CMPE 230 Systems Programming
Homework 3 Documentation

Volkan Öztürk & Arda Arslan
2019400033 2020400078

Bogazici University, Engineering Faculty
Computer Engineering Department

In this project we designed a classical calculator application using QT framework on C++ language. This project is implemented via 4 steps:

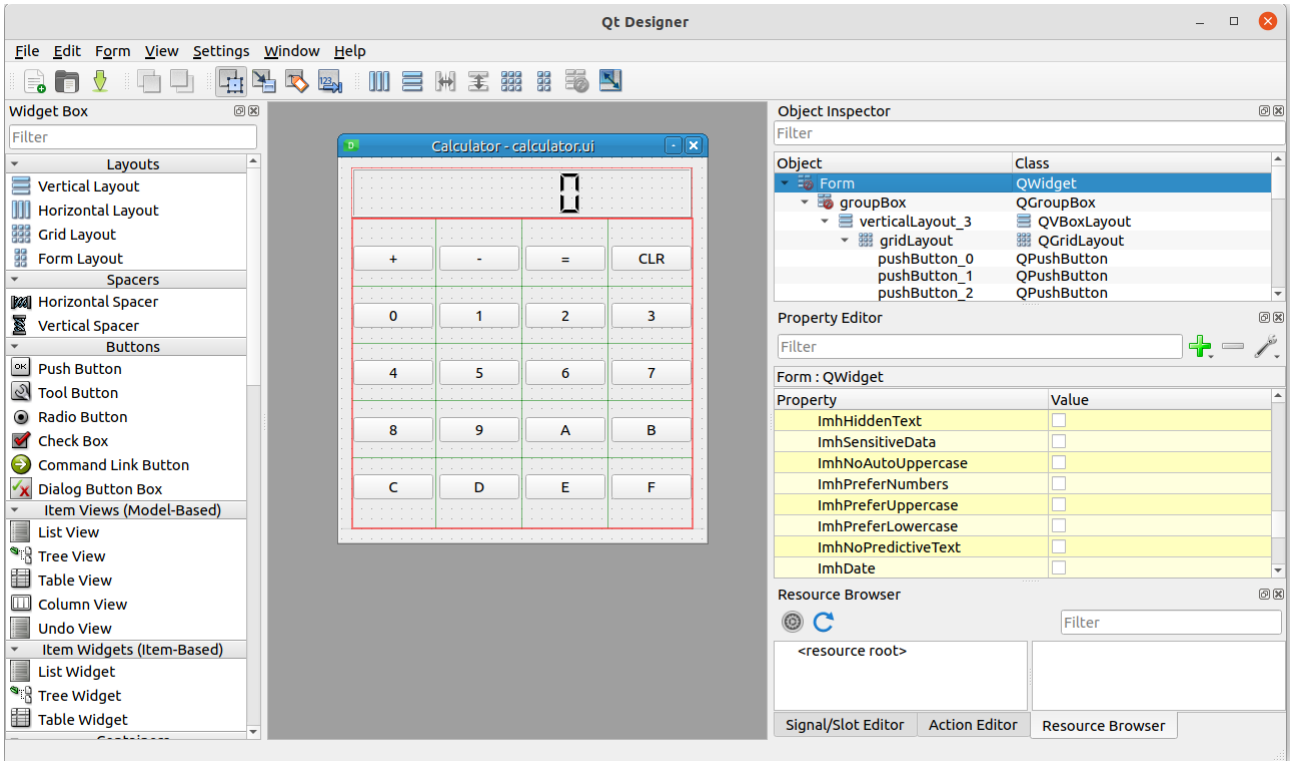
1) **UI Design on QT Designer**
2) **Declaration of Necessary Functions & Variables**
3) **Signal & Slot Connections**
4) **Implentation of Necessary Functions**

IMPORTANT REMARKS

- **After qmake -project command QT += widgets line must be added to .pro file that generated after command.**

UI Design on QT Designer

The UI is designed with QT Designer. Object names are changed accordingly to implement the project.



Declaration of Necessary Functions & Variables

Slot functions and variables of the calculator are declared on .h file. Calculator class is a subclass of QWidget.

Signal & Slot Connections

Pressing hexadecimal digit emits signal that triggers related slot function.

```
connect(ui.pushButton_0, SIGNAL(clicked()), this, SLOT(number_display_0()));
```

Pressing clear button emits signal that triggers related slot function.

```
connect(ui.pushButton_clear, SIGNAL(clicked()), this, SLOT(clear_display()));
```

Pressing add button emits signal that triggers related slot function. Similar for minus and equals buttons.

```
connect(ui.pushButton_plus, SIGNAL(clicked()), this, SLOT(add_display()));
```

Implementation of Necessary Functions

For detailed explanations of functions, please go for comments.

```
/*
number_display_i()

Gets the current number's value, multiplies by 16 to shift left and adds i, given by the user.

NOTE:   If operator equal sign is given before pushing buttons;
        operand, current operator and current number are reset to zero,
        in order to reflect initialization of a new calculation.
*/
void Calculator::number_display_0() {
    if (current_operator == -2) {
        current_number = 0;
        current_operator = 0;
        operand = 0;
    }
    current_number = (current_number * 16) + 0;
    ui.LCDdisplay -> display(current_number);
}
```

```
/*
    Clears the display by refreshing all the data stored in:
        int current_number;
        int current_operator;
        int operand;

*/
void Calculator::clear_display() {
    current_number = 0;
    current_operator = 0;
    operand = 0;
    ui.LCDdisplay -> display(current_number);
}
```

```
/*
    If <current_operator> is not null, calculates the operation stored in <current_operator>
        with the operands <current_num> and <operand>.
    Else if <current_operator> is null stores the value of <current_number> on <operand>.

    Updates the <current_operator> with plus sign and resets <current_number>.

    NOTE: Result of the calculation is reflected on <operand>.
*/
void Calculator::add_display() {
    if (current_operator) {
        if (current_operator == 1) {
            operand = current_number + operand;
            current_number = 0;
            current_operator = 1;
        }else if (current_operator == -1) {
            operand = operand - current_number;
            current_number = 0;
            current_operator = 1;
        }else {
            current_operator = 1;
        }
    }else {
        operand = current_number;
        current_number = 0;
        current_operator = 1;
    }
    ui.LCDdisplay -> display(operand);
}
```