Interfejsy aplikacji w środowisku Windows

Qt – kontrolki, sygnały, sloty

Projekt z szablonu, sygnały i sloty

1. **Prezentacja prowadzącego**

2. **Przykład**

   Poniższy przykład pokazuje sposób budowy aplikacji *Qt Widgets* z wbudowanego szablonu.

   - Wygeneruj nowy projekt za pomocą opcji **File | New File or Project...**, w panelu **Projects** wybierz **Application (Qt)** i następnie **Qt Widgets Application**.
   - Podaj nazwę projektu, bez spacji i polskich znaków, sprawdź poprawność położenia projektu (dysk sieciowy **i:** i odpowiedni katalog).
   - Zostaw domyślną konfigurację projektu, w kolejnych ekranach kreatora projektu powinno być m. in. ustawione: **Build system: qmake**, **Class name: MainWindow**, **Base class: QMainWindow**, **Kits: Desktop Qt 5.*xx.xx* MinGW {32,64}-bit**.
   - W panelu **Projects** (po lewej stronie okna) rozwiń drzewo projektu, kliknij prawym przyciskiem myszy na nazwie projektu i wybierz opcję **Expand All**.
   - Sprawdź i przeanalizuj zawartość plików nagłówkowych (**`*.h`**) z sekcji **Headers** i źródłowych (**`*.cpp`**) z sekcji **Sources**.
   - Kliknij dwa razy na plik **`mainwindow.ui`** z sekcji **Forms** drzewa projektu. *QtCreator* przejdzie do trybu graficznego projektowania okna.
   - Przeciągnij kontrolkę (widżet) **Push Button** z palety kontrolek do okna aplikacji, zmień rozmiar przycisku.
   - Zmień napis na przycisku z domyślnego na **Zamknij**.
   - Przełącz się w tryb edycji sygnałów i slotów, wciśnij **Edit Signals/Slots** (z górnego paska narzędzi).
   - Kliknij w przycisk i przeciągnij kursor myszki i upuść nad oknem (powinna pojawić się czerwona linia powiązania pomiędzy przyciskiem a obszarem okna.
   - W oknie **Configure Connection** zaznacz opcję **Show signals and slots inherited from QWidget** i następnie w lewej liście wybierz sygnał **`clicked()`** a w prawej slot **`close()`**.
   - Uruchom aplikację i sprawdź działanie przycisku.
   - Przejdź do trybu edycji i sprawdź kod pliku **`mainwindow.ui`**. Poszukaj fragmentu odpowiedzialnego za obsługę połączenia.

```
<connections>
  <connection>
   <sender>pushButton</sender>
   <signal>clicked()</signal>
   <receiver>MainWindow</receiver>
   <slot>close()</slot>
   ...
```

   - Umieszczone są tam informacje o nadajniku (**`sender`**) i odbiorniku (**`receiver`**) sygnału oraz nazwy sygnału i slotu (rodzaje metod).
   - W katalogu **`build-`*`projekt`*`-Desktop_Qt_5.`*`xx.xx`*`_MinGW_{32,64}_bit-Debug`** poszukaj pliku o nazwie **`ui_mainwindow.h`** i przeanalizuj jego zawartość.
   - Dodaj do okna aplikacji kontrolki: **Horizontal Slider** i **Progress Bar**.
   - Utwórz połączenie sygnał-slot (w trybie **Edit Signals/Slots**) pomiędzy kontrolkami **Horizontal Slider** i **Progress Bar** (kliknij w pierwszą kontrolkę i przeciągnij połączenie do drugiej).

