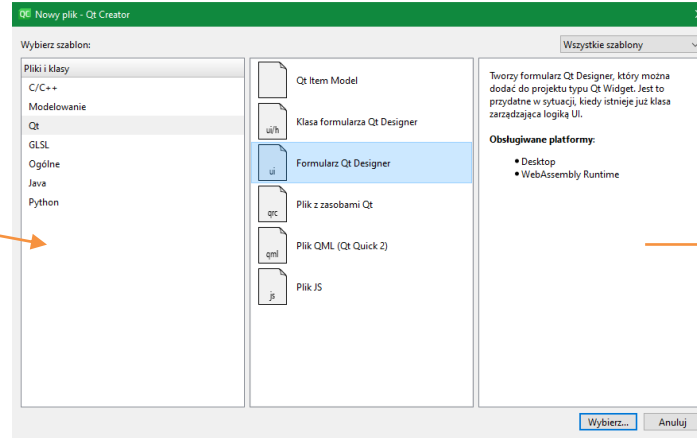
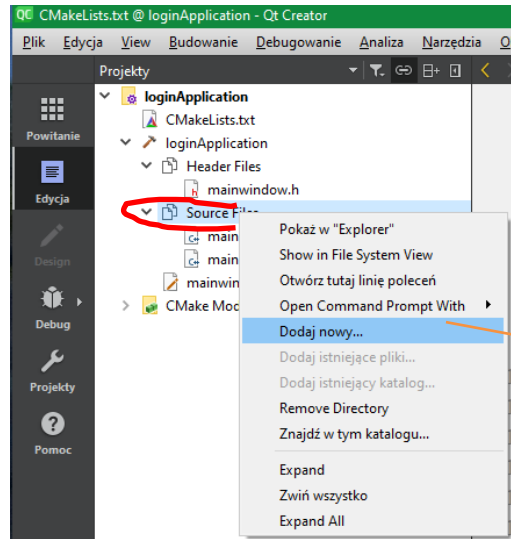
dodajemy bibliotekę dla QMessageBox

slot połączony z sygnałem kliknięcia przycisku

# login

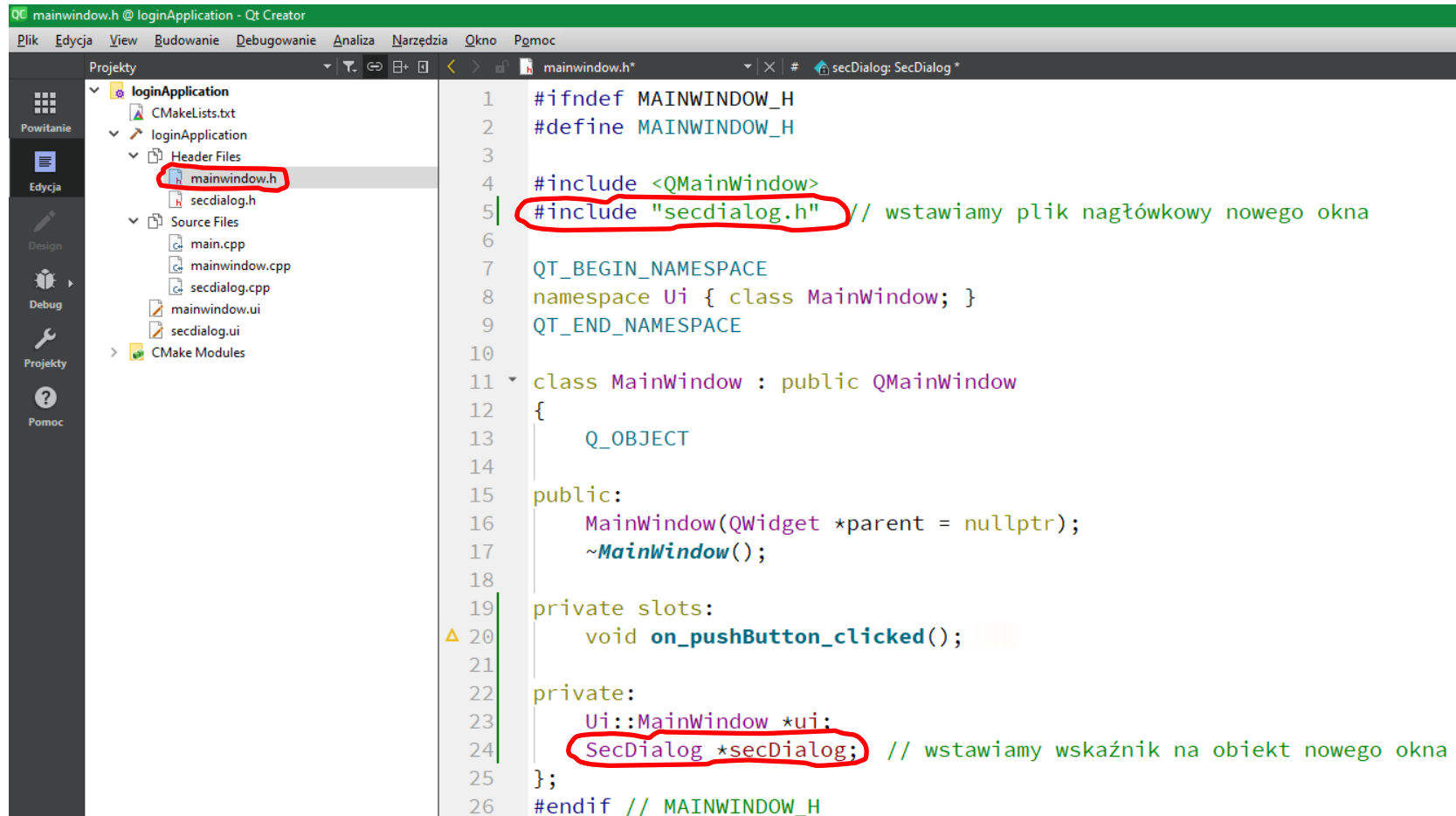
tworzymy  
nowe okno

itd. ...



wpisujemy nazwy plików  
nowego okna do  
CMakeList.txt  
(dzięki temu dodamy pliki  
do naszego projektu)

# login



```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include "secdialog.h" // wstawiamy plik nagłwkowy nowego okna
6
7  QT_BEGIN_NAMESPACE
8  namespace Ui { class MainWindow; }
9  QT_END_NAMESPACE
10
11 class MainWindow : public QMainWindow
12 {
13     Q_OBJECT
14
15 public:
16     MainWindow(QWidget *parent = nullptr);
17     ~MainWindow();
18
19 private slots:
20     void on_pushButton_clicked();
21
22 private:
23     Ui::MainWindow *ui;
24     SecDialog *secDialog; // wstawiamy wskaźnik na obiekt nowego okna
25 };
26 #endif // MAINWINDOW_H
```

odajemy nowe okno od klasy MainWindow, w tym celu w pliku nagłwkowym klasy MainWindow musimy zaimportować plik nagłwkowy nowego okna oraz dodać nowe prywatne pole

# login

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QMessageBox>
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 MainWindow::~MainWindow()
13 {
14     delete ui;
15 }
16
17
18 void MainWindow::on_pushButton_clicked()
19 {
20     QString login = ui->lineEditLogin->text();
21     QString password = ui->lineEditPassword->text();
22
23     if (login == "aaa" && password == "aaa"){
24         QMessageBox::information(this, "Logowanie", "zalogowany");
25         hide(); // chowa okno glowne programu
26         secDialog = new SecDialog(this); // tworzymy obiekt nowego okna za pomoca operatora new
27         // (new zwraca adres nowego okna w pamieci komputera, czyli wskaźnik)
28         //secDialog->setModal(true); // mozemy zablokować okno glowne programu
29         secDialog->show(); // pokazujemy nowe okno
30     }else{
31         QMessageBox::information(this, "Logowanie", "Zły login lub password");
32     }
33 }
```

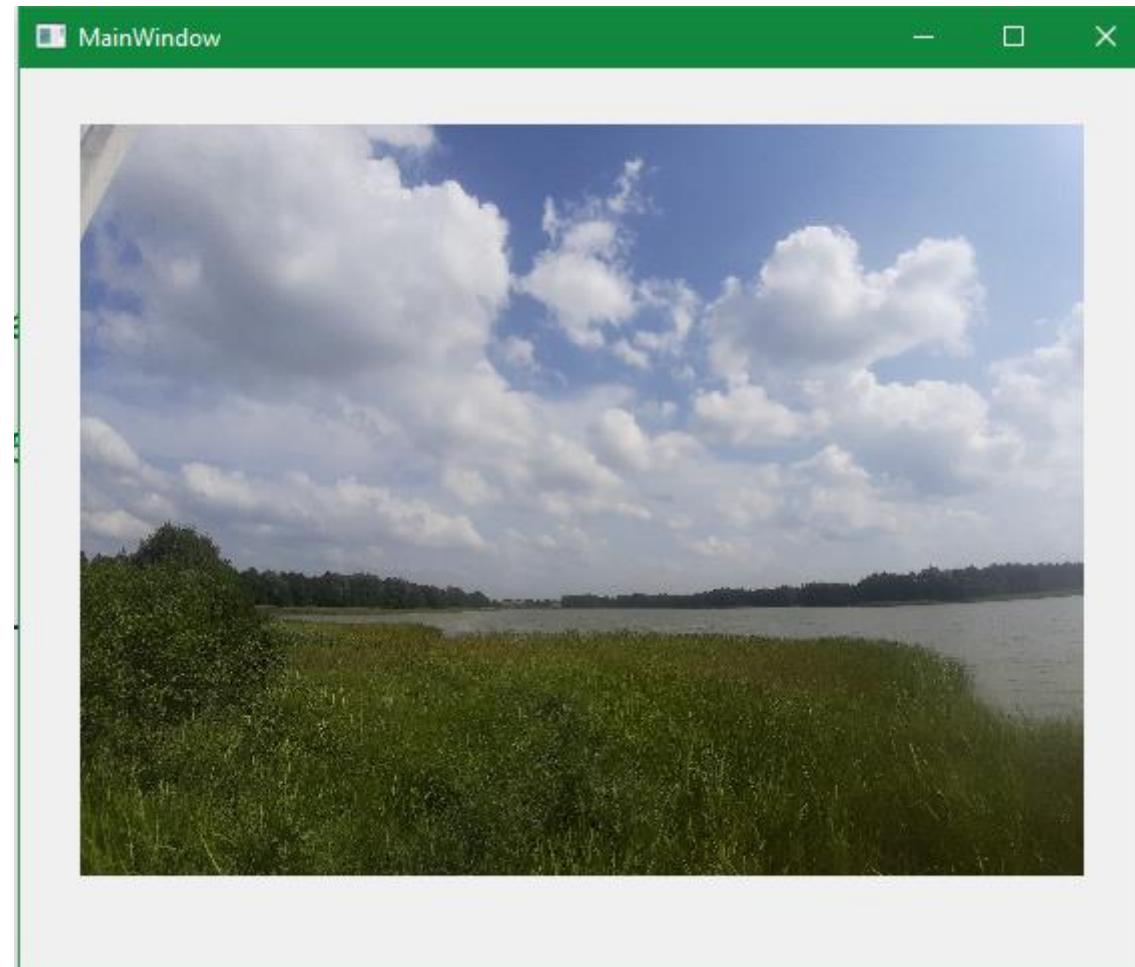
nie musimy dodawać pliku nagłówkowego nowego okna, gdyż dodaliśmy go w pliku mainwindow.h który inkludujemy.

slot połączony z sygnałem kliknięcia przycisku

informacja o zalogowaniu

obsługa nowego okna

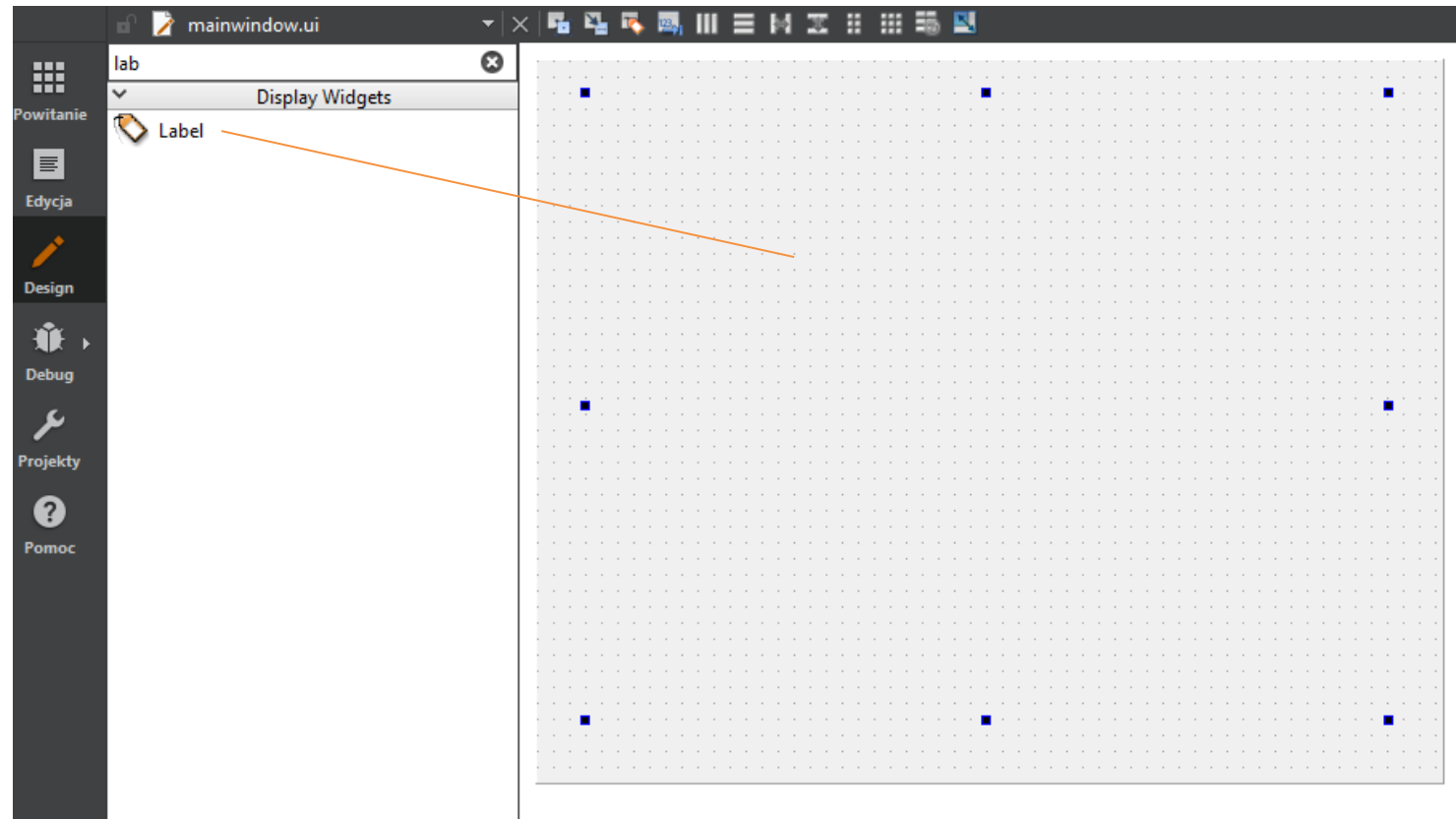
# picture





# picture

wstawiamy obrazek za  
pomocą komponentu  
Label



# picture

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QPixmap>
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10    QPixmap picture("D:/mazury.jpg");
11    //ui->label->setPixmap(picture);
12    //ui->label->setPixmap(picture.scaled(600,500,Qt::KeepAspectRatio));
13    int labelWidth = ui->label->width();
14    int labelHeight = ui->label->height();
15    ui->label->setPixmap(picture.scaled(labelWidth,labelHeight,Qt::KeepAspectRatio));
16 }
17
18 MainWindow::~MainWindow()
19 {
20     delete ui;
21 }
```

ścieżka bezwzględna obrazka!  
(trzeba dodać folder z zasobami)

