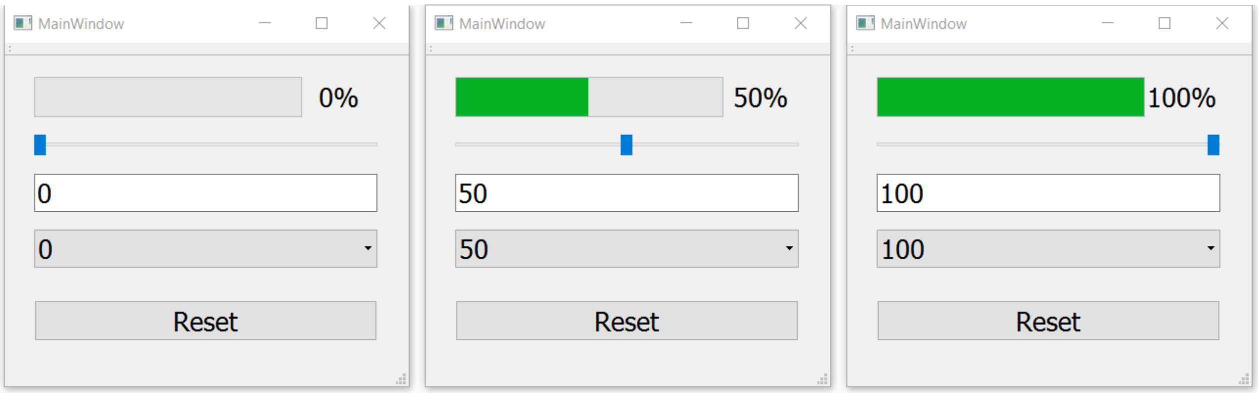
Homework 17

The goal of this homework is for you to practice collaboration modeling as well as the mediator pattern. For this homework, fill in this word file with your answers. I suggest you to work on this problem purely on paper, with your computer and textbook closed, because a problem of this style may be given in Exam 2. The images shown below are screen shots of a simple application. The application starts with the main window that contains the following objects: a progress bar, a horizontal slider, a text input, a combo box and a push button. The progress bar ranges from 0 to 100. The slider range is from 0 to 100 and its default location is the 0th position. The text input shows the value of slider position, it can also be used to change the slider position, progress bar and combo box values. The combo box contains three options; 0, 50, and 100. Changes in combo box selections are immediately reflected in all other objects. The reset button resets the values of all objects to zero.

(1) Complete the code given below to implement the functionality described above. For your convenience, parts of the code are already given to you while some functions have been removed and replaced with empty boxes that you need to fill. Here is a list of QT signals for your reference: textChanged(QString), currentTextChanged(QString), valueChanged(QString), textEdited(QString), returnPressed(), editingFinished(), clicked(bool), clicked(), pressed(), released(), stateChanged(int)



Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
namespace Ui {
class MainWindow;  }

class MainWindow : public QMainWindow
{  Q_OBJECT
public:
  explicit MainWindow(QWidget *parent = 0);
```

```
    ~MainWindow();
  public slots:
    void actByYourChange(QObject*);
  private:
    Ui::MainWindow *ui;
  };
  #endif // MAINWINDOW_H
```

Mainwindow.cpp

```
  #include "mainwindow.h"
  #include "ui_mainwindow.h"
  #include "myslider.h"
  #include "mylineedit.h"
  #include "mycombobox.h"

  MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
  {
    ui->setupUi(this);
    ui->horizontalSlider->setMinimum(0);
    ui->horizontalSlider->setMaximum(100);
    ui->horizontalSlider->setValue(0);
    ui->progressBar->setMinimum(0);
    ui->progressBar->setMaximum(100);
    ui->progressBar->setValue(0);
    ui->lineEdit->setText("0");
    connect(ui->comboBox,SIGNAL(currentTextChanged(Qstring)),ui->comboBox,
    SLOT(myTextChanged()));
    connect(ui->comboBox,SIGNAL(iChanged(QObject*)),this,
    SLOT(actByYourChange(QObject*)));

    connect(ui->lineEdit,SIGNAL(editingFinished()),ui->lineEdit,
    SLOT(myEditingFinished()));
    connect(ui->lineEdit,SIGNAL(iChanged(QObject*)),this,
    SLOT(actByYourChange(QObject*)));

    connect(ui->horizontalSlider,SIGNAL(valueChanged(Qstring)),ui->horizontalSlider,
    SLOT(myValueChanged()));
    connect(ui->horizontalSlider,SIGNAL(iChanged(QObject*)),this,
    SLOT(actByYourChange(QObject*)));

    connect(ui->resetButton,SIGNAL(clicked()),ui->resetButton,
    SLOT(myButtonClicked()));
    connect(ui->resetButton,SIGNAL(iChanged(QObject*)),this,
    SLOT(actByYourChange(QObject*)));
```

