Signals and slots:

made by: -Nour el houda Darbal-Khaoula Mzoudi-

supervised by: -pr.Belcaid-

20 November 2021

Definition:

Signals and Slots are an implementation of the observer design pattern used by the Qt and Wt software libraries.The concept is that objects, if their classes are declared correctly, can emit signals, whether or not they contain information. In turn, other objects can receive these signals via slots if they are explicitly connected to these signals.

Objectif

1. This exercise follows up to add **interactive** functionality to the **calculator** widgets written in the previous homework. The goal is to use Signals and Slots to simulate a basic calculator behavior. The supported operations are `*, +, -, /`.

Projects

We have in this work one project which are: "calculator" .

explanation:

Set up

In the **starter project** Callcultaor.zip . we find a Qt project to create the main widget as a custom class to configure and Run the project. we should see a main widget with the calculator buttons.

In order to have a computing functionality, we will represent any mathematical operation by:

```
left    (op)   right
```

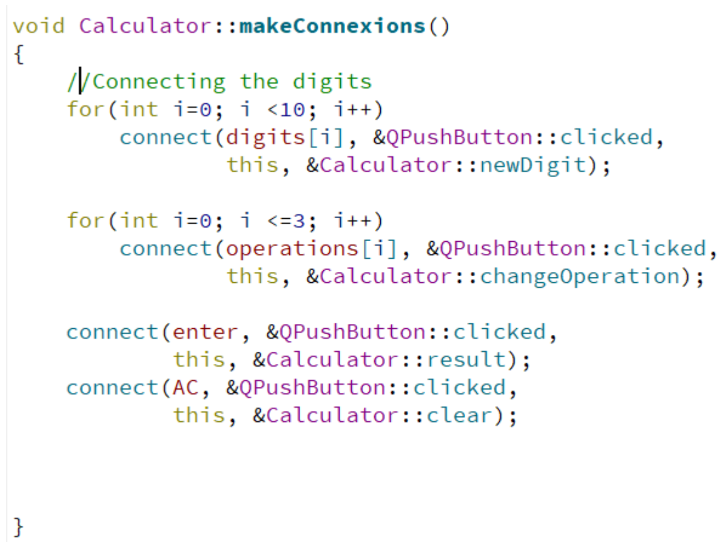
```
// Where to display the numbers
int * left=nullptr;          //left operand
int * right=nullptr;         // right operand
QString *operation=nullptr;  // Pointer on the current operation
```

to map multiple signals to the same slot. The slot has to behave **differently** according the which digit was pressed.we use the function “new Digit” :

```
void Calculator::newDigit( )
{
    //getting the sender
    auto button = dynamic_cast<QPushButton*>(sender());

    //getting the value
    int value = button->text().toInt();
```

after we should make connexion of **all the buttons** to this slot. This function will use the `Sender` method to get the identity of which button was clicked and act accordingly.



```
void Calculator::makeConnexions()
{
    //Connecting the digits
    for(int i=0; i <10; i++)
        connect(digits[i], &QPushButton::clicked,
                this, &Calculator::newDigit);

    for(int i=0; i <=3; i++)
        connect(operations[i], &QPushButton::clicked,
                this, &Calculator::changeOperation);

    connect(enter, &QPushButton::clicked,
            this, &Calculator::result);
    connect(AC, &QPushButton::clicked,
            this, &Calculator::clear);



}
```

Now that we can react to each digit,we should know two things important to implement “New Digit” correctly:

-Which number should be constructed `left` or `right`:The response to this question is easy. If we have an operation, then we already have our **left operand** and we should focus on the right.

-How to add a digit to an existing number:Suppose we are working on the left = 43. What would happen if we clicked digit 2. Simply we should move all digit by one digit **(x10)** and then **add** the Programmatically speaking this could done by :

```
  *left = (*left) * 10 + digit
```

```
void Calculator::newDigit( )
{
    //getting the sender
    auto button = dynamic_cast<QPushButton*>(sender());

    //getting the value
    int value = button->text().toInt();

    //Check if we have an operation defined
    if(operation)
    {
        //check if we have a value or not
        if(!right)
            right = new int{value};
        else
            *right = 10 * (*right) + value;

        disp->display(*right);

    }
    else
    {
        if(!left)
            left = new int{value};
        else
            *left = 10 * (*left) + value;

        disp->display(*left);
    }
}
```

Using the `sender` method,we should now move on the `operation` of the four buttons:

```
void Calculator::changeOperation()
{
    //Getting the sender button
    auto button = dynamic_cast<QPushButton*>(sender());

    //Storing the operation
    operation = new QString{button->text()};

    //Initiating the right button
    right = new int{0};

    //reseting the display
    disp->display(0);
}
```

Also, we should implement the `slot` for the enter button to compute the result of combining the left and right value according to the operation:

```
void Calculator::result()
{
     auto button = dynamic_cast<QPushButton*>(sender());
     enter = new QPushButton{button};


     if(*operation=="+") disp->display(*left + *right) ;
     else if(*operation=="/") disp->display(*left / *right) ;
     else if(*operation=="-") disp->display(*left - *right) ;
     else if(*operation=="*") disp->display(*left * *right) ;

}
```

The final touch is add the Button AC which allows us to erase everything and return all the elements to 0:

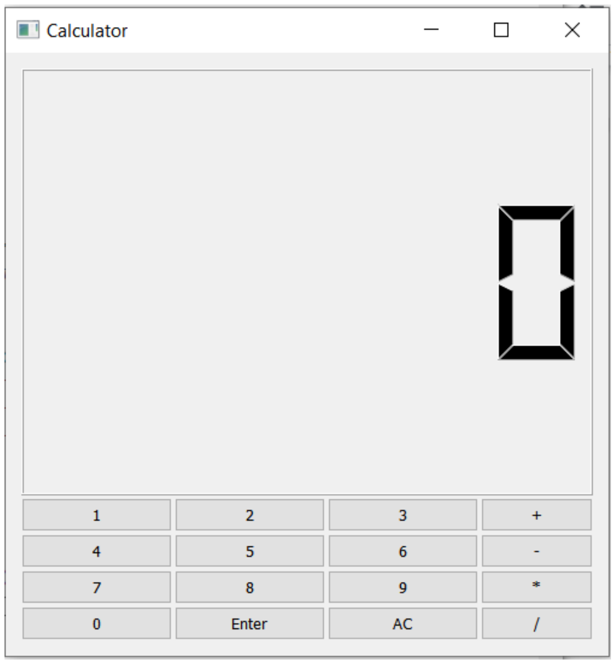
```
void Calculator::clear()
{
    auto button = dynamic_cast<QPushButton*>(sender());
    AC = new QPushButton{button};

    left=nullptr;
    operation=nullptr;
    right=nullptr;

    disp->display(0);


}
```

here we will see the final version of the project:



Thank You!