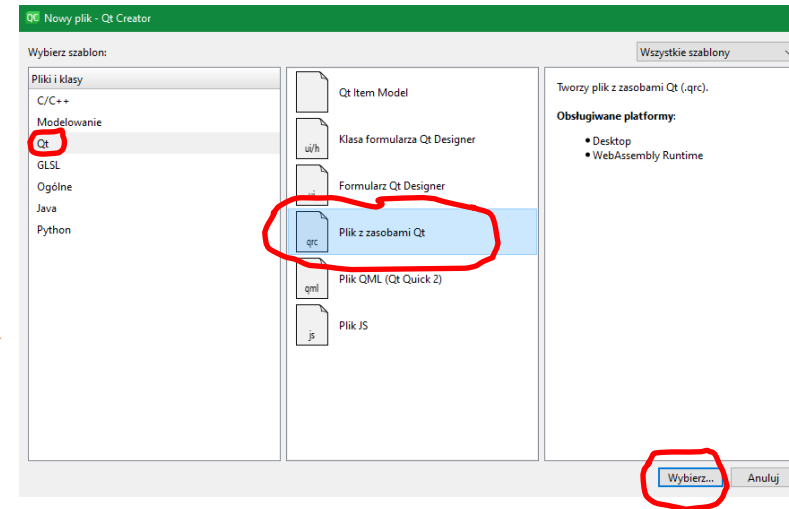
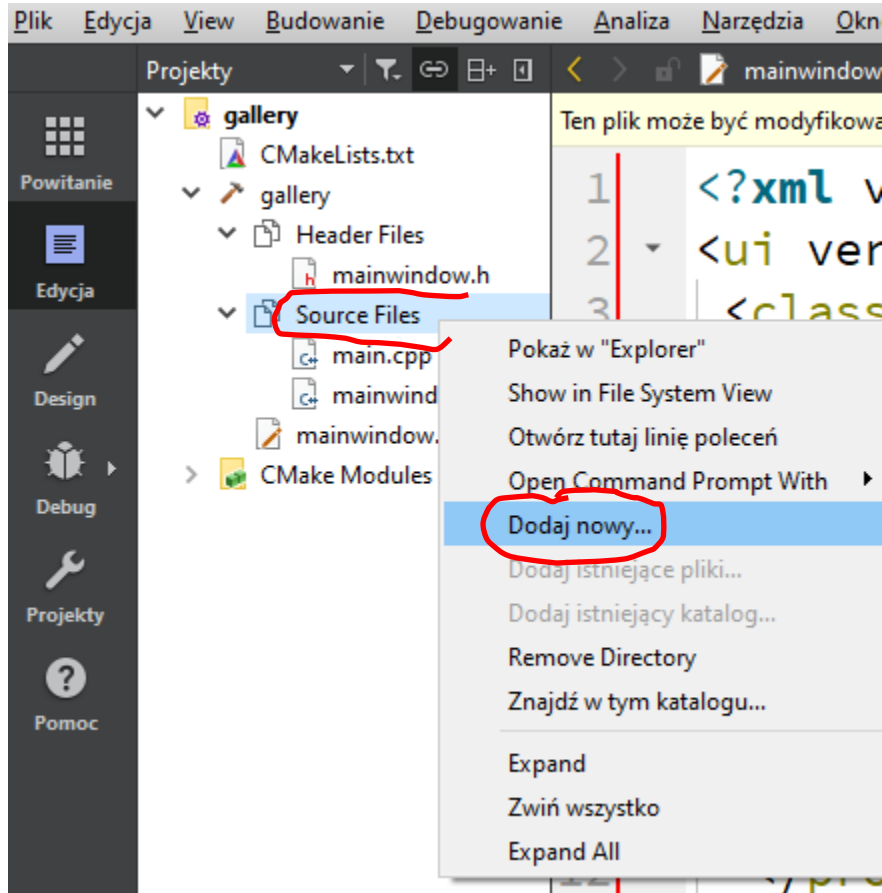
najprostszy sposób

sposób z podaniem wymiarów

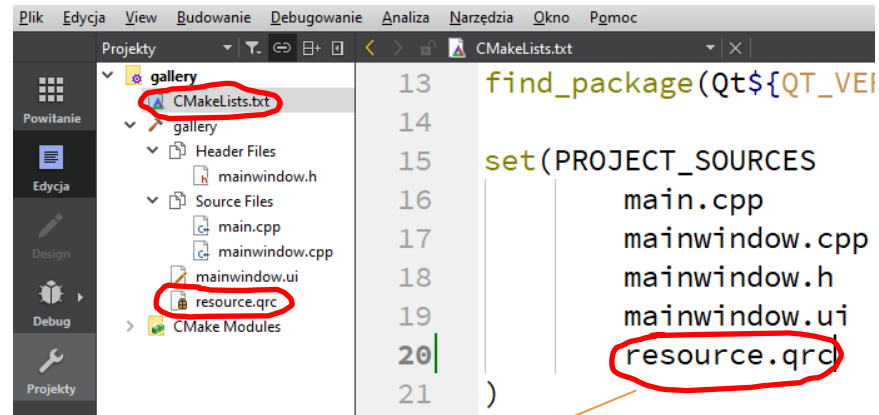
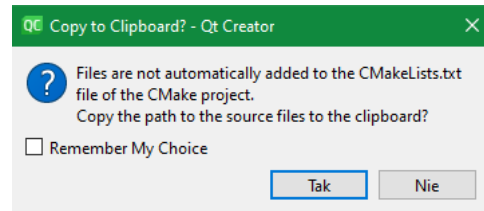
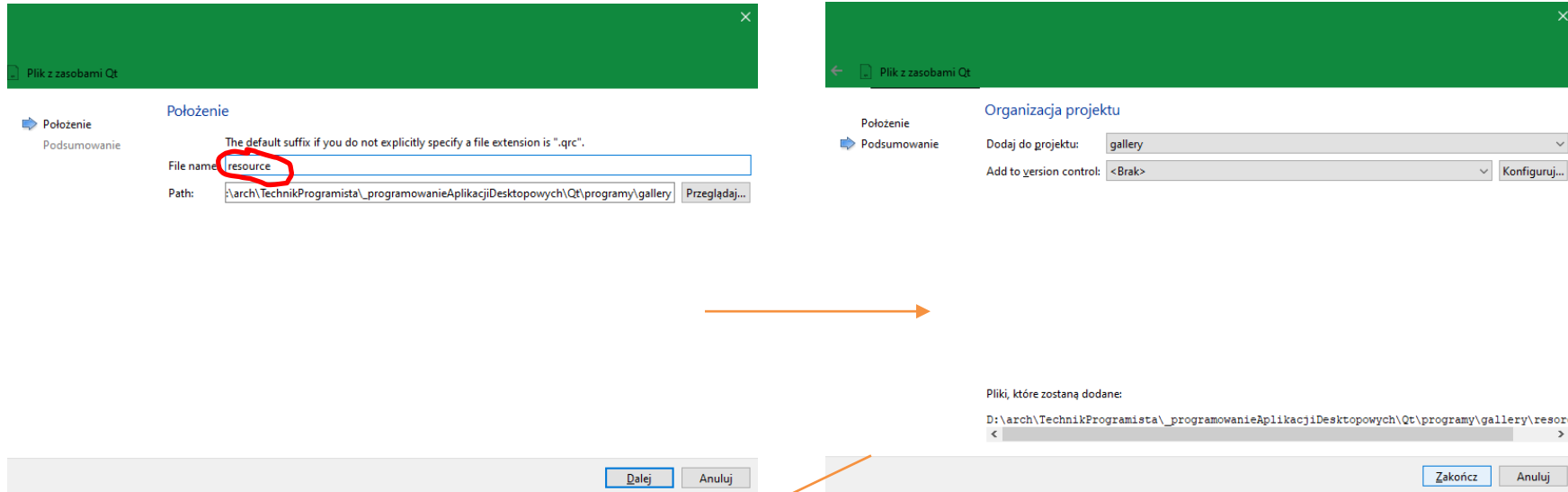
wymiary obrazka dostosowują się do wymiarów labela

# picture plik z zasobami

dodajemy do projektu plik resource.qrc  
z zasobami, aby uniknąć podawania  
ścieżki bezwzględnej obrazka

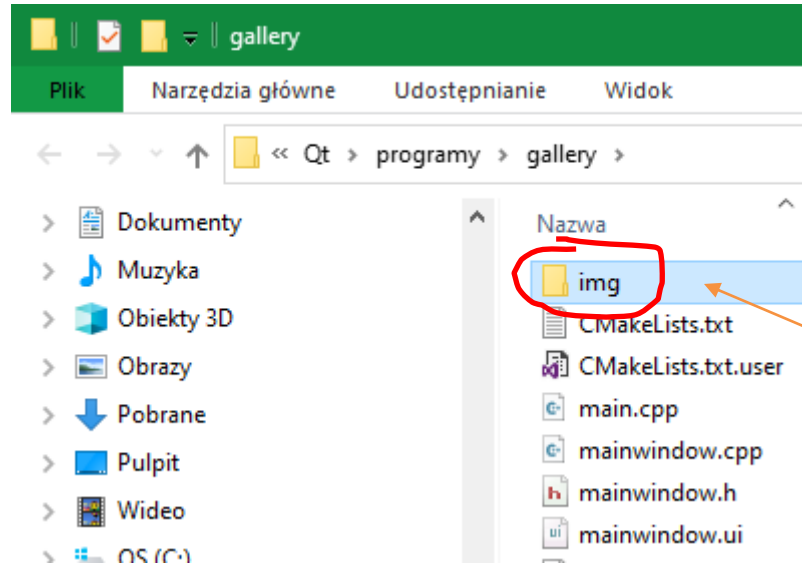


# picture plik z zasobami

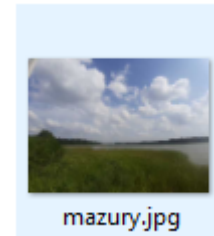


wpisujemy nazwę pliku z zasobami  
w pliku CMakeLists.txt

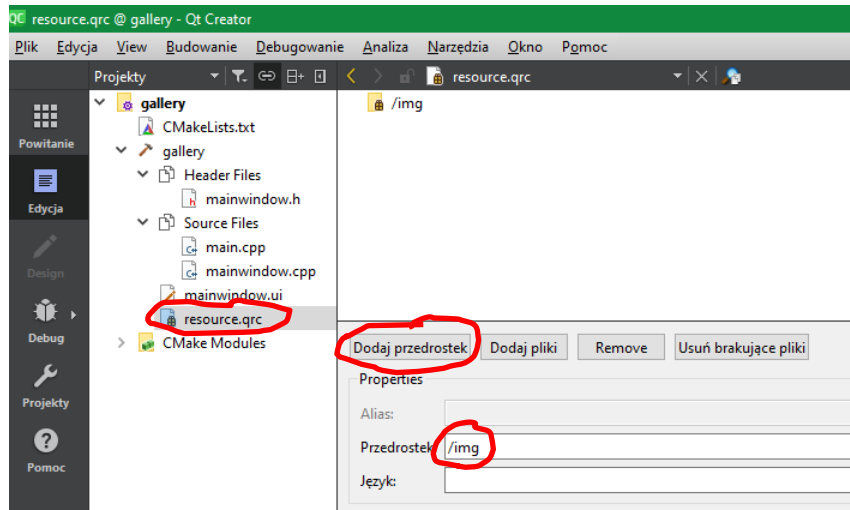
# picture plik z zasobami



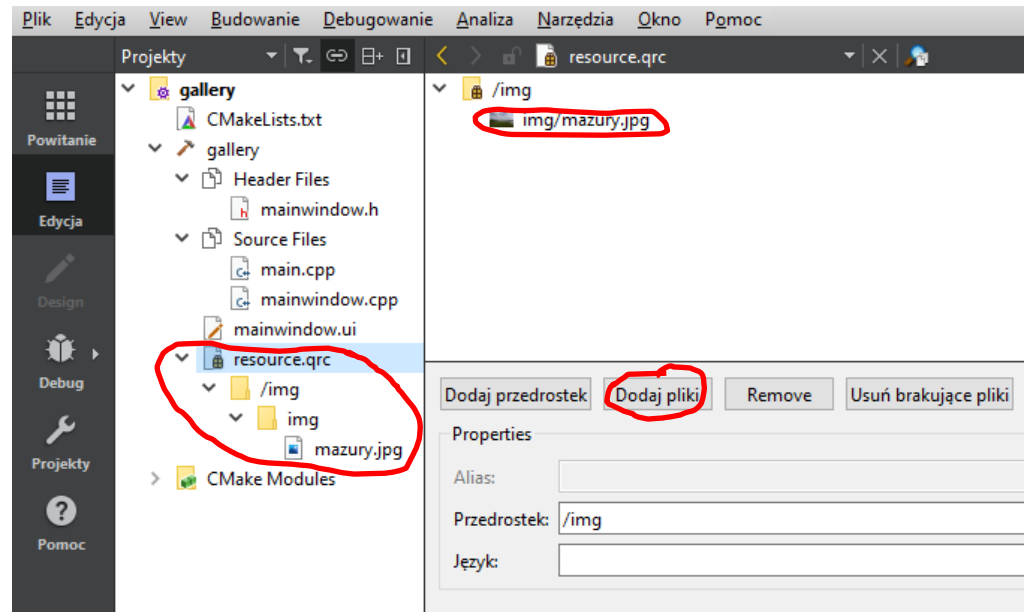
w katalogu projektu tworzymy  
folder z obrazkiem



