Diagram klas

# 1. Szkic diagramu (UML) Obraz zawierający tekst, zrzut ekranu, wizytówka, Czcionka Opis wygenerowany automatycznie

# Opis klas i odpowiedzialności

### 2.1. **MainWindow** (interfejs użytkownika)

**Odpowiedzialność:** Główne okno aplikacji, zarządzanie interakcją użytkownika

**Główne zadania**:

* Inicjalizuje obiekty logiki (np. DataClient) i interfejsu (np. PlotWidget).
* Odbiera akcje użytkownika (np. kliknięcie przycisku „Połącz z serwerem”) i wywołuje odpowiednie metody logiki.
* Umożliwia konfigurację dodatkowych wtyczek / filtrów i ich zastosowanie do danych.

**Kluczowe pola/metody**:

* dataClient : DataClient
* plotWidget : PlotWidget
* onConnectButtonClicked()
* onApplyFilterButtonClicked()

### **DataServer** (warstwa serwera)

**Odpowiedzialność**: Generowanie danych pomiarowych w pętli oraz udostępnianie ich przez sieć.

**Główne zadania**:

* Tworzy gniazdo i słucha na ustalonym porcie.
* Okresowo generuje losowe dane pomiarowe.
* Wysyła dane do podłączonych klientów.

**Kluczowe pola/metody**:

* port : int - port nasłuchu.
* startServer() - uruchamia serwer i rozpoczyna wysyłanie danych.
* stopServer() - zatrzymuje serwer.

### 2.3 **DataClient** (warstwa klienta)

**Odpowiedzialność**: Nawiązywanie połączenia z serwerem, odbieranie danych oraz ich buforowanie.

**Główne zadania**:

* Łączy się z serwerem po TCP.
* Cyklicznie odbiera dane pomiarowe.
* Przechowuje dane w wewnętrznym buforze.
* Zapewnia metodę dostępu do odebranych danych innym modułom.

**Kluczowe pola/metody**:

* serverAddress : std::string
* port : int
* dataBuffer : std::vector<double>
* connectToServer()
* receiveData()
* getDataBuffer() : std::vector<double>

### 2.4 **DataProcessor** (interfejs lub klasa bazowa – logika przetwarzania)

**Odpowiedzialność**: Definiowanie wspólnego kontraktu dla wszelkich filtrów / metod przetwarzania danych.

**Główne zadania**:

* Deklaruje metodę processData(), która przyjmuje surowe dane i zwraca dane przetworzone.
* Może zostać rozszerzona o różne filtry

**Kluczowe pola/metody**:

* virtual std::vector<double> processData(const std::vector<double>& inputData) = 0;

**Przykład konkretnej klasy dziedziczącej:** FilterSmooth

* Wykonuje wygładzanie danych w oknie czasowym.
* Metoda processData() implementuje algorytm wygładzenia.

### 2.5 **PlotWidget** (wizualizacja danych)

**Odpowiedzialność**: Prezentacja otrzymanych danych na wykresie w czasie rzeczywistym.

**Główne zadania**:

* Rysuje wykres na podstawie dostarczonych próbek.
* Umożliwia konfigurację wyglądu.
* Oferuje metody do odświeżania wykresu, gdy nadejdą nowe dane.

**Kluczowe pola/metody**:

* updatePlot(const std::vector<double>& data)
* setPlotParameters(const PlotParams& params) (gdzie PlotParams może zawierać np. zakres osi, kolory itp.)