```
}
 MainWindow::~MainWindow()
 {   delete ui;
  }

 void MainWindow::actByYourChange(QObject* sender){
```

```
if(senderObj==ui->comboBox){
    ui->progressBar->setValue((int)ui->comboBox->currentText());
    ui->horizontalSlider->setValue((int)ui->comboBox->currentText());
    ui->lineEdit->setText(ui->comboBox->currentText());
}
else if(senderObj==ui->lineEdit){
    ui->progressBar->setValue((int)ui->lineEdit->currentText());
    ui->horizontalSlider->setValue((int)ui->lineEdit->currentText());
    ui->comboBox->setCurrentText(ui->lineEdit->currentText());
}
else if(senderObj==ui->horizontalSlider){
    ui->progressBar->setValue(ui->horizontalSlider->currentValue());
    ui->comboBox->setCurrentText(ui->horizontalSlider->currentValue());
    ui->lineEdit->setText(ui->horizontalSlider->currentValue());
}
else if(senderObj==ui->resetButton){
    ui->progressBar->setValue(0);
    ui->horizontalSlider->setValue(0);
    ui->comboBox->setCurrentText(0);
    ui->comboBox->setCurrentText("0");
}
```

```
  ui->centralWidget->adjustSize();
```

```
    }
```

Mycombobox.h

```
    #ifndef MYCOMBOBOX_H
    #define MYCOMBOBOX_H
    #include <QComboBox>

    class mycomboBox: public QComboBox {
    Q_OBJECT
    public:
      mycomboBox(QWidget* qw):QComboBox(qw){};
    signals:
     void iChanged(QObject*);

   public slots:
     void myTextChanged(QString*);

    };
    #endif // /MYCOMBOBOX_H
```

Mycombobox.cpp

```
    #include "mycombobox.h"
     void mycomboBox::myTextChanged(const QString&){
         emit iChanged(this);
     }
```

MylineEdit.h

```
    #ifndef MYLINEEDIT_H
    #define MYLINEEDIT_H
    #include <QLineEdit>//box
    class MyLineEdit:public QLineEdit{
    Q_OBJECT
    public:
      MyLineEdit(const QString& qs):QLineEdit(qs){};
      MyLineEdit(QWidget* qw):QLineEdit(qw){};

    signals:
     void iChanged(QObject*);
```

```
public slots:
```

```
void myEditingFinished();
```

```
};
#endif // MYLINEEDIT_H
```

MylineEdit.cpp

```
#include "mylineedit.h"
```

```
void MyLineEdit::myEditingFinished(){
    emit iChanged(this);
}
```

MySlider.h

```
#ifndef MYSLIDER_H
#define MYSLIDER_H

#include <QSlider>//box here
class MySlider:public QSlider {
Q_OBJECT
public:
  MySlider(QWidget* qw):QSlider(qw){}
signals:
```

```
void iChanged(QObject*);
```

```
public slots:
```

```
void myValueChanged();
```

```
 };
#endif // MYSLIDER_H
```

MySlider.cpp

```
#include "myslider.h"
```

```
void MySlider::myValueChanged(){
    emit iChanged(this);
}
```

Resetbutton.h

```
#ifndef RESETBUTTON_H
#define RESETBUTTON_H
#include<QPushButton>
class ResetButton:public QPushButton  {
Q_OBJECT
public:
  ResetButton(QWidget* qw):QPushButton(qw){}
signals:
```