# picture plik z zasobami

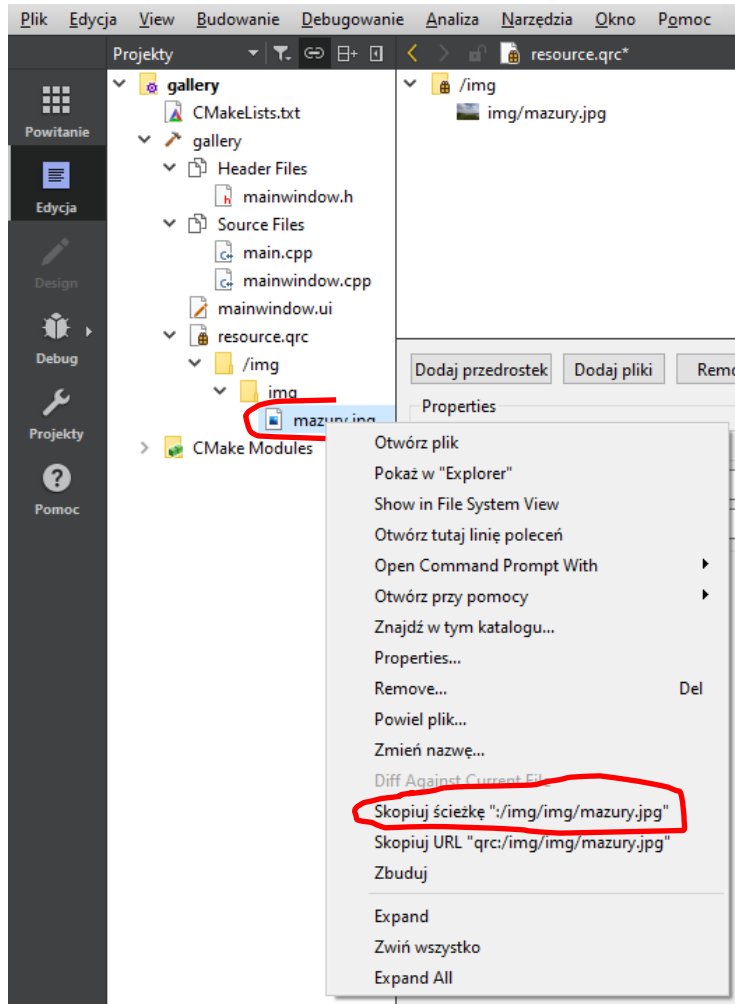


w pliku resource.qrc  
dodajemy przedrostek /img



a następnie dodajemy obrazek  
z katalogu img

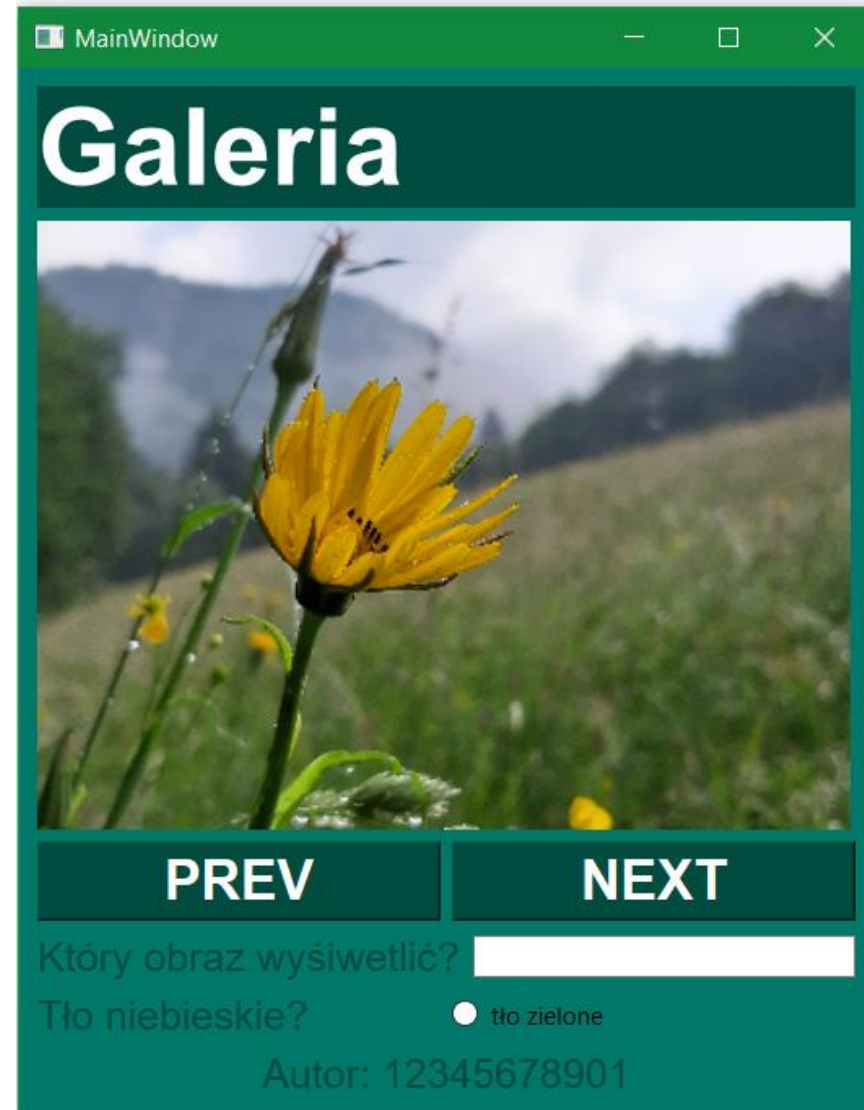
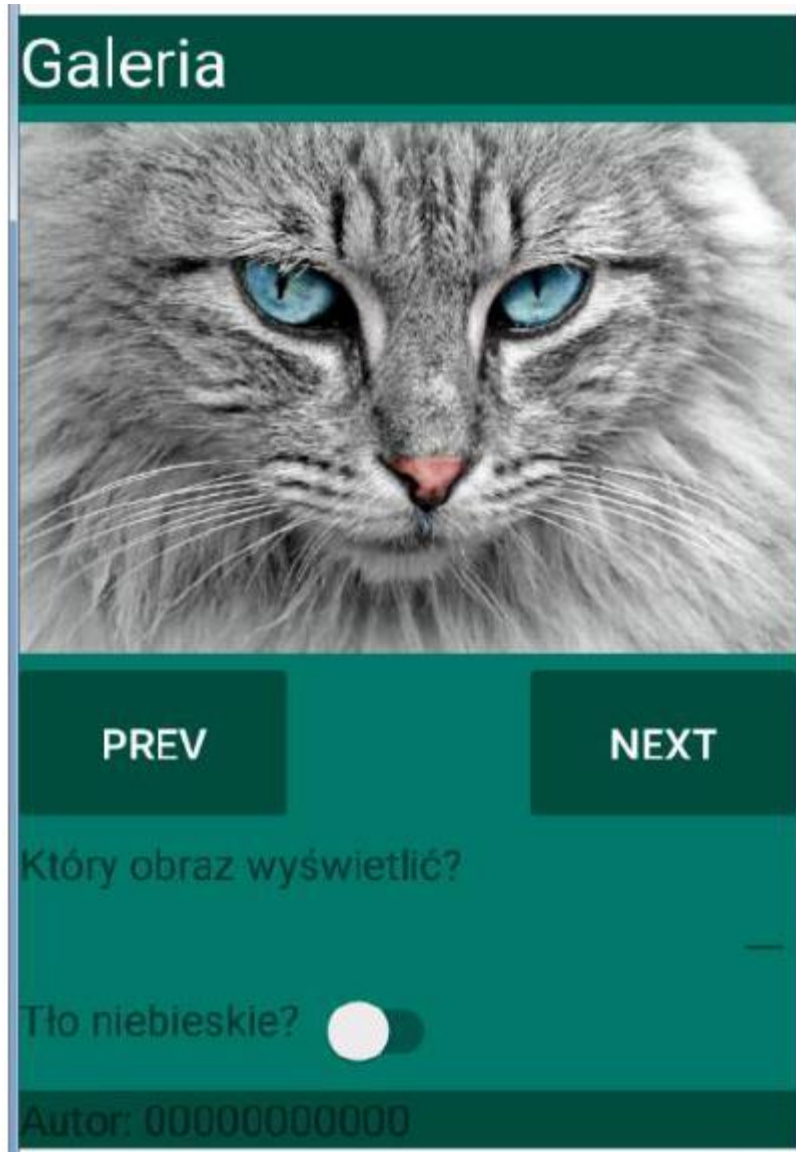
# picture plik z zasobami



```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    QPixmap picture(":/img/img/mazury.jpg");
    int labelWidth = ui->label->width();
    int labelHeight = ui->label->height();
    ui->label->setPixmap(picture.scaled(labelWidth, labelHeight, Qt::KeepAspectRatio));
}
```

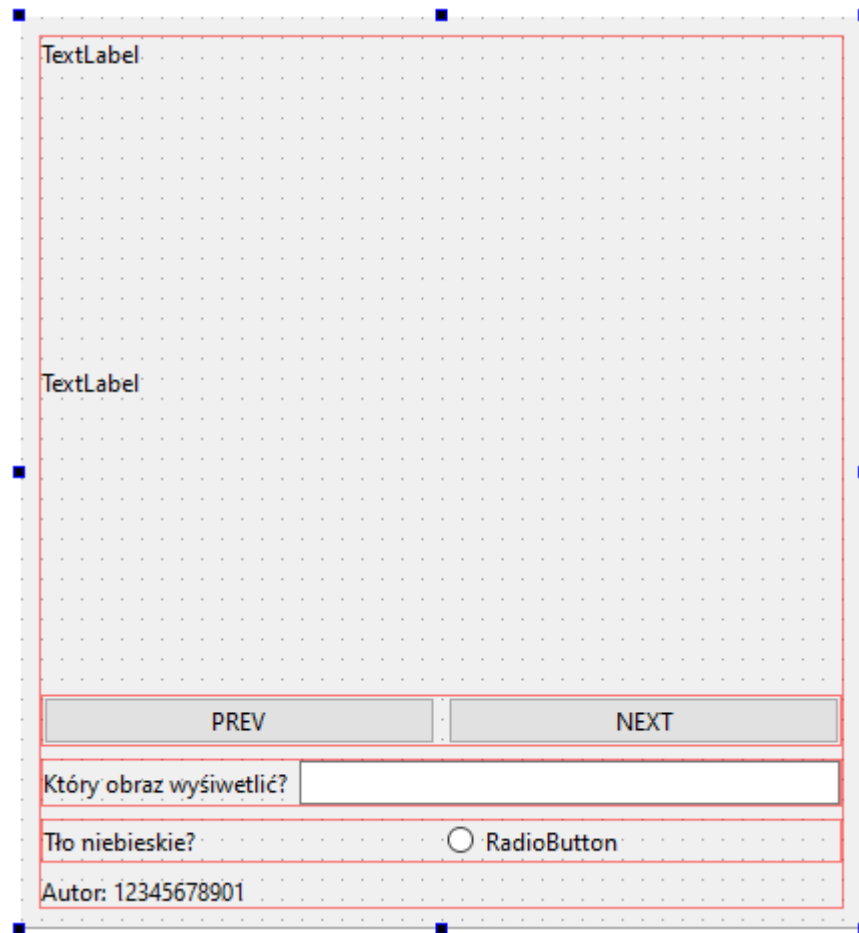
kopiujemy ścieżkę do zasobu i  
wklejamy na miejsce ścieżki  
bezwzględnej.

# galeria





# galeria



Obiekt	Klasa
MainWindow	QMainWindow
centralwidget	QWidget
verticalLayout	QVBoxLayout
horizontalLayout	QHBoxLayout
pushButtonNext	QPushButton
pushButtonPrev	QPushButton
horizontalLayout_2	QHBoxLayout
labelWhichPicture	QLabel
lineEditWhichPicture	QLineEdit
horizontalLayout_3	QHBoxLayout
labelBackground	QLabel
radioButtonBackground	QRadioButton
labelFooter	QLabel
labelHeader	QLabel
labelPicture	QLabel

# galeria

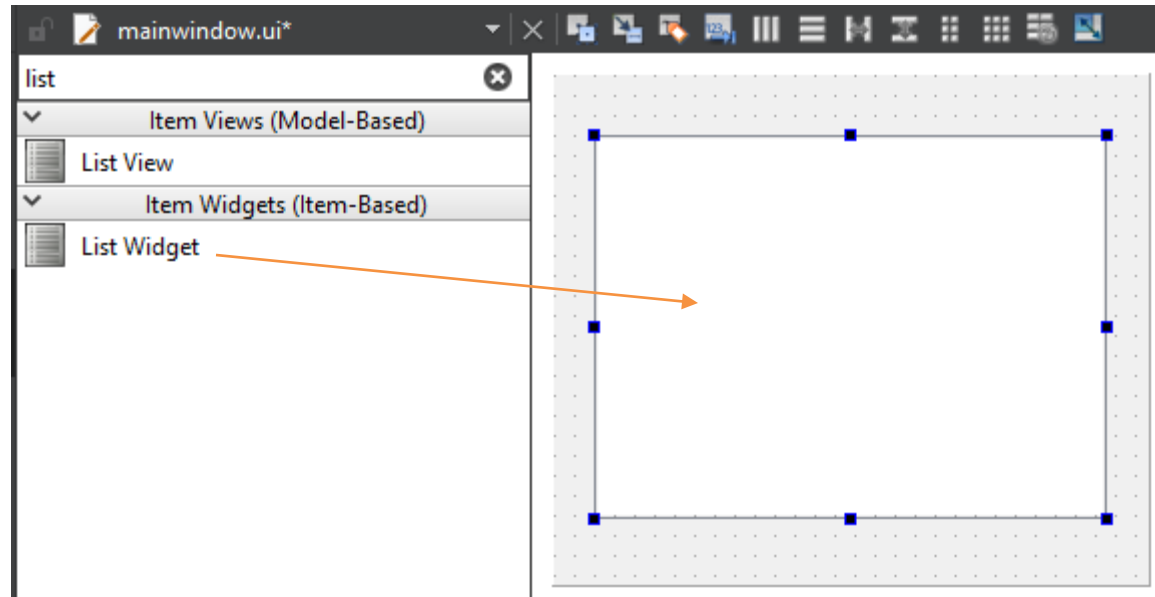
labelPicture : QLabel	
Właściwość	Wartość
▼ QObject	
<b>objectName</b>	labelPicture
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
> geometry	[(1, 23), 400 x 300]
▼ sizePolicy	[Fixed, Fixed, 0, 0]
Strategia pozioma	Fixed
Strategia pionowa	Fixed
Rozciąganie w poziomie	0
Rozciąganie w pionie	0
▼ minimumSize	400 x 300
Szerokość	400
Wysokość	300

labelHeader : QLabel	
Właściwość	Wartość
▼ QObject	
<b>objectName</b>	labelHeader
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
> geometry	[(1, 1), 514 x 98]
▼ sizePolicy	[Preferred, Fixed, 0, 0]
Strategia pozioma	Preferred
Strategia pionowa	Fixed

Właściwość	Wartość
▼ QObject	
<b>objectName</b>	labelFooter
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
> geometry	[(1, 553), 400 x 69]
▼ sizePolicy	[Preferred, Fixed, 0, 0]
Strategia pozioma	Preferred
Strategia pionowa	Fixed

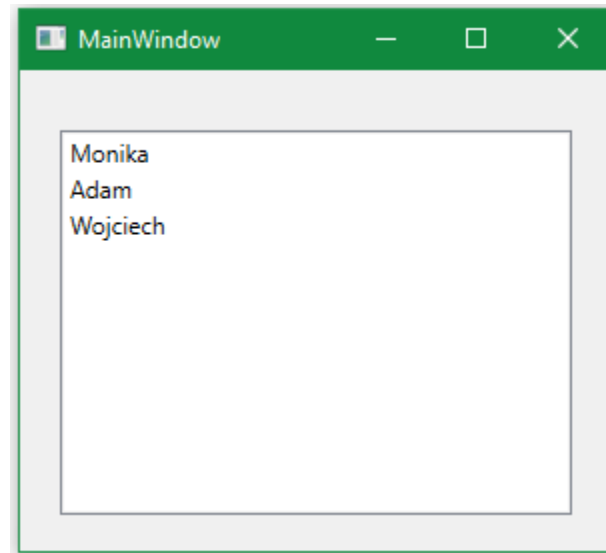
przykładowa konfiguracja labeli

# List Widget



# List Widget

```
Projekty
└─ listWidget
   └─ mainwindow.cpp
      1 #include "mainwindow.h"
      2 #include "./ui_mainwindow.h"
      3
      4 MainWindow::MainWindow(QWidget *parent)
      5     : QMainWindow(parent)
      6     , ui(new Ui::MainWindow)
      7 {
      8     ui->setupUi(this);
      9     ui->listWidget->addItem("Monika");
     10     ui->listWidget->addItem("Adam");
     11     ui->listWidget->addItem("Wojciech");
     12 }
     13
     14 MainWindow::~MainWindow()
     15 {
     16     delete ui;
     17 }
```



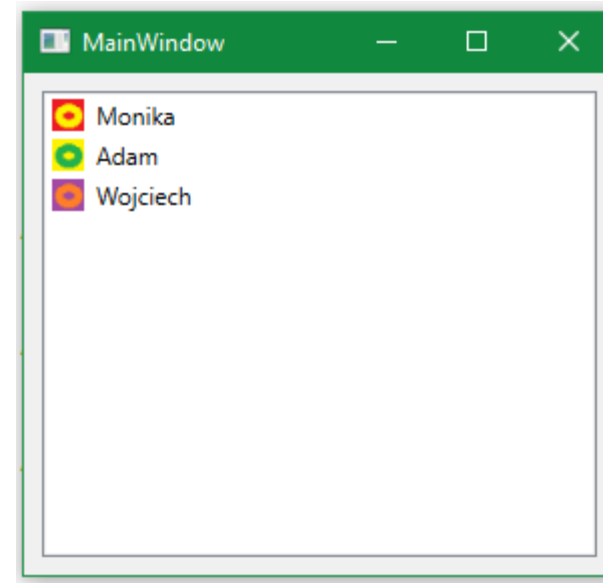
lista z imionami

# List Widget

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     QListWidgetItem *item1 = new QListWidgetItem(QIcon(":/pic/icon/icon1.png"), "Monika");
10    ui->listWidget->addItem(item1);
11    QListWidgetItem *item2 = new QListWidgetItem(QIcon(":/pic/icon/icon2.png"), "Adam");
12    ui->listWidget->addItem(item2);
13    QListWidgetItem *item3 = new QListWidgetItem(QIcon(":/pic/icon/icon3.png"), "Wojciech");
14    ui->listWidget->addItem(item3);
15 }
16
17 MainWindow::~MainWindow()
18 {
19     delete ui;
20 }
```

plik z zasobami w postaci ikon

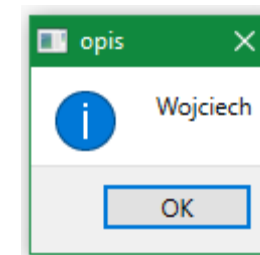
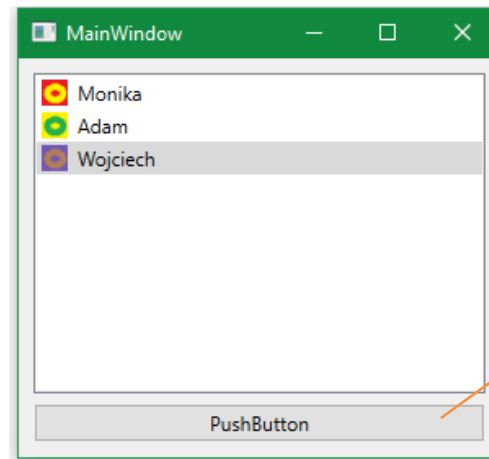
lista z imionami  
oraz ikonami



# List Widget