- W oknie **Configure Connection** wybierz sygnał **`valueChanged(int)`** (lewa lista wyboru) i slot **`setValue(int)`** (prawa lista wyboru).
- Uruchom aplikację i sprawdź jej działanie.
- Usuń powiązanie pomiędzy suwakiem i paskiem postępu, w trybie **Edit Signals/Slots** kliknij prawym przyciskiem myszy w linię powiązania i z menu podręcznego wybierz **Delete**.
- Utwórz ręcznie ponowne połączenie dopisując do konstruktora klasy kod wykorzystujący makro **`connect()`**.

```
MainWindow::MainWindow(QWidget *parent)
  : QMainWindow(parent)
  , ui(new Ui::MainWindow)
{
  ui->setupUi(this);
  connect(ui->horizontalSlider, SIGNAL(valueChanged(int)), ui->progressBar,
        SLOT(setValue(int)));
}
```

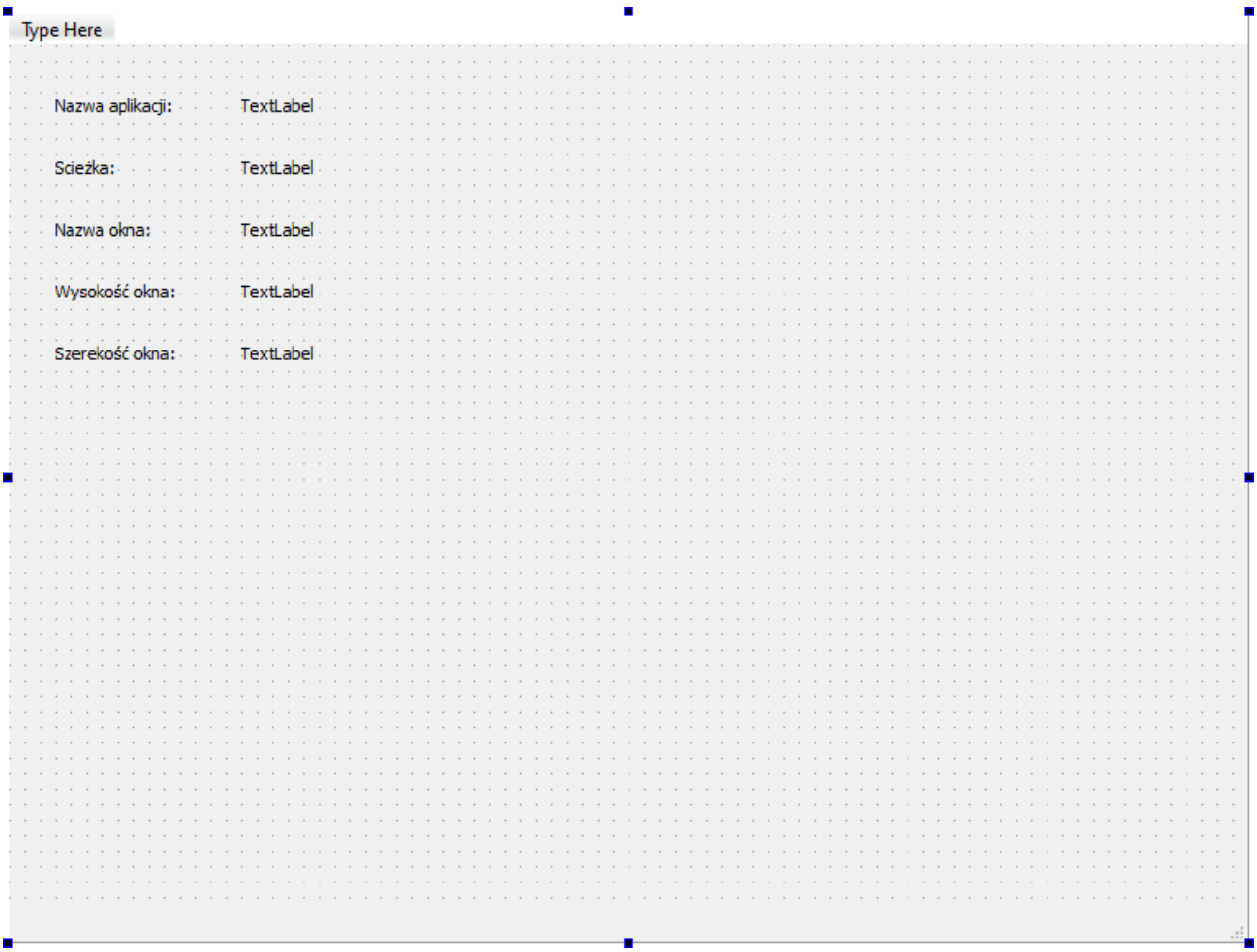
- Uruchom aplikację i sprawdź jej działanie.

