Sprawozdanie projekt 1

Jakub Kłopotek Główczewski	186067	ACIR
Tymoteusz Byrwa	184414	ACIR

Założeniem naszego projektu było stworzenie kalkulatora posiadającego podstawowe funkcje wraz z zegarem analogowym/cyfrowym. Oba elementy są możliwe do zmiany pod względem wizualnym.

Zadanie zostało wykonane w pythonie przy użyciu biblioteki PySide6. Projekt napisaliśmy obiektowo opierając się na klasach. Grafika została zaimplementowana w postaci zegaru analogowego oraz możliwości zmiany w wyglądzie aplikacji. Layout całej aplikacji został stworzony w QT Designerze.

Zaczynając od Klasy “Calculator”. Logika opiera sie na bardzo prostej koncepcji. Stosujemy stos na który wpychamy liczby po kolei tak jak zostały wciśnięte:

```
def onButtonClicked(self,number):
    self.stack.append(number)
    text = "".join(str(n) for n in self.stack)
    self.updatetext(text)
```

Następnie łączymy całość w jednego stringa i używamy bardzo przydatnej funkcji (eval()) która jest w stanie ze stringa policzyć daną wartość. Łączmy wartości ze stosu w jeden string i wysyłamy go do funkcji eval(). Bardzo ułatwia to implementacje algorytmu obliczeń.

```
    def equal(self):
        expression = "".join(self.stack)
        self.result = eval(expression)
        self.updatetext(str(self.result))
```

Następnie wynik wyświetlamy na ekranie.

Klasa Clock().
Startujemy timer a następnie od przyciśnięcia wyświetlamy albo analogowy albo cyfrowy zegar.

```
def toggle(self):
    self.is_analog=not self.is_analog
    self.update()
```

```
def paintEvent(self, event):
    if self.is_analog:
        self.draw_analog_clock(event)
    else:
        self.draw_digital_clock(event)
```

Oba Widgety pakujemy do głównego widgetu o nazwie “CombinedWidget”,

```
self.style_sheet=True
# Create instances of Calculator and Clock
self.clock = Clock()
self.calculator = Calculator()
# Set up the layout
layout = QHBoxLayout()
layout.addWidget(self.calculator)
layout.addWidget(self.clock)
```

gdzie ustalamy layout oraz ustalamy style dla kalkulatora, tła oraz zegarów.

```
def changestyles(self):
    if self.style_sheet:
        self.setStyleSheet("background-color:blue")
        self.calculator.setStyleSheet("""
            QPushButton{background-color:yellow}
            QPushButton{font:bold}
            QPlainTextEdit{background-color:lightgray}
            QPlainTextEdit{color:black}
            QPlainTextEdit{font:bold}
        """)
        self.clock.change_colors(Qt.green, Qt.red, Qt.white)  # Added digital clock font color
    else:
        self.setStyleSheet("background-color:cyan")
        self.calculator.setStyleSheet("""
            QPushButton{background-color:green}
            QPushButton{font:italic}
            QPlainTextEdit{background-color:lightyellow}
            QPlainTextEdit{color:blue}
            QPlainTextEdit{font:italic}
        """)
        self.clock.change_colors(Qt.darkBlue, Qt.darkMagenta, Qt.black)  # Added digital clock font c
    # Set the layout for this widget
```

Style zegara cyfrowego oraz analogowego są zawarte w funkcji “draw digital clock” w klasie “Clock”, natomiast w change_colors wykonujemy kolorowanie.

```
def change_colors(self, b_color, s_color, clock_font_color):
    self.bColor = b_color
    self.sColor = s_color
    self.clock_style = clock_font_color  # Update the digital clock font color
    self.update()
```

W funkcji draw_digital_clock rysujemy zegar a w funckji draw_analog_clock rysujemy zegar analogowy (oraz je aktualizujemy przez timer wywołany w kontruktorze klasy “Clock”)

```
def __init__(self):
    super().__init__()
    self.is_analog=True
    # creating a timer object
    timer = QTimer(self)
    self.clock_style = Qt.white
    timer.timeout.connect(self.update)

    timer.start(1000)
    self.initializeclock()
```

Do obsługi klawiszy wykorzystaliśmy KeyPress eventy z dokładnie taka sama przypisana akcją jak przyciski:

```
def keyPressEvent(self, e):
    if e.key() == Qt.Key_0:
        self.onButtonClicked("0")
    if e.key() == Qt.Key_1:
        self.onButtonClicked("1")
    if e.key() == Qt.Key_2:
       self.onButtonClicked("2")
    if e.key() == Qt.Key_3:
        self.onButtonClicked("3")
    if e.key() == Qt.Key_4:
        self.onButtonClicked("4")
    if e.key() == Qt.Key_5:
       self.onButtonClicked("5")
    if e.key() == Qt.Key_6:
        self.onButtonClicked("6")
    if e.key() == Qt.Key_7:
        self.onButtonClicked("7")
    if e.key() == Qt.Key_8:
       self.onButtonClicked("8")
    if e.key() == Qt.Key_9:
       self.onButtonClicked("9")
    if e.key() == Qt.Key_Comma or e.key() == Qt.Key_Period:
        self.onButtonClicked(".")
    if e.key() == Qt.Key_Backspace:
        self.erase()
    if e.key() == Qt.Key_C:
        self.clear()
    elif e.key() == Qt.Key_Plus:
        self.onButtonClicked("+")
    elif e.key() == Qt.Key_Minus:
        self.onButtonClicked("-")
    elif e.key() == Qt.Key_Asterisk:
        self.onButtonClicked("*")
```

Dyskusja podjętych rozwiązań:
Nasza aplikacja jest prosta w obsłudze oraz posiada intuicyjne sterowanie. Posiada tak naprawdę wszystko czego wymaga sie od kalkulatora oraz zegara.
Wadami aplikacji jest na pewno brak obsługi nawiasów, przez co nie jest możliwe wykonanie bardziej zaawansowanych obliczeń. Poza tym aplikacja spełnia swoje zadanie w stu procentach