```
Projekty
├── listWidget
│   ├── CMakeLists.txt
│   └── listWidget
│       ├── Header Files
│       │   └── mainwindow.h
│       ├── Source Files
│       │   ├── main.cpp
│       │   └── mainwindow.cpp
│       └── mainwindow.ui
│           ├── resource.qrc
│           └── /pic
│               ├── icon
│               │   ├── icon1.png
│               │   ├── icon2.png
│               │   └── icon3.png
│           └── CMake Modules
└── CMake Modules

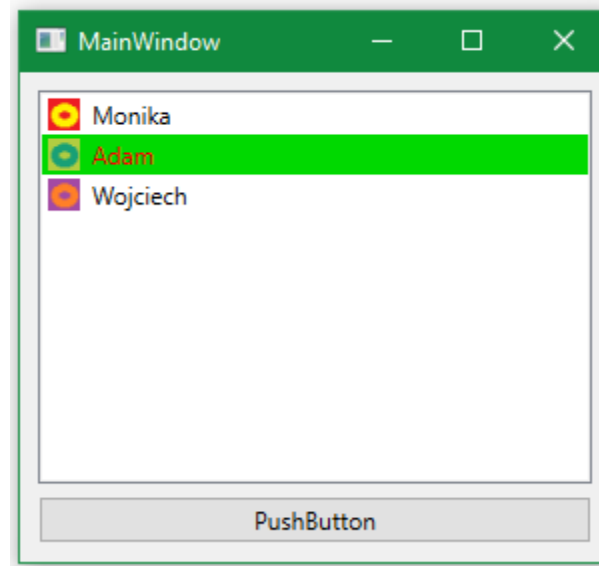
mainwindow.cpp
1  #include "mainwindow.h"
2  #include "./ui_mainwindow.h"
3  #include <QMessageBox>
4
5  MainWindow::MainWindow(QWidget *parent)
6      : QMainWindow(parent)
7      , ui(new Ui::MainWindow)
8  {
9
10     ui->setupUi(this);
11     QListWidgetItem *item1 = new QListWidgetItem(QIcon(":/pic/icon/icon1.png"), "Monika");
12     ui->listWidget->addItem(item1);
13     QListWidgetItem *item2 = new QListWidgetItem(QIcon(":/pic/icon/icon2.png"), "Adam");
14     ui->listWidget->addItem(item2);
15     QListWidgetItem *item3 = new QListWidgetItem(QIcon(":/pic/icon/icon3.png"), "Wojciech");
16     ui->listWidget->addItem(item3);
17 }
18
19 ~MainWindow()
20 {
21     delete ui;
22 }
23
24 void MainWindow::on_pushButton_clicked()
25 {
26     QMessageBox::information(this, "opis", ui->listWidget->currentItem()->text());
27 }
```



po wybraniu imienia i kliknięciu na przycisk pojawia się okienko informacyjne z wybranym imieniem

# List Widget

```
23  
24 void MainWindow::on_pushButton_clicked()  
25 {  
26     //QMessageBox::information(this, "opis", ui->listWidget->currentItem()->text());  
27     ui->listWidget->currentItem()->setBackground(Qt::green);  
28     ui->listWidget->currentItem()->setForeground(Qt::red);  
29 }  
30  
31
```

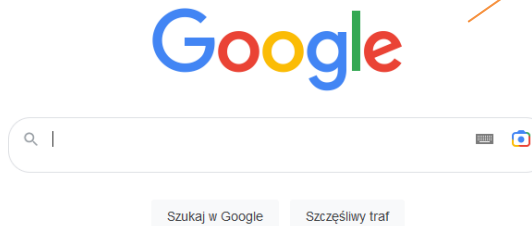
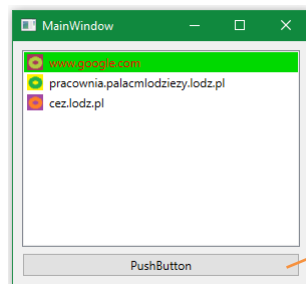


po wybraniu imienia i kliknięciu  
na przycisk zmieniają się kolory  
tła i czcionki wybranego imienia

# List Widget

```
Projekty
└─ listWidget
   └─ CMakeLists.txt
      └─ listWidget
         └─ Header Files
            └─ mainwindow.h
               └─ Source Files
                  └─ main.cpp
                     └─ mainwindow.cpp
                        └─ resource.qrc
                           └─ /pic
                              └─ icon
                                 └─ icon1.png
                                    └─ icon2.png
                                       └─ icon3.png
└─ CMake Modules

mainwindow.cpp
1  #include "mainwindow.h"
2  #include "./ui_mainwindow.h"
3  #include <QMessageBox>
4  #include <QDesktopServices>
5  #include <QUrl>
6
7  MainWindow::MainWindow(QWidget *parent)
8      : QMainWindow(parent)
9      , ui(new Ui::MainWindow)
10 {
11     ui->setupUi(this);
12     QListWidgetItem *item1 = new QListWidgetItem(QIcon(":/pic/icon/icon1.png"), "www.google.com");
13     ui->listWidget->addItem(item1);
14     QListWidgetItem *item2 = new QListWidgetItem(QIcon(":/pic/icon/icon2.png"), "pracownia.palacmlodziezy.lodz.pl");
15     ui->listWidget->addItem(item2);
16     QListWidgetItem *item3 = new QListWidgetItem(QIcon(":/pic/icon/icon3.png"), "cez.lodz.pl");
17     ui->listWidget->addItem(item3);
18 }
19
20 MainWindow::~MainWindow()
21 {
22     delete ui;
23 }
24
25
26 void MainWindow::on_pushButton_clicked()
27 {
28     //QMessageBox::information(this, "opis", ui->listWidget->currentItem()->text());
29     ui->listWidget->currentItem()->setBackground(Qt::green);
30     ui->listWidget->currentItem()->setForeground(Qt::red);
31     //Const QString link = "http://www.google.com";
32     const QString link = "http://" + ui->listWidget->currentItem()->text();
33     QDesktopServices::openUrl(QUrl(link));
34 }
35
```

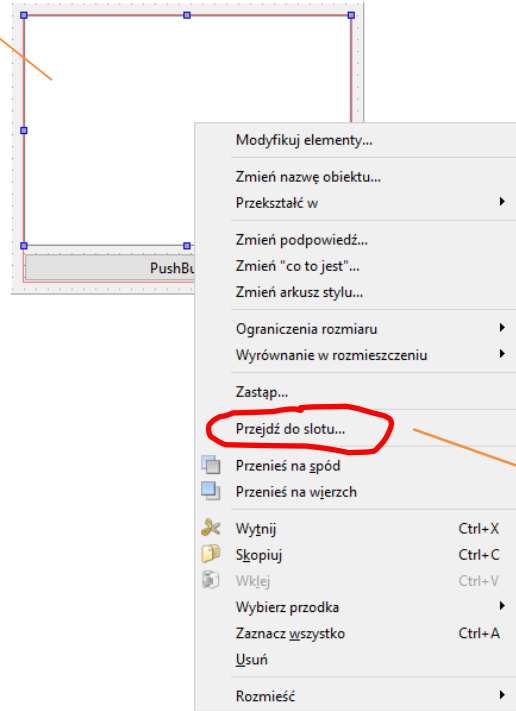


po wybraniu pozycji z listy i kliknięciu na przycisk otwiera się przeglądarka z wybranym adresem

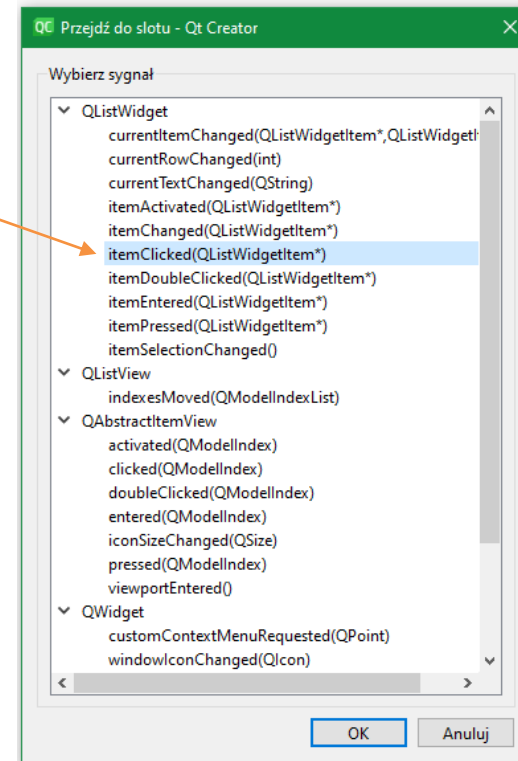


# List Widget

listWidget



wyberamy slot:  
itemClicked()



# List Widget

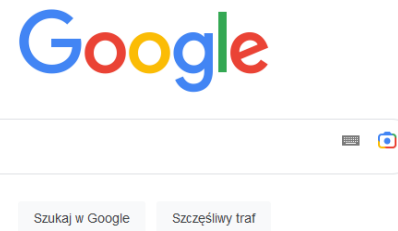
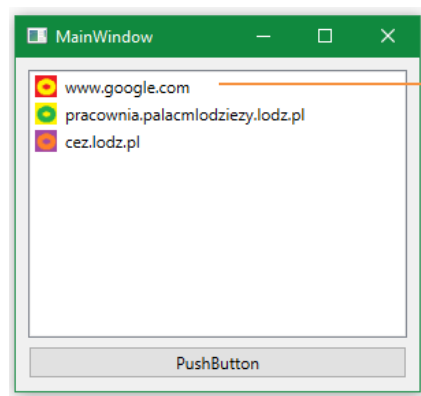
```
Projekty
└─ listWidget
   └─ CMakeLists.txt
      └─ listWidget
         └─ Header Files
            └─ mainwindow.h
               └─ Source Files
                  └─ main.cpp
                     └─ mainwindow.cpp
                        └─ mainwindow.ui
                           └─ resource.qrc
                              └─ CMake Modules

mainwindow.h
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5  #include <QListWidgetItem>
6
7  QT_BEGIN_NAMESPACE
8  namespace Ui { class MainWindow; }
9  QT_END_NAMESPACE
10
11 class MainWindow : public QMainWindow
12 {
13     Q_OBJECT
14
15 public:
16     MainWindow(QWidget *parent = nullptr);
17     ~MainWindow();
18
19 private slots:
20     void on_pushButton_clicked();
21
22     void on_listWidget_itemClicked(QListWidgetItem *item);
23
24 private:
25     Ui::MainWindow *ui;
26 };
27 #endif // MAINWINDOW_H
28
```

# List Widget

w pliku mainwindow.cpp  
dodajemy kod obsługujący  
kliknięcie na pozycję listy

```
35  
36 void MainWindow::on_listWidget_itemClicked(QListWidgetItem *item)  
37 {  
38     const QString link = "http://" + item->text();  
39     QDesktopServices::openUrl(QUrl(link));  
40 }
```



po kliknięciu na  
wybrany adresie  
otwiera się  
przeglądarka

# Ankieta

MainWindow

## Ankieta

imie

nazwisko

**Płeć**

mężczyzna  kobieta

**zainteresowania**

film

sport

programowanie

nauka

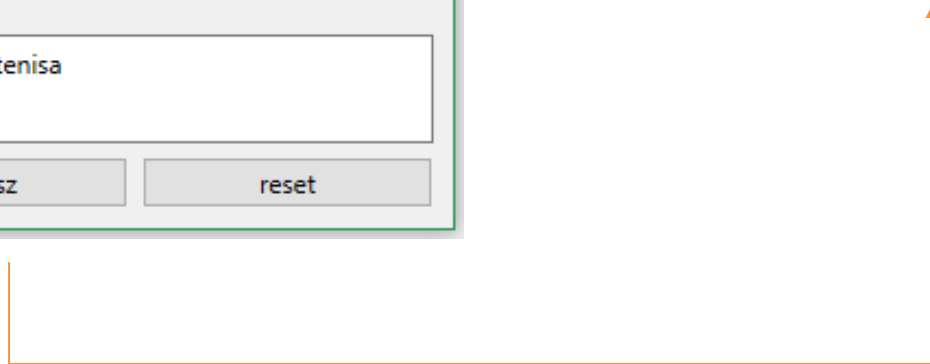
**szkoła**

**krótko o sobie**

ankieta.txt — Notatnik

Plik Edycja Format Widok Pomoc

```
imię: Jan
nazwisko: Kowalski
płeć: mężczyzna
zainteresowania: film programowanie nauka sport
szkoła: Technikum
o sobie:
lubię grać w tenisa
```



# Ankieta

The image shows a Qt Designer widget form titled "Ankieta". The form is enclosed in a red border and contains the following elements:

- Title:** "Ankieta"
- Form Fields:**
  - imię:** A text input field (QLineEdit).
  - nazwisko:** A text input field (QLineEdit).
  - Płeć:** Two radio buttons (QRadioButton) labeled "mężczyzna" and "kobieta".
  - zainteresowania:** Four checkboxes (QCheckBox) labeled "film", "sport", "programowanie", and "nauka".
  - szkoła:** A dropdown menu (QComboBox).
  - krótko o sobie:** A text area (QTextEdit).
- Buttons:** Two buttons (QPushButton) labeled "zapisz" and "reset" at the bottom.

Labels on the left side of the form point to the following QLabels:

- imię
- nazwisko
- Płeć
- zainteresowania
- szkoła
- krótko o sobie

Labels on the right side of the form point to the following Qt Widgets:

- QLineEdit (for the "imię" and "nazwisko" fields)
- QRadioButton (for the "mężczyzna" and "kobieta" radio buttons)
- QCheckBox (for the "film", "sport", "programowanie", and "nauka" checkboxes)
- QComboBox (for the "szkoła" dropdown menu)
- QTextEdit (for the "krótko o sobie" text area)
- QPushButton (for the "zapisz" and "reset" buttons)

# Ankieta

mainwindow.cpp

```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QFile>
4 #include <QFileDialog>
5
6 MainWindow::MainWindow(QWidget *parent)
7     : QMainWindow(parent)
8     , ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11     ui->comboBoxSchool->addItem("Podstawowa");
12     ui->comboBoxSchool->addItem("Liceum");
13     ui->comboBoxSchool->addItem("Technikum");
14     ui->comboBoxSchool->addItem("Politechnika");
15     ui->comboBoxSchool->addItem("Uniwersytet");
16 }
```

w konstruktorze klasy MainWindow  
inicjalizujemy comboBoxSchool  
poszczególnymi pozycjami

# Ankieta

mainwindow.cpp