Zasoby, komunikaty, sygnały i sloty

3. Przykład

Przykład aplikacji wykorzystującej kontrolki **`QLabel`** i komunikaty **`QMessegeBox`**.

- Wygeneruj nowy projekt **Qt Widgets Application**.



- Rozmieść w projektowanym oknie 10 kontrolek **QLabel**, ustal ich wielkość, położenie i treści napisów.
- Dodaj plik nagłówkowy **<QFileInfo>** i zupełnij konstruktor obiektu okna o kod (pamiętaj o sprawdzeniu nazw kontrolek, mogą się różnić od przykładowych).

```
QString sciezka = QCoreApplication::applicationFilePath();
QFileInfo plik(sciezka);

ui->label_6->setText(plik.fileName());
ui->label_7->setText(plik.absolutePath());

ui->label_8->setText(this->windowTitle());
ui->label_9->setText(QString::number(this->height()));
ui->label_10->setText(QString::number(this->width()));
```

- Uruchom aplikację i sprawdź jej działanie.
- Uzupełnij okno aplikacji o przycisk **Push Button**, zmień napis na **Wyświetl komunikat**.
- Zaznacz przycisk, kliknij prawym przyciskiem myszy i wybierz z menu podręcznego opcję **Go to slot...**
- Wybierz w oknie sygnał **clicked()** i zaakceptuj, w pliku **mainwindow.cpp** powinna zostać wygenerowana metoda o nazwie **on_pushButton_clicked()**.
- Dodaj plik nagłówkowy **<QMessageBox>** i uzupełnij metodę **on_pushButton_clicked()** o podane poniżej przykłady (pojedynczo).

```
QMessageBox::about(this, "Komunikat", "Treść komunikatu");
```

```
QMessageBox::about(this, "Komunikat", "Treść komunikatu\nw wielu\nliniach");
```

```
QMessageBox::critical(this, "Komunikat", "Treść komunikatu");
```

```
QMessageBox::information(this, "Komunikat", "Treść komunikatu");
```

```
QMessageBox::warning(this, "Komunikat", "Treść komunikatu");
```

```
QMessageBox::question(this, "Komunikat", "Treść komunikatu");
```

```
QMessageBox::question(this, "Komunikat", "Treść komunikatu",
        QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);
```

```
QMessageBox::question(this, "Komunikat", "Treść komunikatu",
        QMessageBox::Ok | QMessageBox::Cancel, QMessageBox::Cancel);
```

- Dodaj do aplikacji etykietę **Label** i plik nagłówkowy **<QDebug>**.
- Zmień metodę **on_pushButton_clicked()** na podaną poniżej (sprawdź nazwęswojej etykiety, może się różnić od przykładowej).

```
QMessageBox::StandardButton odpowiedz = QMessageBox::question(this, "Komunikat",
        "Treść komunikatu", QMessageBox::Yes | QMessageBox::Ok | QMessageBox::No,
        QMessageBox::Yes);

if(odpowiedz == QMessageBox::Yes)
{
  ui->label_11->setText("Wcisnąłeś Yes");
  qDebug() << "Yes";
}
else
```

```
{
  if(odpowiedz == QMessageBox::Ok)
  {
    ui->label_11->setText("Wcisnąłeś Ok");
    qDebug() << "Ok";
  }
  else
  {
    ui->label_11->setText("Wcisnąłeś No");
    qDebug() << "No";
  }
}
```

- Zwróć uwagę na zawartość panelu **Application Output** (na dole okna *Qt Creatora*).

4. Przykład

Pliki zasobów w Qt.

