C++&Qt day2 hw2 결과 보고서

2025407012/로봇학부/송연우

목차

1. 구현 기능-Queue: 은행 입금/출금 시뮬레이션

1) vector

: STL의 vector를 사용해 은행 계좌를 각 vector의 index로 설정하고, 사용자가 입력하는 money값을 각 index 칸에 저장하고 편집할 수 있게 했습니다.

2) 버튼 클릭/스핀 박스 입력을 통한 은행 입/출금 기능

: 입력하는 수 범위가 1000000000까지인 스핀 박스를 생성해 사용자의 입력을 받고, 출금/입금 버튼을 눌렀을 때 사용자 잔고에 알맞은 메시지가 출력되도록 했습니다. Queue 자료형의 특성인 FIFO을 구현하기 위해 입금 버튼을 누르면 index가 0부터 계좌가 생성되어 값이 저장되고, 출금 버튼을 누르면 index 0부터 출금할 수 있게 합니다.

코드를 다 작성한 후에 깨달은 아쉬운 점이 있었는데, 완성된 코드가 가장 처음 입력한 값이 다 빠지지 않은 상태에서 바로 다음 값으로 넘어가는 식으로 구성되어 Queue 구조 구현이 부족했습니다.

2. hpp 파일

다음은 작성한 헤더파일 코드입니다.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <iostream>
#include <vector>
using namespace std;

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void Push(int money);
    void Pop(int money, int &Pop_index);


private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_spinBox_valueChanged(int arg1);

private:
    Ui::MainWindow *ui;
    vector<int> bank_account;
    int money;
    int Push_index;
    int Pop_index;
};
#endif // MAINWINDOW_H
```

Vector, iostream을 include해 사용하였고 클래스 MainWindow 아래에서 함수들을 선언했습니다. public 부분에서는 슬롯 함수를 제외한 생성자/소멸자/Push/Pop 함수를 만들었고 UI 슬롯은 버튼 2개와 스핀 박스 1개 함수를 사용했습니다. 나머지 private 부분에서는 back_account와 사용자 입력을 받을 money, 입/출금 시 인덱스 각각을 선언했습니다.

3. cpp 파일

다음은 작성한 소스파일입니다. main.cpp에서는 수정 사항이 없으므로 mainwindow.cpp 파일을 보겠습니다.

```cpp
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <iostream>

using namespace std;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui->spinBox, SIGNAL(valueChanged(int)), ui->lcdNumber, SLOT(display(int)));
    Push_index=0;
    Pop_index=0;
    money=0;