```
24 void MainWindow::on_pushButtonSave_clicked()
25 {
26     // Dialog zapisu pliku
27     QString filename= QFileDialog::getSaveFileName(this, "Save As", "ankieta.txt");
28
29     QFile file(filename);
30
31     // Jeśli nie powiedzie się otwarcie pliku kończymy
32     if (!file.open(QIODevice::WriteOnly | QIODevice::Text))
33         return;
34
35     QString plec = "";
36
37     if (ui->radioButtonMan->isChecked()){
38         plec = "mężczyzna";
39     }else{
40         plec = "kobieta";
41     }
42
43     QString zainteresowania = "";
44
45     if (ui->checkBoxFilm->isChecked()){
46         zainteresowania += " film ";
47     }
48     if (ui->checkBoxProgramming->isChecked()){
49         zainteresowania += " programowanie ";
50     }
51     if (ui->checkBoxScience->isChecked()){
52         zainteresowania += " nauka ";
53     }
54     if (ui->checkBoxSport->isChecked()){
55         zainteresowania += " sport ";
56     }
57
58     QString szkola = ui->comboBoxSchool->currentText();
59
60     QString oSobie = ui->textEditAboutYou->toPlainText();
61
62     QTextStream out(&file);
63
64     out << "imię: " + ui->lineEditName->text() +
65         "\nnazwisko: " + ui->lineEditSurname->text() +
66         "\npleć: " + plec +
67         "\nzainteresowania: " + zainteresowania +
68         "\nszkoła: " + szkola +
69         "\no sobie: \n" + oSobie;
70
71     file.close();
72 }
73 }
```

pobieramy wpisy z  
ankiety i zapisujemy  
do pliku ankieta.txt

# Ankieta

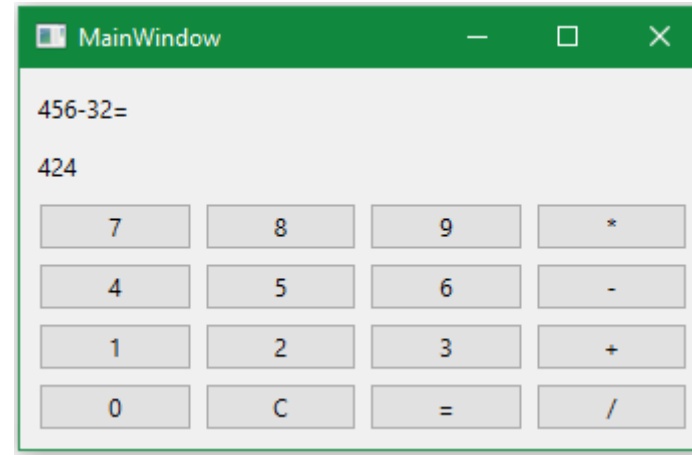
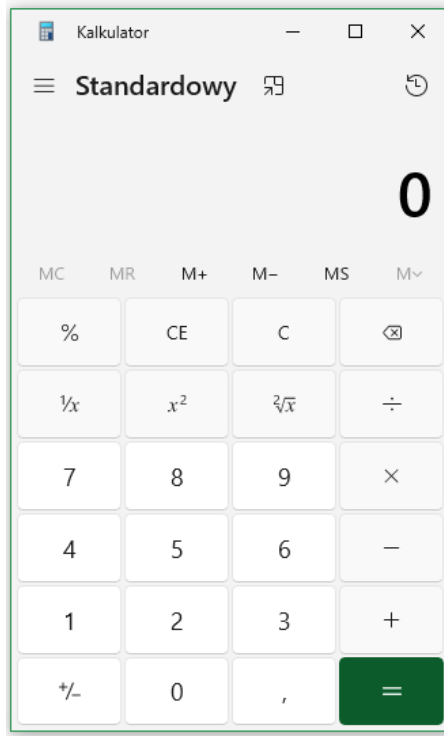
mainwindow.cpp

```
76 void MainWindow::on_pushButtonReset_clicked()  
77 {  
78     ui->lineEditName->setText("");  
79     ui->lineEditSurname->setText("");  
80  
81     ui->radioButtonMan->setAutoExclusive(false);  
82     ui->radioButtonMan->setChecked(false);  
83     ui->radioButtonMan->setAutoExclusive(true);  
84  
85     ui->radioButtonWoman->setAutoExclusive(false);  
86     ui->radioButtonWoman->setChecked(false);  
87     ui->radioButtonWoman->setAutoExclusive(true);  
88  
89     ui->checkBoxFilm->setChecked(false);  
90     ui->checkBoxProgramming->setChecked(false);  
91     ui->checkBoxScience->setChecked(false);  
92     ui->checkBoxSport->setChecked(false);  
93  
94     ui->comboBoxSchool->setCurrentIndex(0);  
95  
96     ui->textEditAboutYou->clear();  
97 }
```

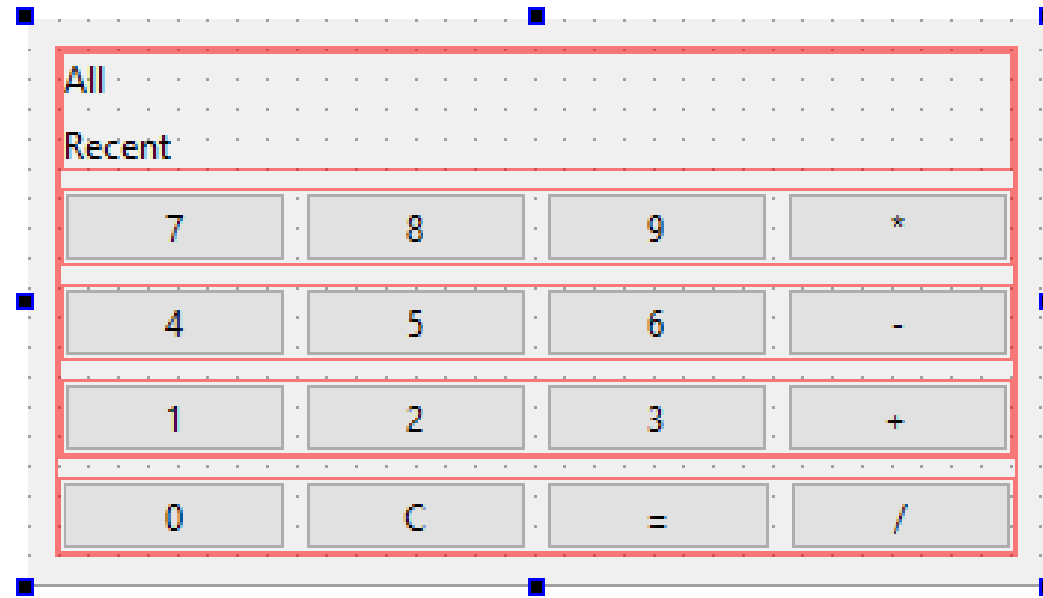
reset ankiety



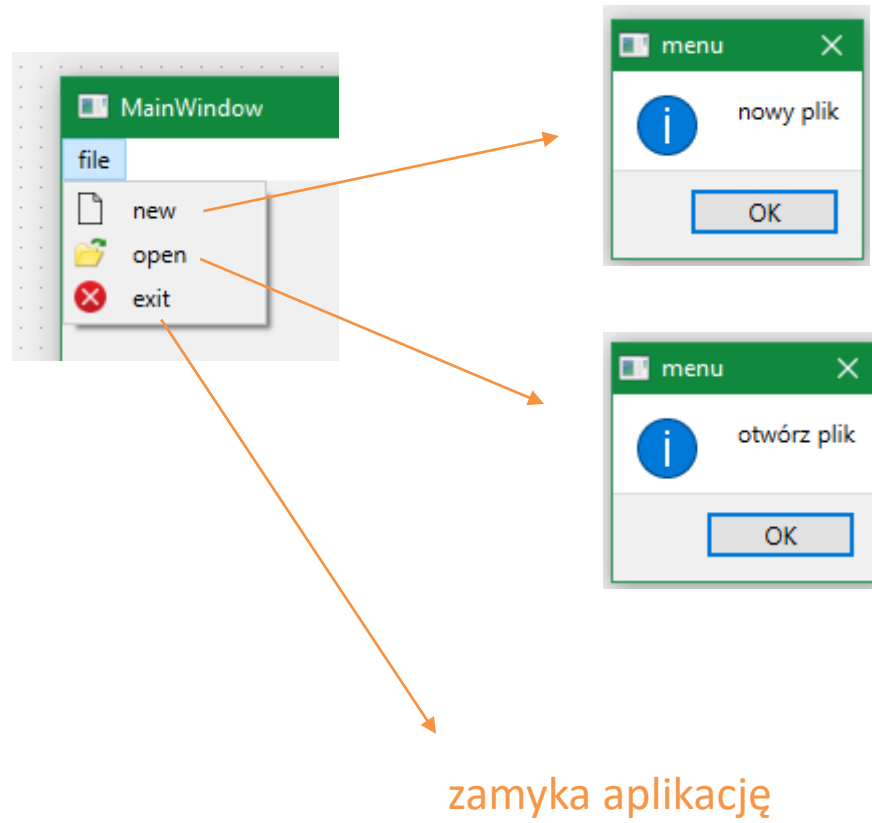
# Kalkulator



# Kalkulator

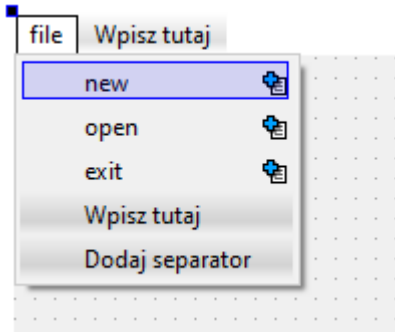


# Menu

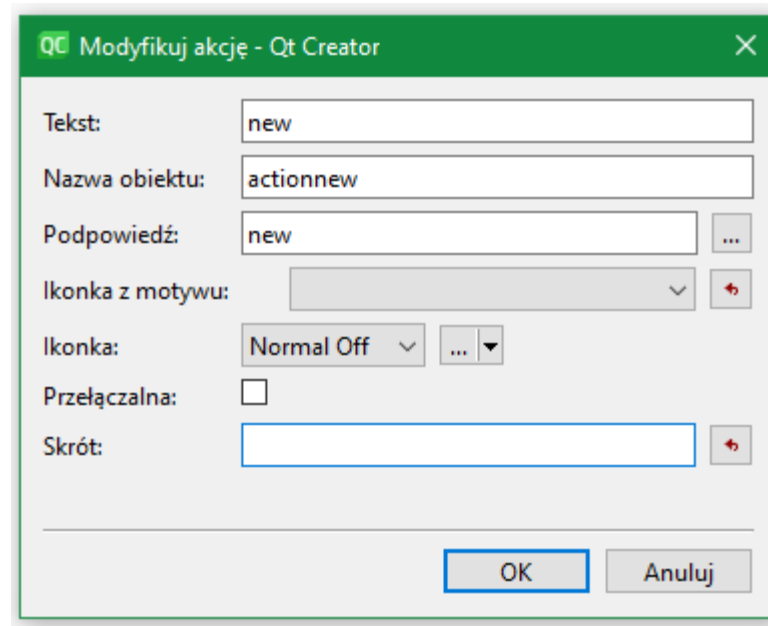


# Menu

Każdy nowy projekt ma już kontrolkę QMenuBar



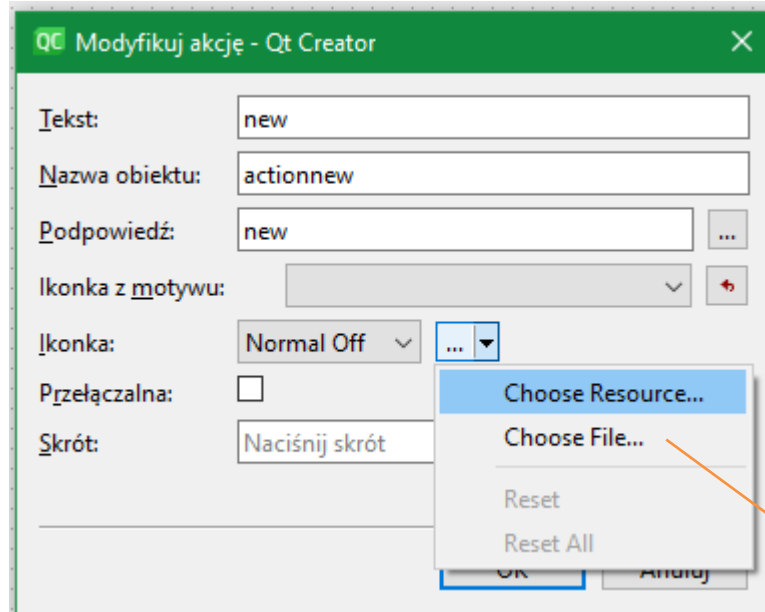
okienko modyfikacji akcji



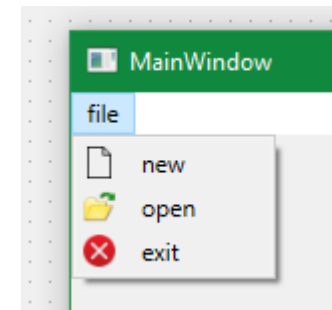
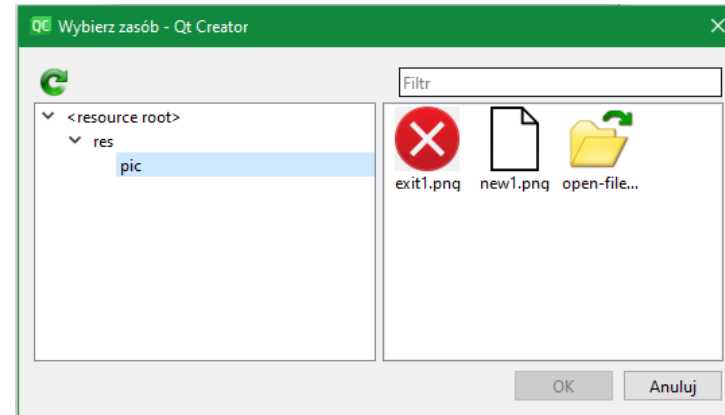
Nazwa	Użyta	Tekst	Skrót	Przełączalny	Podpowiedź
actionnew	<input checked="" type="checkbox"/>	new	db click	<input type="checkbox"/>	new
actionopen	<input checked="" type="checkbox"/>	open		<input type="checkbox"/>	open
actionexit	<input checked="" type="checkbox"/>	exit		<input type="checkbox"/>	exit

do każdej pozycji menu jest przypisana akcja

# Menu

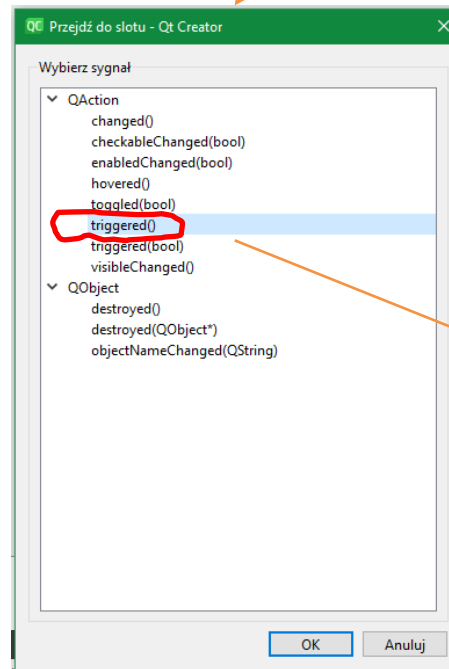
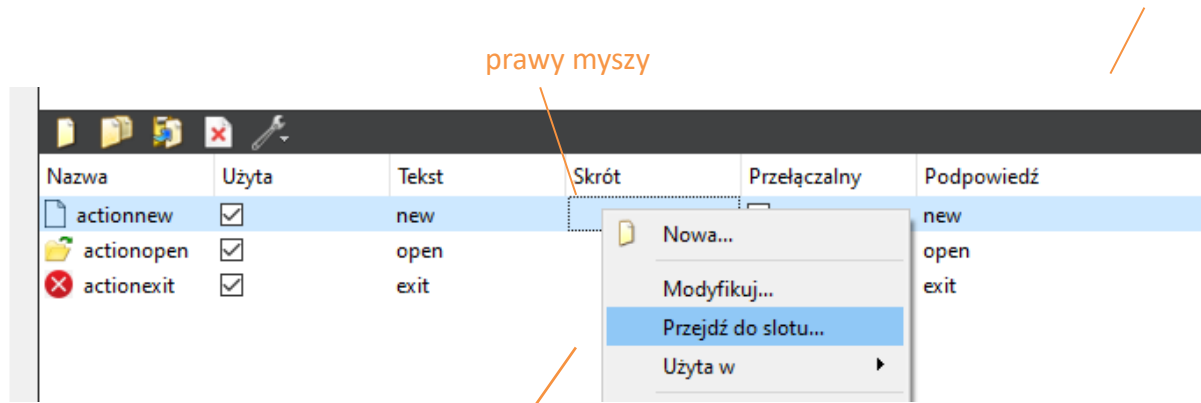


wybieranie ikonki z zasobów  
(które najpierw należy utworzyć)



# Menu

dodawanie akcji