- Wygeneruj nowy projekt **Qt Widgets Application**.
- Pobierz pliki **`car.png`**, **`left.png`** i **`right.png`** ze strony przedmiotu, zapisz je w katalogu projektu.
- Do utworzonego projektu dodaj plik zasobów, kliknij prawym przyciskiem myszy na nazwie projektu w panelu **Projects** i wybierz opcję **Add New**...
- W otwartym oknie, w liście wyboru **Files and Classes** zaznacz **Qt** i następnie **Qt Resource File** (w środkowym panelu).
- Nadaj nazwę nowemu plikowi, np. **`resources`**.
- W drzewie projektu powinien pojawić się plik o nazwie **`resources.qrc`**. a *Qt Creator* powinien otworzyć nowy panel umożliwiający dodawanie zasobów.
- Kliknij przycisk **Add Prefix** i ustal nazwę, np. **`ikony`** i dodaj wszystkie pliki graficzne (przycisk **Add Files**).
- Do okna aplikacji dodaj kontrolkę **Label**, w panelu **Property** (prawa dolna część ekranu) odnajdź właściwość **`pixmap`**, rozwiń menu (kliknij w mały trójkąt) i wybierz opcję **Choose Resource...**
- W nowo otwartym oknie w lewym panelu zaznacz **`ikony`** a w prawym obraz samochodu, dostosuj wielkość kontroli do wielkości obrazka.
- Do okna aplikacji dodaj dwa przyciski **Push Button** i ustaw ich właściwości **`icon`** na odpowiednie obrazki z zasobów, zmień wielkość ikon (np. 64x64 pikseli, właściwość **`iconSize`**), wykasuj napisy (właściwość **`text`**).
- Wygeneruj odpowiednie sloty (**PPM** na kontrolce i opcja **Go to slot...**) dla sygnałów **`clicked()`** i uzupełnij metody.
- Strzałka w prawo.

```
ui->label->setGeometry(ui->label->x() - 10, ui->label->y(), ui->label->width(),
        ui->label->height());
```

- Strzałka w lewo.

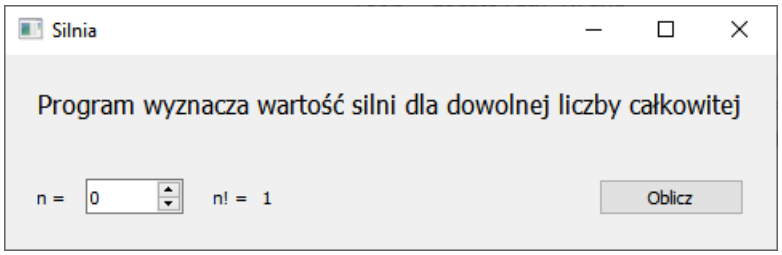
```
ui->label->setGeometry(ui->label->x() + 10, ui->label->y(), ui->label->width(),
        ui->label->height());
```

- Sprawdź i przeanalizuj działanie programu.

5. Zadanie

Opracuj program do wyznaczania silni.

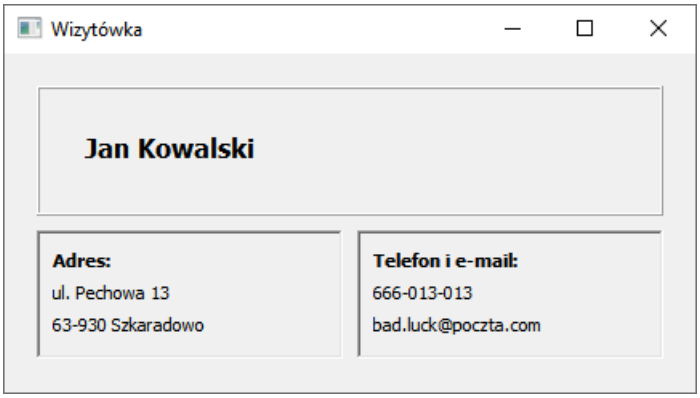
- Postać wyświetlanego okienka.



- Do budowy okna aplikacji użyj kontrolek:
  - etykieta **`QLabel`**,
  - pole edycyjne **`QSpinBox`**,
  - przycisk **`QPushButton`**.

6. Zadanie

Opracuj aplikację wyświetlającą wizytówkę według zamieszczonego poniżej wzorca.



- Do budowy okna aplikacji użyj kontrolek:
  - etykieta **`QLabel`**,
  - ramka **`QFrame`**.
- Dodaj przyciski **`QPushButton`** służące do zmiany wyświetlanych informacji na prywatne lub służbowe.