}

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

void MainWindow::Push(int money){
    bank_account.push_back(money);
    cout<<"계좌 "<<Push_index<<"입금 내역:"<<money<<endl;
    Push_index++;
}
```

생성자와 소멸자, Push함수입니다. 생성자에서는 스핀 박스에서 감지하는 변동된 값 arg1을 lcd에 바로 표시하도록 스핀박스에서 시그널이 발생하면 lcd를 사용하도록 슬롯으로 연결해 줍니다. 나머지 입/출금 인덱스는 -으로 초기화해 주고 money값도 0으로 초기화 시킵니다.

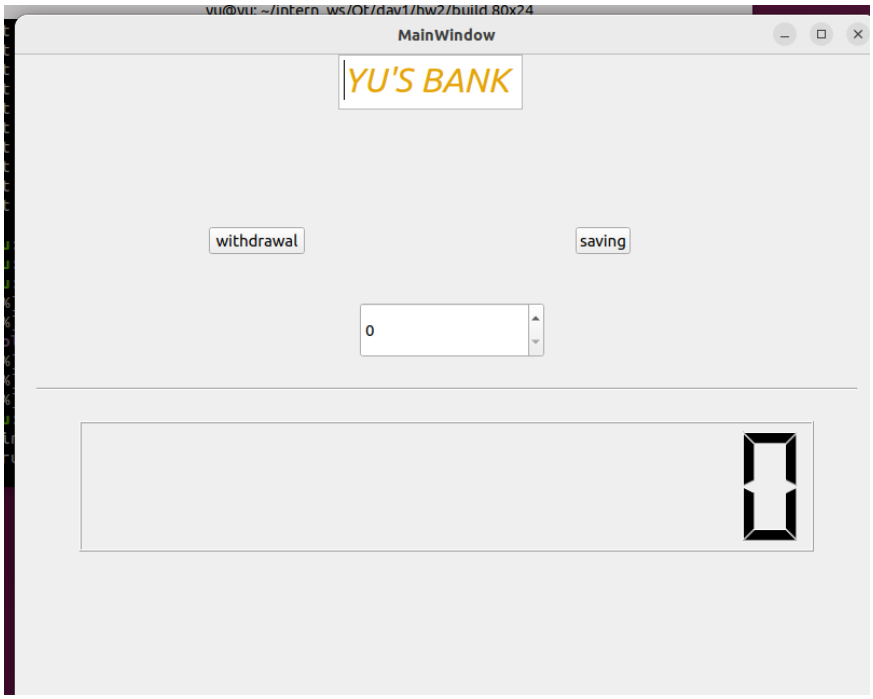
```
30  void MainWindow::Pop(int money, int &Pop_index){
31      if(Push_index >= 0 && Pop_index <bank_account.size()){
32          if(bank_account.at(Pop_index)>=money){
33              cout<<Pop_index<<"-인출: "<<money<<endl;
34              bank_account.at(Pop_index)-=money;
35              Pop_index++;
36          }else{
37              cout<<"잔고 부족-잔고"<<':'<<bank_account[Pop_index]<<endl;
38          }
39      }else{
40          cout<<"Wrong Access!"<<endl;
41      }
42  }
43
44
45  void MainWindow::on_pushButton_clicked()
46  {
47      //saving