```
#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_actionnew_triggered()
{
    QMessageBox::information(this, "menu", "nowy plik");
}

void MainWindow::on_actionopen_triggered()
{
    QMessageBox::information(this, "menu", "otwórz plik");
}

void MainWindow::on_actionexit_triggered()
{
    QApplication::quit();
}
```

# QDir

mainwindow.cpp

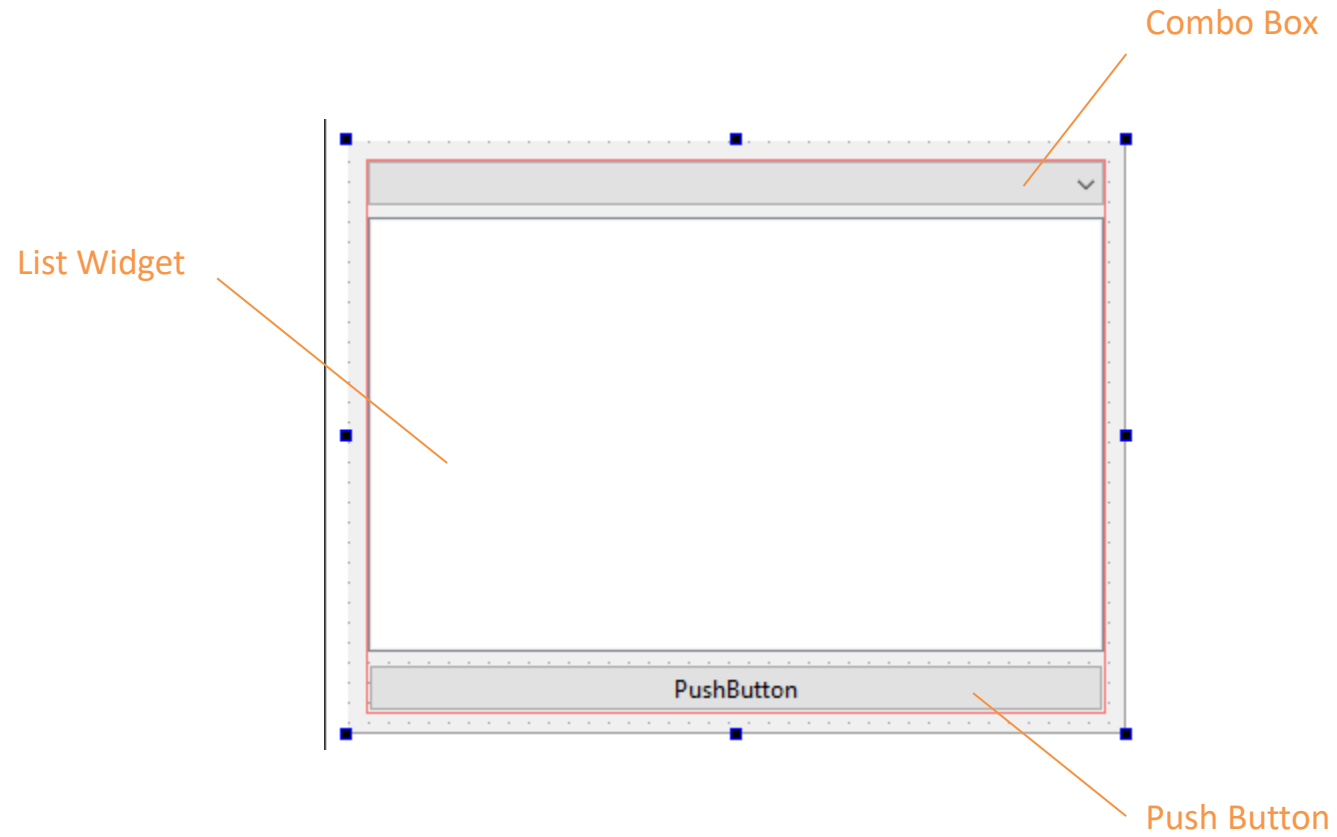
```
#include "mainwindow.h"  
#include "./ui_mainwindow.h"  
#include <QDir>  
#include <QMessageBox>
```

```
MainWindow::MainWindow(QWidget *parent)  
    : QMainWindow(parent)  
    , ui(new Ui::MainWindow)  
{  
    ui->setupUi(this);  
    QDir dir("C:/komp");  
    if (dir.exists()){  
        QMessageBox::information(this,"Check","folder istnieje");  
    }else{  
        QMessageBox::information(this,"Check","folder nie istnieje");  
    }  
}
```

```
MainWindow::~MainWindow()  
{  
    delete ui;  
}
```

sprawdzenie czy folder c:/komp istnieje

# QDir





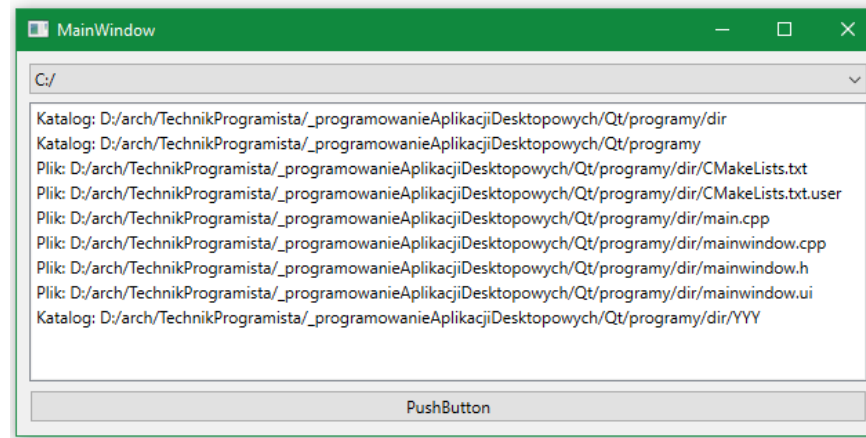
# QDir

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <QDir>
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    // wstawienie liter dysków do ComboBox np: c:/
    QDir dir;
    foreach (QFileInfo var, dir.drives()) {
        ui->comboBox->addItem(var.absoluteFilePath());
    }

    // wylistowanie zawartości katalogu projektu
    QDir dir_2("D:/arch/TechnikProgramista/_programowanieAplikacjiDesktopowych/Qt/programy/dir");
    foreach (QFileInfo var, dir_2.entryInfoList()) {
        if(var.isDir()){
            ui->listWidget->addItem("Katalog: " + var.absoluteFilePath());
        }
        if(var.isFile()){
            ui->listWidget->addItem("Plik: " + var.absoluteFilePath());
        }
    }
}
```



# QDir

po naciśnięciu przycisku w katalogu projektu  
tworzony jest folder YYY (jeśli nie istnieje)

```
void MainWindow::on_pushButton_clicked()
{
    // tworzymy nowy folder w katalogu projektu
    QDir dir_3("D:/arch/TechnikProgramista/_programowanieAplikacjiDesktopowych/Qt/programy/dir/YYY");
    if(dir_3.exists()){
        QMessageBox::information(this,"Check","folder istnieje");
    }else{
        // tworzy folder YYY
        dir_3.mkpath("D:/arch/TechnikProgramista/_programowanieAplikacjiDesktopowych/Qt/programy/dir/YYY");
    }
}
```

# QDir

```
#include "mainwindow.h"  
#include "./ui_mainwindow.h"  
#include <QDir>  
#include <QMessageBox>
```

```
MainWindow::MainWindow(QWidget *parent)
```

```
    : QMainWindow(parent)  
    , ui(new Ui::MainWindow)
```

```
{
```

```
    ui->setupUi(this);
```

```
    // wstawienie liter dysków do ComboBox np: c:/
```

```
    QDir dir;
```

```
    foreach (QFileInfo var, dir.drives()) {
```

```
        ui->comboBox->addItem(var.absoluteFilePath());
```

```
    }
```

```
    // wylistowanie zawartości pierwszego dysku np: c:/ (pierwszy na liście dysk ma index 0)
```

```
    comboBox_add_items(0);
```

```
}
```

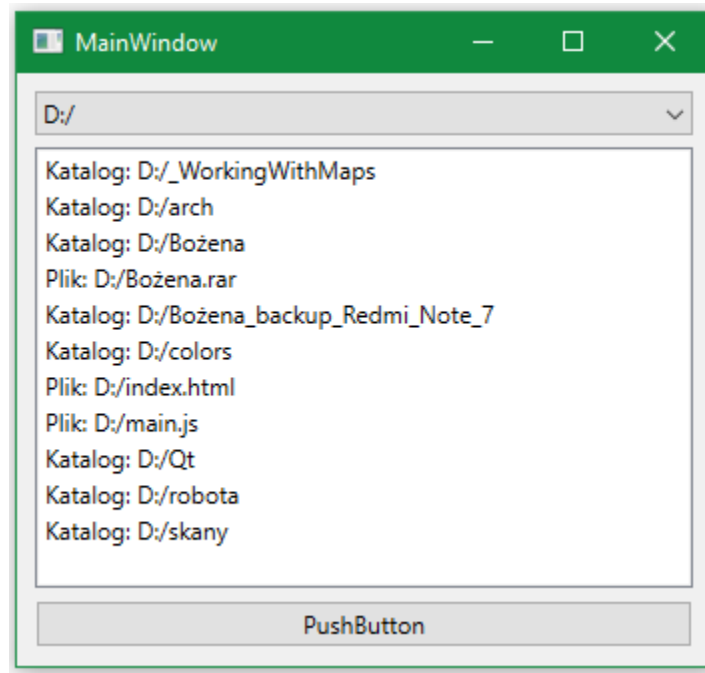
```
MainWindow::~MainWindow()
```

```
{
```

```
    delete ui;
```

```
}
```

po wybraniu litery dysku  
wypisują się pliki i katalogi  
zapisane na nim



metoda pomocnicza do wylistowania zawartości wybranego  
dysku (indeksy zaczynają się od 0)

# QDir

```
void MainWindow::on_pushButton_clicked()
{
    // tworzymy nowy folder w katalogu projektu
    QDir dir_3("D:/arch/TechnikProgramista/_programowanieAplikacjiDesktopowych/Qt/programy/dir/YYYY");
    if(dir_3.exists()){
        QMessageBox::information(this,"Check","folder istnieje");
    }else{
        // tworzy folder YYY
        dir_3.mkpath("D:/arch/TechnikProgramista/_programowanieAplikacjiDesktopowych/Qt/programy/dir/YYYY");
    }
}
```

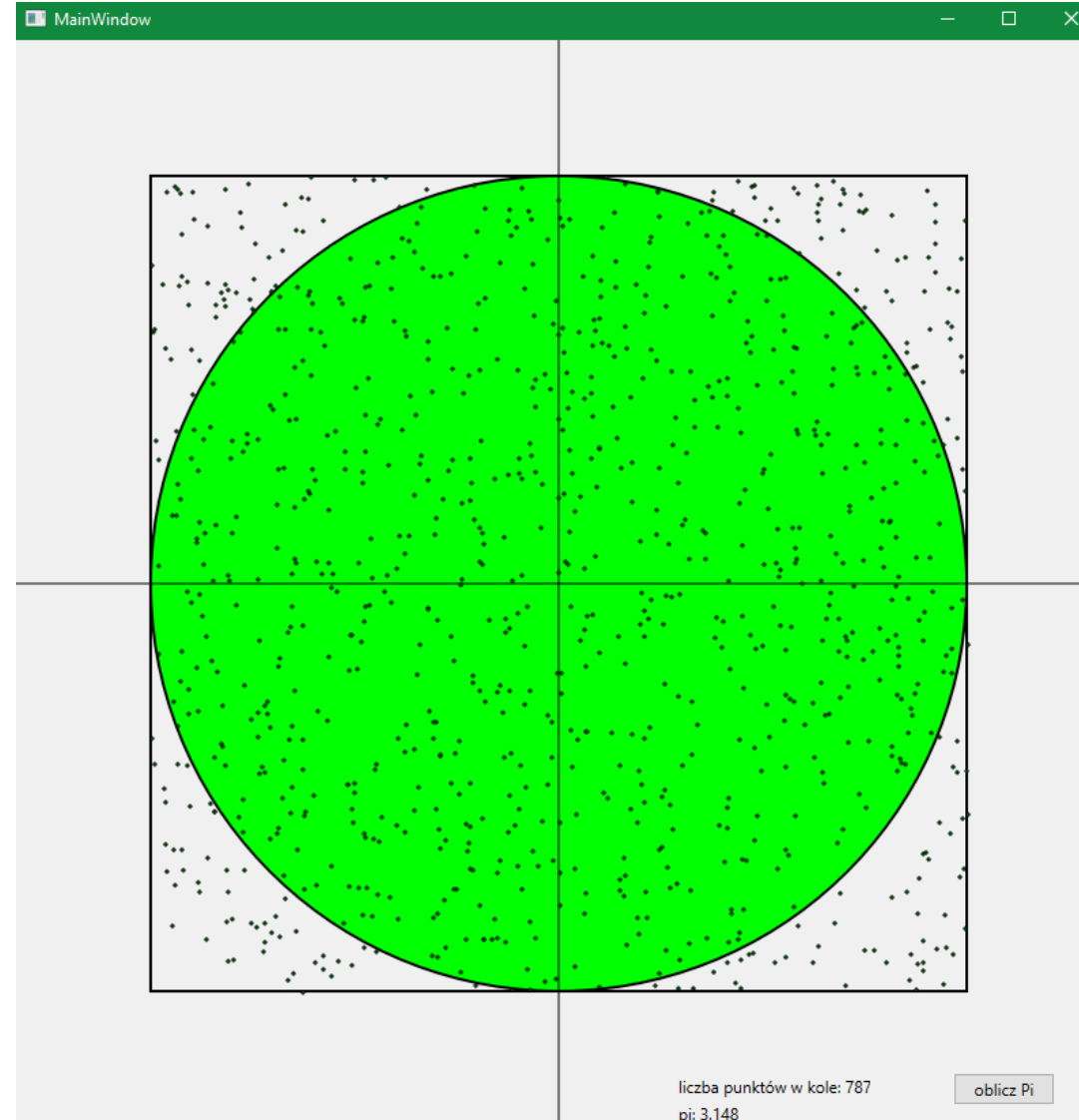
po wybraniu dysku w comboBoxie wywołujemy metodę pomocniczą do wylistowania jego zawartości.

```
// slot uruchamiany po wybraniu pozycji na comboBoxie
void MainWindow::on_comboBox_activated(int index)
{
    comboBox_add_items(index);
}
```

```
// metoda pomocnicza do wylistowania zawartości wybranego dysku (indeksy zaczynają się od 0)
void MainWindow::comboBox_add_items(int index){
    // czyszczenie zawartości listWidget (bez czyszczenia kolejne wybrane itemy dopisują się na końcu)
    ui->listWidget->clear();
    //QDir dir_2(ui->comboBox->currentText());
    QDir dir_2(ui->comboBox->itemText(index));
    foreach (QFileInfo var, dir_2.entryInfoList()) {
        if(var.isDir()){
            ui->listWidget->addItem("Katalog: " + var.absoluteFilePath());
        }
        if(var.isFile()){
            ui->listWidget->addItem("Plik: " + var.absoluteFilePath());
        }
    }
}
```

# QPainter

obliczenie liczby  $\pi$  metodą Monte Carlo



<http://programmingexamples.wikidot.com/qt-qpainter-example>

# Obsługa poczty

Nadaj Przesyłkę, PESEL: 00000000000


Rodzaj przesyłki

Pocztówka

List

Paczka

Sprawdź Cenę

 **Cena: 1,5 zł**

Dane adresowe

Ulica z numerem  
Konopnickiej 98

Kod pocztowy  
65124

Miasto

Zatwierdź

[poczta.zip](http://poczta.zip)

# Obsługa poczty

GroupBox

Rodzaj przesyłki

RadioButton

RadioButton

RadioButton

PushButton

TextLabel      TextLabel

Dane adresowe

TextLabel

TextLabel

TextLabel

PushButton

LineEdit

nazwy obiektów powinny być adekwatne do wykonywanej funkcji np, pushButton\_confirm