```
void iChanged(QObject*);
```

```
private slots:
```

```
void myButtonClicked();
```

```
};
#endif // RESETBUTTON_H
```

Resetbutton.cpp

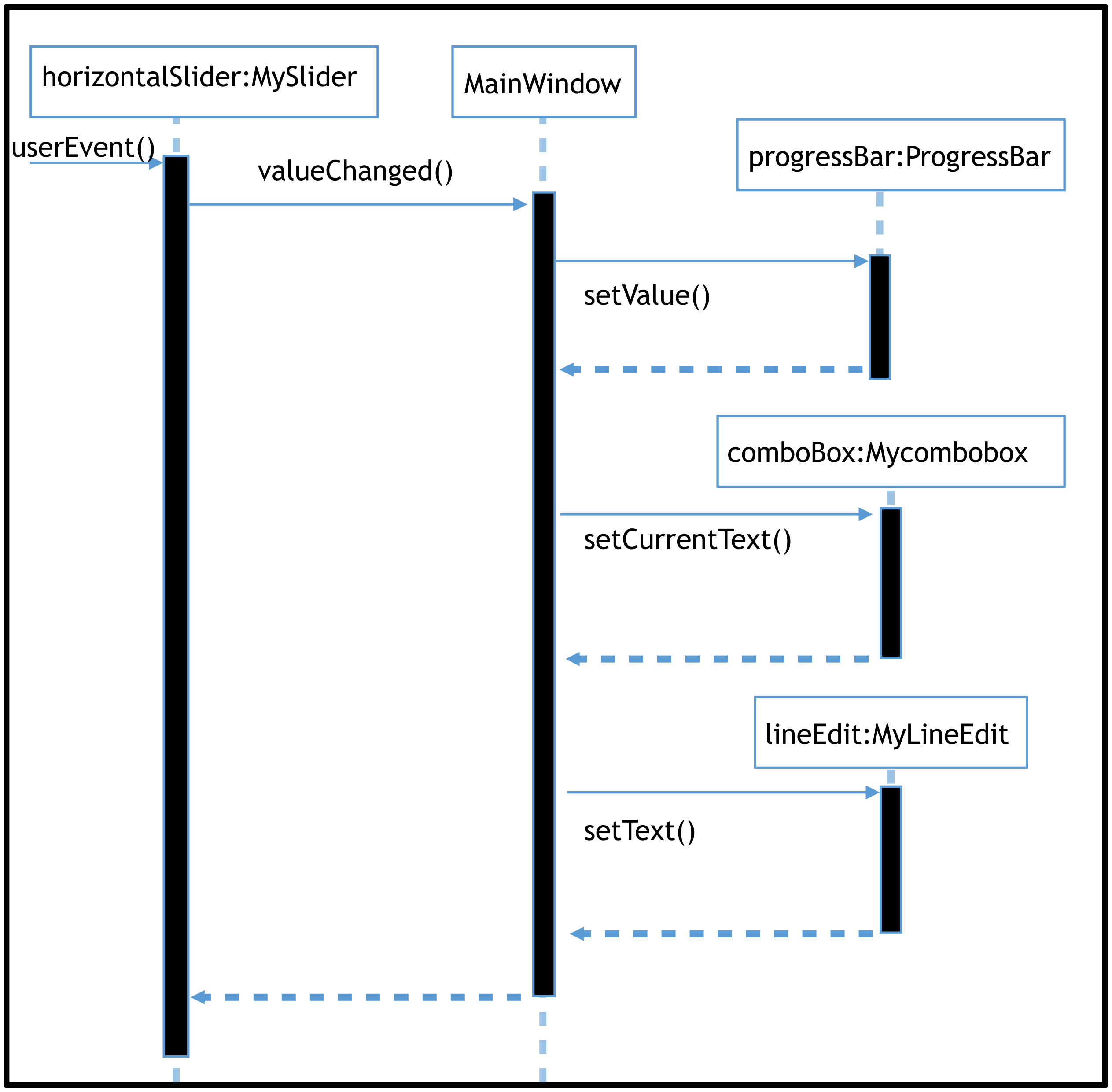
```
#include "resetbutton.h"
```

```
void ResetButton::myButtonClicked(){
    emit iChanged(this);
}
```

(2) Sequence Diagram

Consider the above application that uses a slider bar, progress bar, text input, combobox and a push button object. Draw a sequence diagram that depicts the following scenario: The main dialog window is open and a user changes slider position in main window. Please draw the diagram in word. We do not accept handwritten diagrams.

Solution:



Due: April 11, 11:59PM, 2018.
Fill in your answer in this word file, and then save it as a PDF file, and submit the PDF file via handin. The name of your word file should be: LastName_FirstName.pdf. For example, if your name is John Smith, you should turn in one file: Smith_John.pdf.