# Obsługa poczty

Nadaj przesyłkę, PESEL: 00000000000


Rodzaj przesyłki

Pocztaówka

List

Paczka

Sprawdź cenę

 **Cena: 1 zł**

Dane adresowe

Ulica z numerem

Kod pocztowy

Miasto

Zatwierdź



# Galeria z polubieniami

nagłówek zdjęcia

drugi kwiatek



zdjęcie

przycisk poprzednie zdjęcie

<

>

polub

polubienia

pierwszy kwiatek  
czwarty kwiatek  
drugi kwiatek

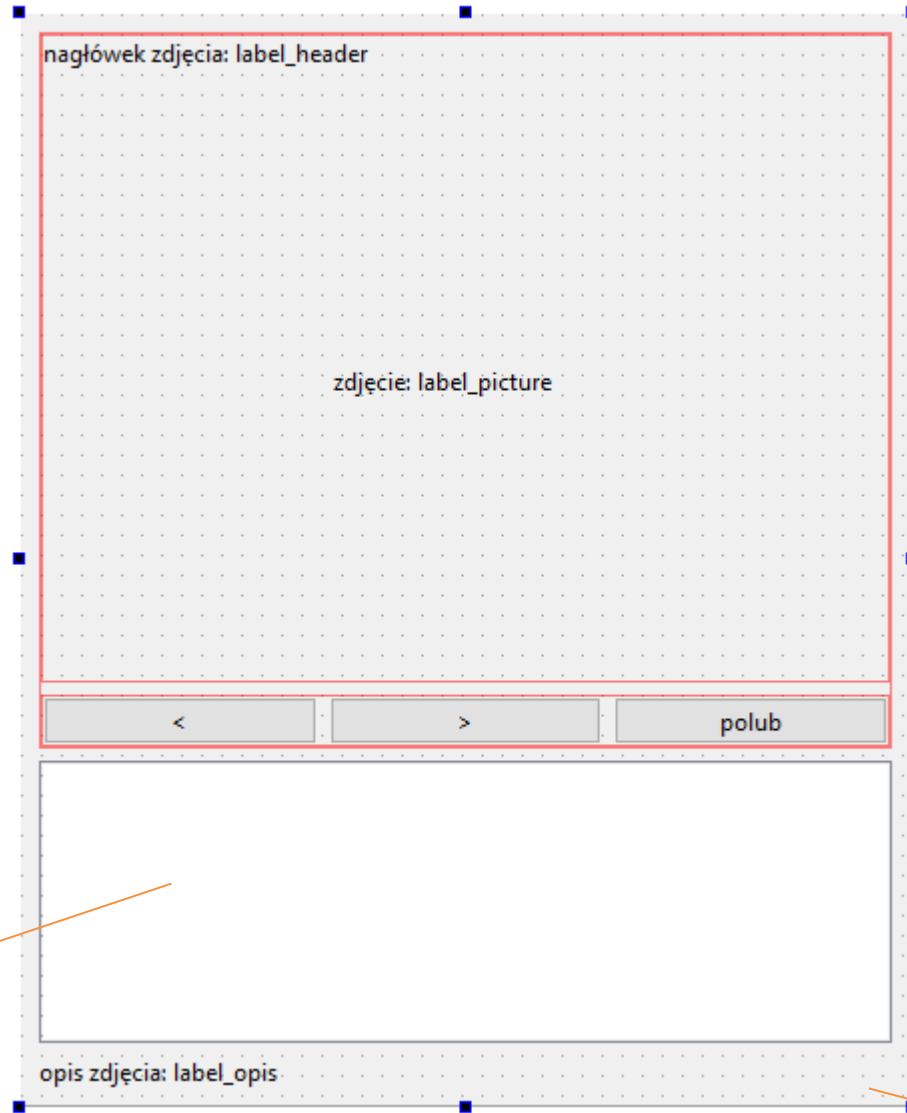
przycisk polubienia

przycisk następne zdjęcie

opis zdjęcia

kwiatek 2

# Galeria z polubieniami

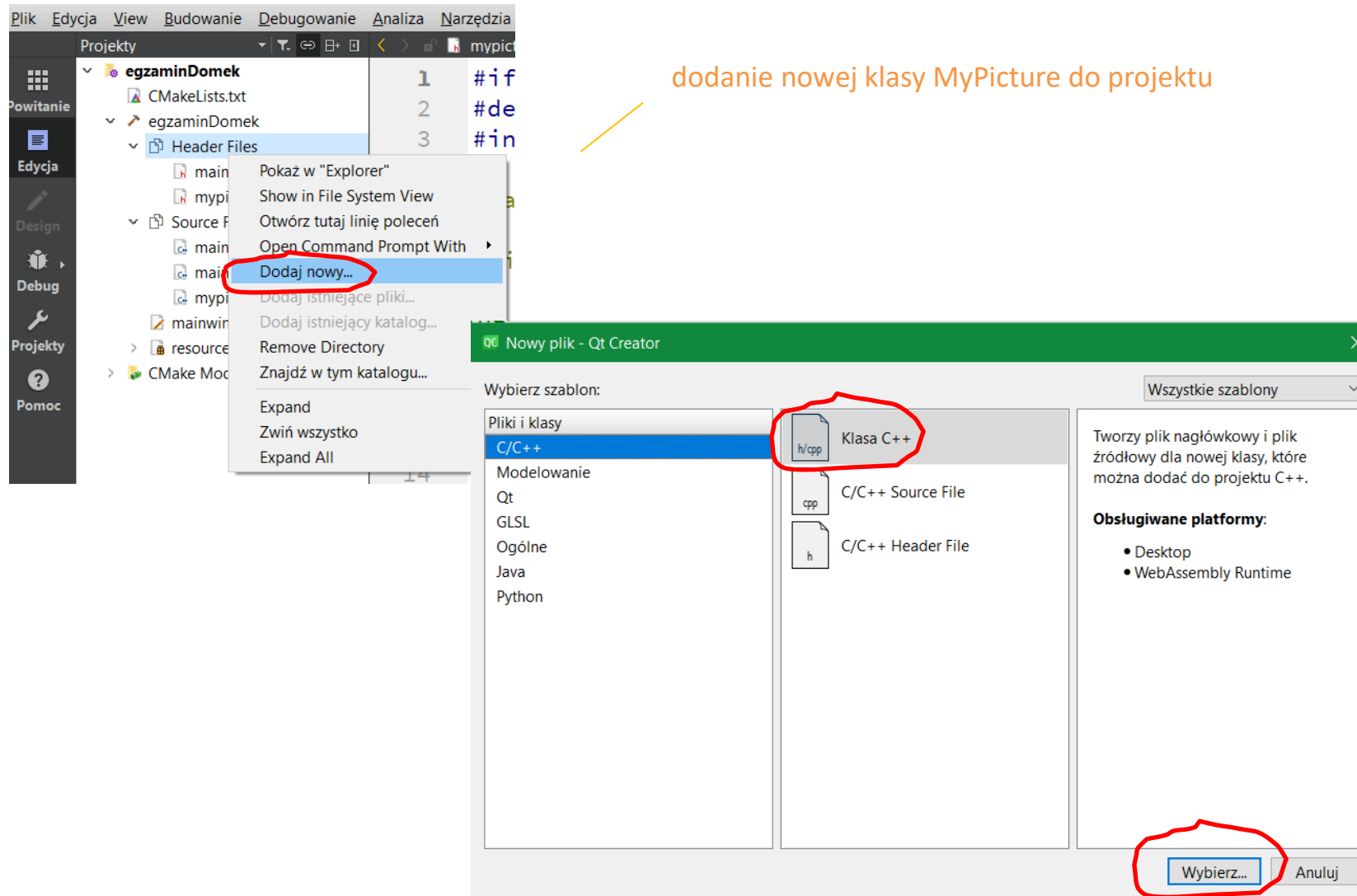


QListWidget

prawy/rozmiar w siatce

Obiekt	Klasa
MainWindow	QMainWindow
centralwidget	QWidget
label_opis	QLabel
listWidget_polubienia	QListWidget
verticalLayout_2	QVBoxLayout
horizontalLayout	QHBoxLayout
btn_lewo	QPushButton
btn_polub	QPushButton
btn_prawo	QPushButton
verticalLayout	QVBoxLayout
label_header	QLabel
label_picture	QLabel

# Galeria z polubieniami



# Galeria z polubieniami

Klasa C++

Szczegóły  
Podsumowanie

### Zdefiniuj klasę

Nazwa klasy:

Klasa bazowa:

Dołącz QObject  
 Dołącz QWidget  
 Dołącz QMainWindow  
 Dołącz QDeclarativeItem - Qt Quick 1  
 Dołącz QQuickItem - Qt Quick 2  
 Dołącz QSharedData  
 Add Q\_OBJECT

Plik nagłówkowy:

Plik źródłowy:

Ścieżka:

dobanie nowej klasy MyPicture

Klasa C++

Szczegóły  
Podsumowanie

### Organizacja projektu

Dodaj do projektu:

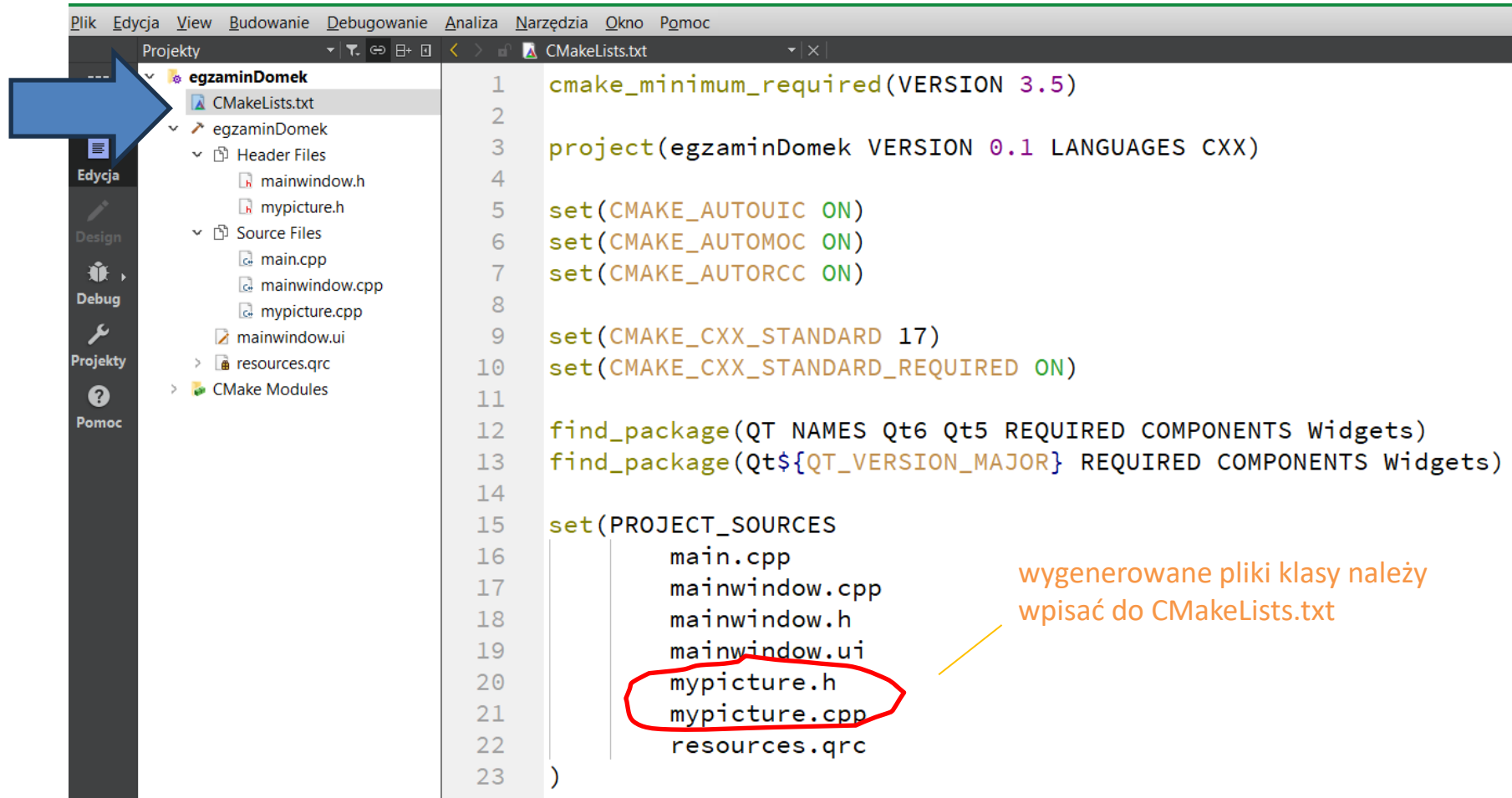
Add to version control:

Pliki, które zostaną dodane w

D:\arch\TechnikProgramista\\_programowanieAplikacjiDesktopowych\Qt

mypicture.cpp  
mypicture.h

# Galeria z polubieniami



Plik Edycja View Budowanie Debugowanie Analiza Narzędzia Okno Pomoc

Projekty

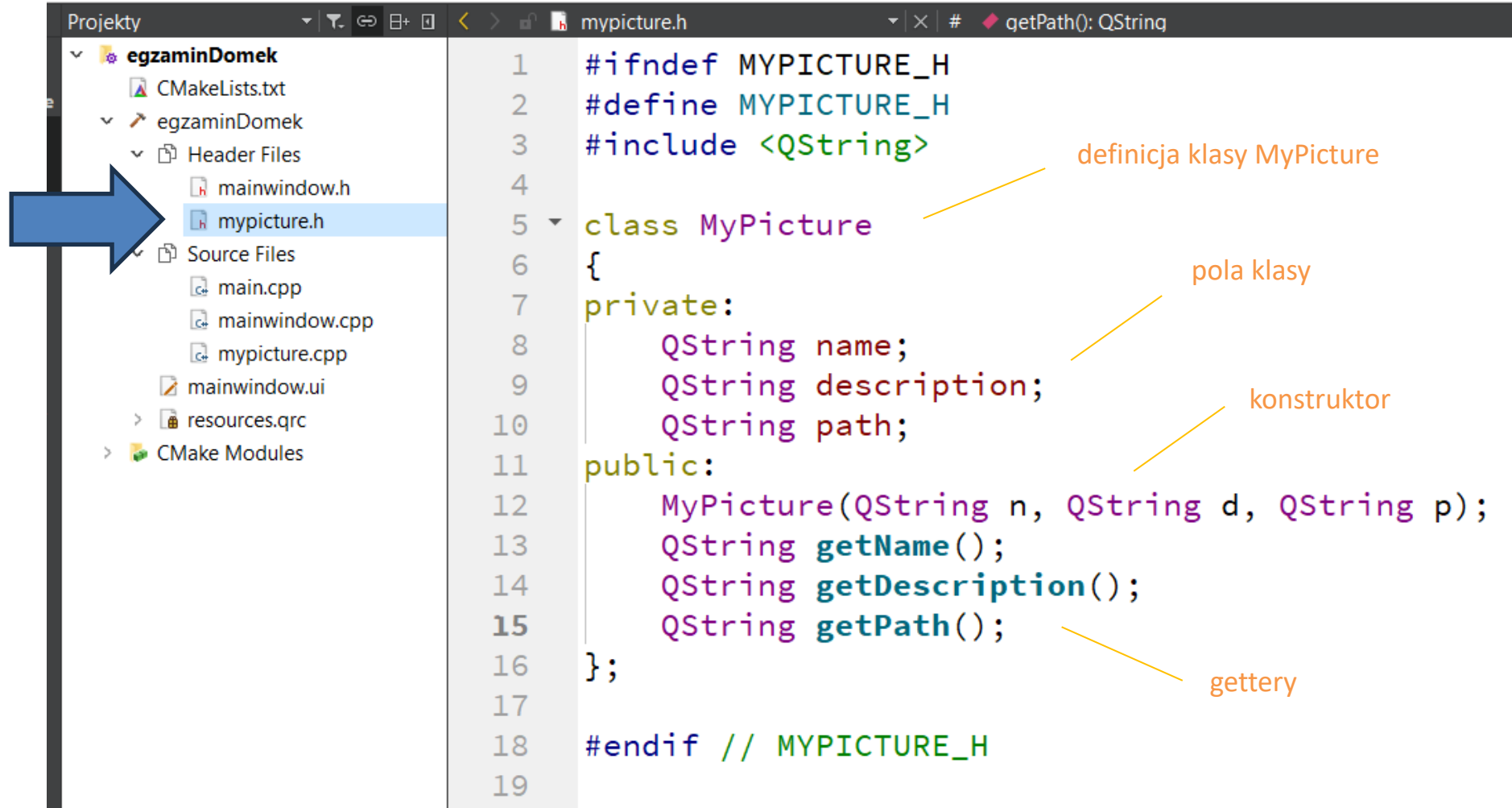
egzaminDomek

- CMakeLists.txt
- egzaminDomek
  - Header Files
    - mainwindow.h
    - mypicture.h
  - Source Files
    - main.cpp
    - mainwindow.cpp
    - mypicture.cpp
  - mainwindow.ui
  - resources.qrc
  - CMake Modules

```
1 cmake_minimum_required(VERSION 3.5)
2
3 project(egzaminDomek VERSION 0.1 LANGUAGES CXX)
4
5 set(CMAKE_AUTOUIC ON)
6 set(CMAKE_AUTOMOC ON)
7 set(CMAKE_AUTORCC ON)
8
9 set(CMAKE_CXX_STANDARD 17)
10 set(CMAKE_CXX_STANDARD_REQUIRED ON)
11
12 find_package(QT NAMES Qt6 Qt5 REQUIRED COMPONENTS Widgets)
13 find_package(Qt${QT_VERSION_MAJOR} REQUIRED COMPONENTS Widgets)
14
15 set(PROJECT_SOURCES
16     main.cpp
17     mainwindow.cpp
18     mainwindow.h
19     mainwindow.ui
20     mypicture.h
21     mypicture.cpp
22     resources.qrc
23 )
```

wygenerowane pliki klasy należy wpisać do CMakeLists.txt

# Galeria z polubieniami



The image shows a code editor window with a project explorer on the left and a code editor on the right. The project explorer shows a project named 'egzaminDomek' with a file 'mypicture.h' selected. A blue arrow points to 'mypicture.h' in the project explorer. The code editor shows the content of 'mypicture.h' with the following code:

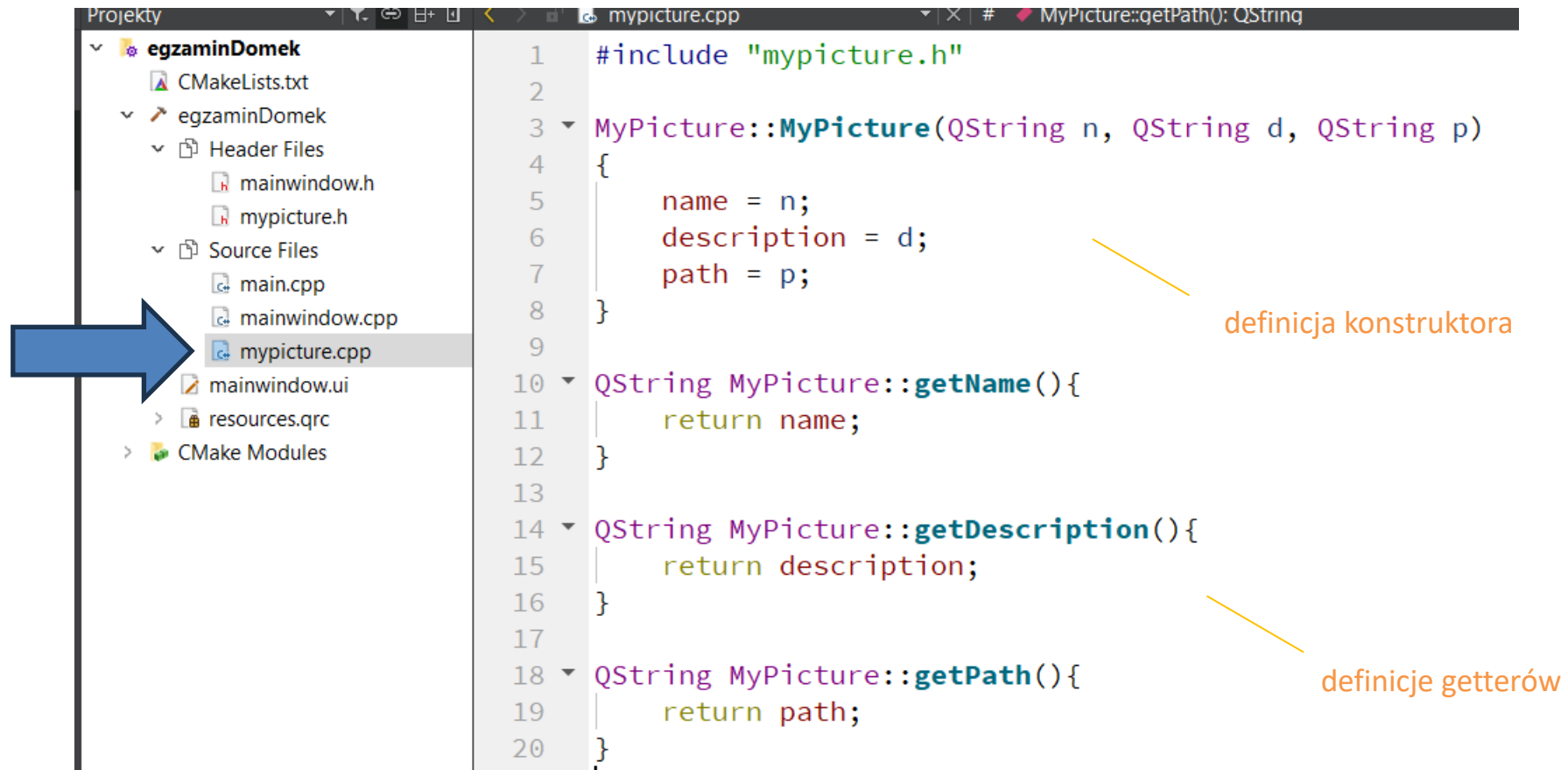
```
1 #ifndef MYPICTURE_H
2 #define MYPICTURE_H
3 #include <QString>
4
5 class MyPicture
6 {
7     private:
8         QString name;
9         QString description;
10        QString path;
11    public:
12        MyPicture(QString n, QString d, QString p);
13        QString getName();
14        QString getDescription();
15        QString getPath();
16 };
17
18 #endif // MYPICTURE_H
19
```

Annotations in orange text with yellow lines pointing to the code:

- definicja klasy MyPicture (points to line 5)
- pola klasy (points to lines 8-10)
- konstruktor (points to line 12)
- gettery (points to lines 13-15)

zawartość pliku nagłówkowego mypicture.h

# Galeria z polubieniami



```
Projekty
└─ egzaminDomek
   └─ CMakeLists.txt
      └─ egzaminDomek
         └─ Header Files
            ├── mainWindow.h
            └─ mypicture.h
            └─ Source Files
               ├── main.cpp
               ├── mainWindow.cpp
               └─ mypicture.cpp
            └─ mainWindow.ui
            └─ resources.qrc
            └─ CMake Modules
```

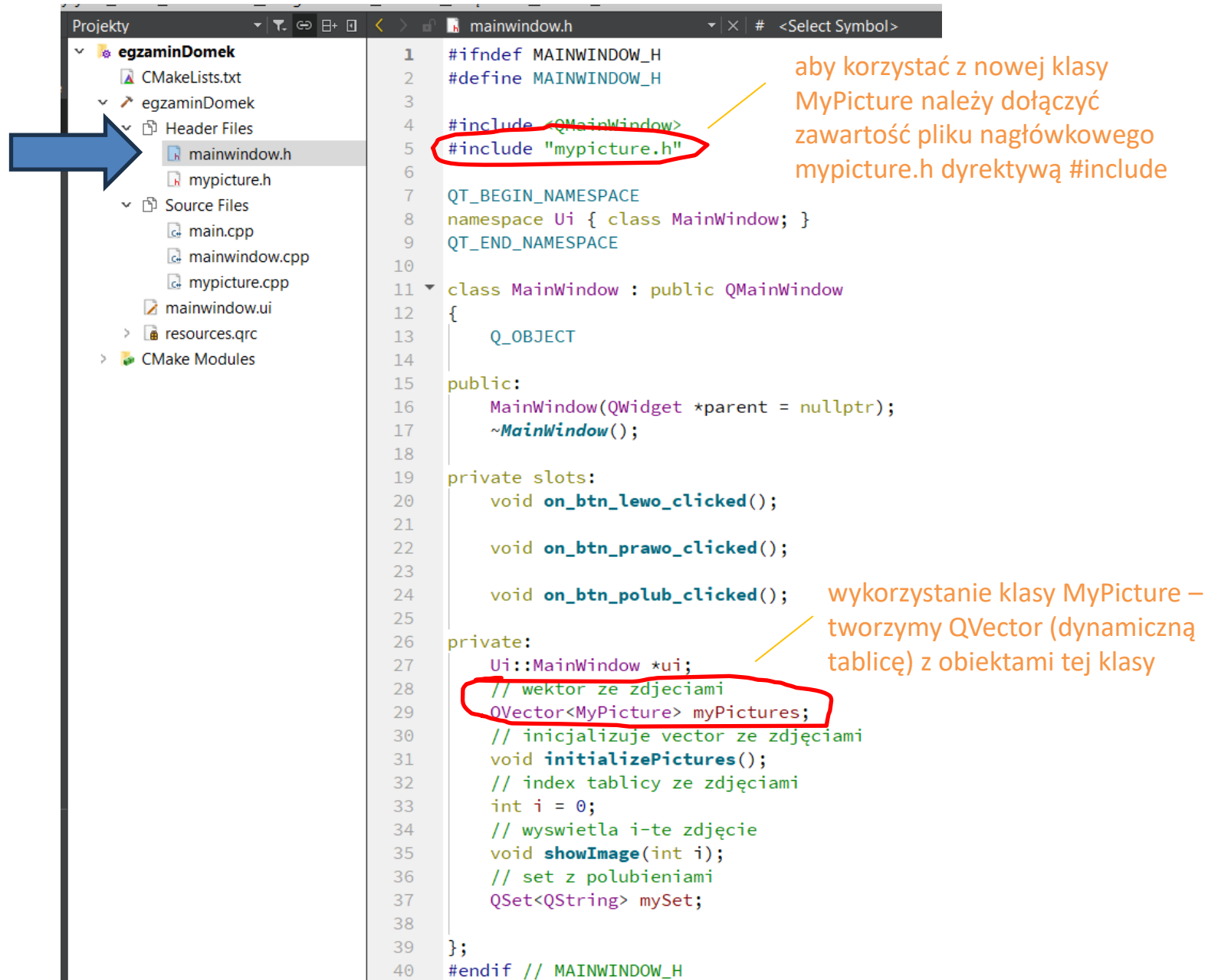
```
1  #include "mypicture.h"
2
3  MyPicture::MyPicture(QString n, QString d, QString p)
4  {
5      name = n;
6      description = d;
7      path = p;
8  }
9
10 QString MyPicture::getName(){
11     return name;
12 }
13
14 QString MyPicture::getDescription(){
15     return description;
16 }
17
18 QString MyPicture::getPath(){
19     return path;
20 }
```

definicja konstruktora

definicje getterów

zawartość pliku mypicture.cpp

# Galeria z polubieniami



Projekt: egzaminDomek

- egzaminDomek
  - Header Files
    - mainwindow.h**
    - mypicture.h
  - Source Files
    - main.cpp
    - mainwindow.cpp
    - mypicture.cpp
    - mainwindow.ui
  - resources.qrc
  - CMake Modules

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include "mypicture.h"
6
7 QT_BEGIN_NAMESPACE
8 namespace Ui { class MainWindow; }
9 QT_END_NAMESPACE
10
11 class MainWindow : public QMainWindow
12 {
13     Q_OBJECT
14
15 public:
16     MainWindow(QWidget *parent = nullptr);
17     ~MainWindow();
18
19 private slots:
20     void on_btn_lewo_clicked();
21
22     void on_btn_prawo_clicked();
23
24     void on_btn_polub_clicked();
25
26 private:
27     Ui::MainWindow *ui;
28     // wektor ze zdjęciami
29     QVector<MyPicture> myPictures;
30     // inicjalizuje vector ze zdjęciami
31     void initializePictures();
32     // index tablicy ze zdjęciami
33     int i = 0;
34     // wyświetla i-te zdjęcie
35     void showImage(int i);
36     // set z polubieniami
37     QSet<QString> mySet;
38
39 };
40 #endif // MAINWINDOW_H
```

aby korzystać z nowej klasy MyPicture należy dołączyć zawartość pliku nagłówkowego mypicture.h dyrektywą #include

wykorzystanie klasy MyPicture – tworzymy QVector (dynamiczną tablicę) z obiektami tej klasy



# Galeria z polubieniami

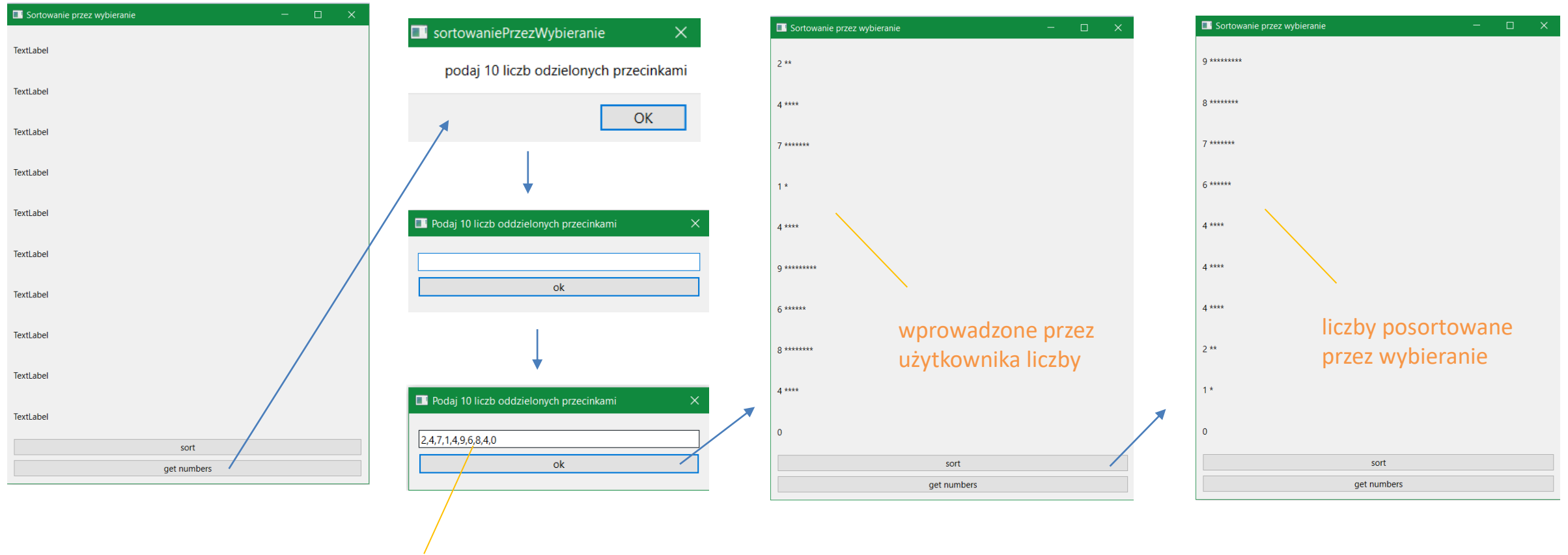


```
1 #include "mainwindow.h"
2 #include "./ui_mainwindow.h"
3 #include <QPixmap>
4 #include <QDebug>
5
6 MainWindow::MainWindow(QWidget *parent)
7     : QMainWindow(parent)
8     , ui(new Ui::MainWindow)
9 {
10     ui->setupUi(this);
11     initializePictures();
12     showImage(i);
13 }
14
15 MainWindow::~MainWindow()
16 {
17     delete ui;
18 }
19
20 void MainWindow::initializePictures(){
21     myPictures.push_back(MyPicture("pierwszy kwiatek", "kwiatek 1", ":/pic/pictures/01.jpg"));
22     myPictures.push_back(MyPicture("drugi kwiatek", "kwiatek 2", ":/pic/pictures/02.jpg"));
23     myPictures.push_back(MyPicture("trzeci kwiatek", "kwiatek 3", ":/pic/pictures/03.jpg"));
24     myPictures.push_back(MyPicture("czwarty kwiatek", "kwiatek 4", ":/pic/pictures/04.jpg"));
25     myPictures.push_back(MyPicture("piąty kwiatek", "kwiatek 5", ":/pic/pictures/05.jpg"));
26 }
27
```

tworzenie obiektów klasy MyPicture

dodanie obiektów klasy MyPicture do dynamicznej tablicy myPictures

# Sortowanie przez wybieranie



użytkownik musi wpisać właściwą ilość liczb oddzielonych przecinkami (inne znaki są ignorowane)