48      Pop(money,Pop_index);
49  }
50
51
52  void MainWindow::on_pushButton_2_clicked()
53  {
54      //withdrawal
55      Push(money);
56  }
57
58
59  void MainWindow::on_spinBox_valueChanged(int arg1)
60  {
61      cout<<arg1<<endl;
62      money=arg1;
63  }
```

소스파일 나머지 부분입니다. Pop 함수에서는 생성된 vector 인덱스가 존재하고, 출금 인덱스가 vector 전체 인덱스를 넘지 않을 때만 계좌에 접근이 가능하며, 접근이 불가능할 때는 Wrong Access 오류 메시지를 띄웁니다. 계좌에 접근해 사용자 입력으로 money를 받고, money가 해당 계좌의 입금된 값 이하일 때만 인출이 가능합니다. 사용자가 계좌에 저장된 값을 초과하여 인출하려고 할 때는 오류 메시지가 출력됩니다.

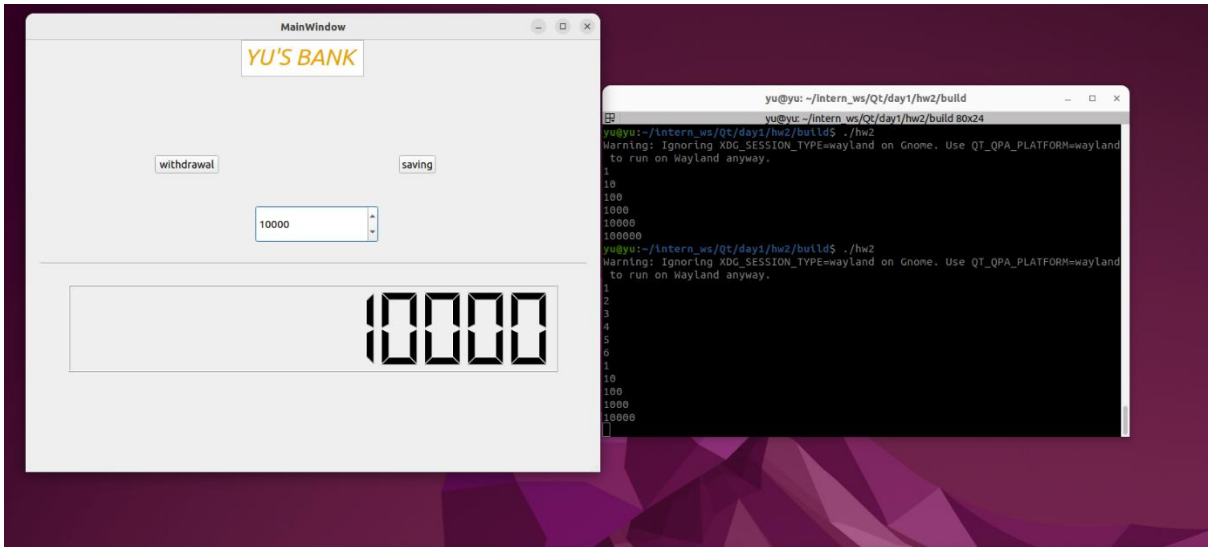
밑의 함수는 버튼1, 2가 클릭되었을 때 Pop, Push 함수를 실행하도록 한 부분이고, 스핀 박스에서 변화가 감지되었을 때 값을 money에 저장하는 슬롯 함수도 만들었습니다.

4. 실행 결과

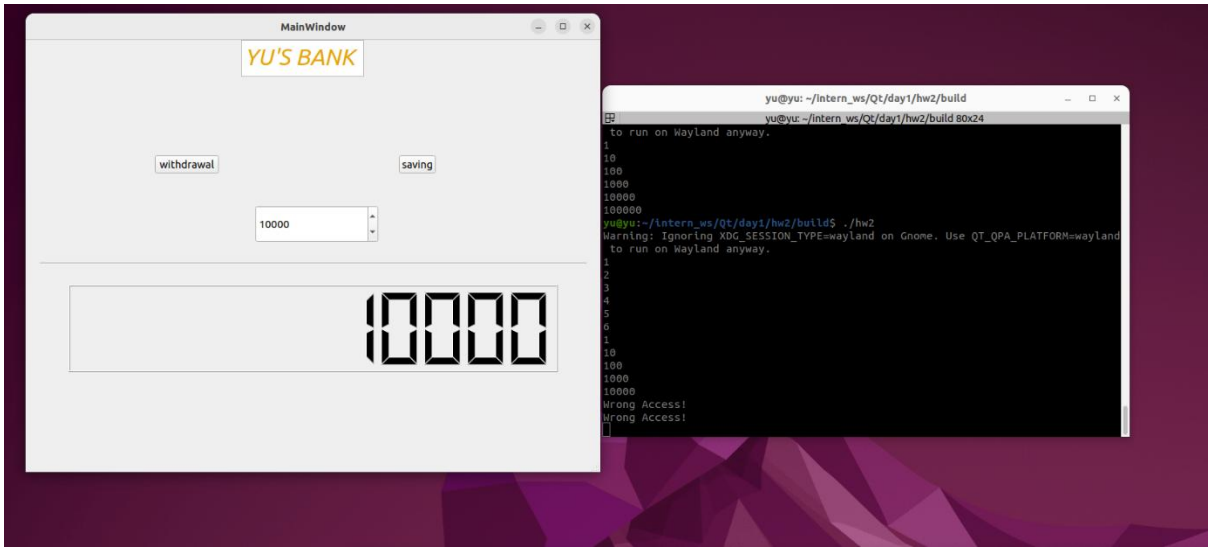
프로젝트를 빌드해 실행한 처음 화면입니다.



맨 위 텍스트 박스는 아무 기능이 없고, 왼쪽에는 출금 버튼이, 오른쪽에는 입금 버튼이 위치해 있습니다. 중앙의 스핀 박스로 입/출금 금액을 입력할 수 있고 아래의 lcd에서도 금액이 갱신되어 나타납니다.

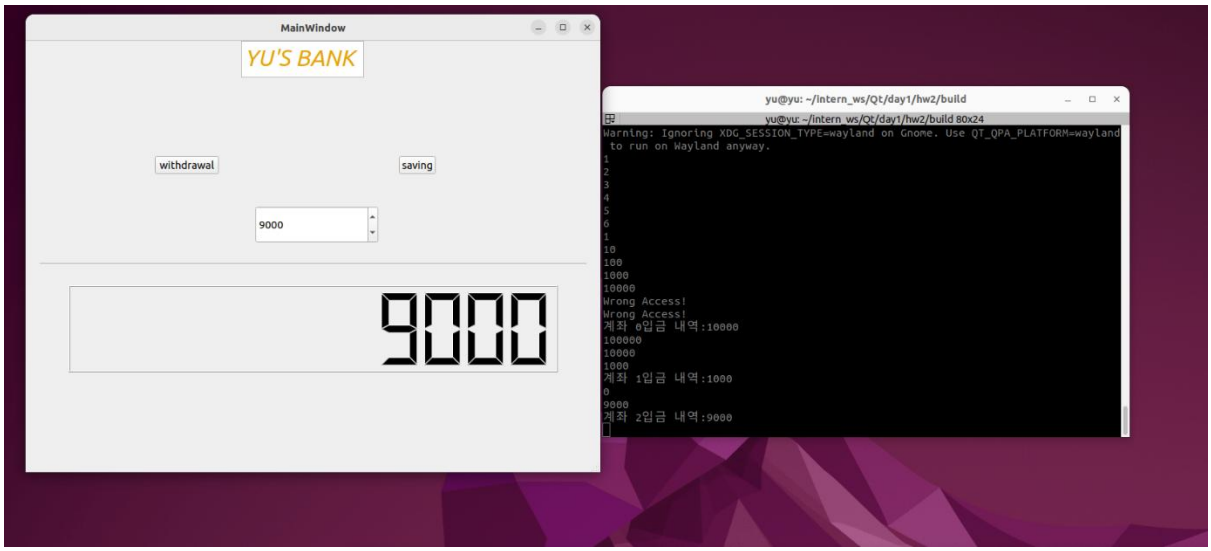


다음과 같이 터미널 창에서도 money값이 갱신됩니다. 계좌가 없는 상태에서 출금 버튼을 눌러보겠습니다.



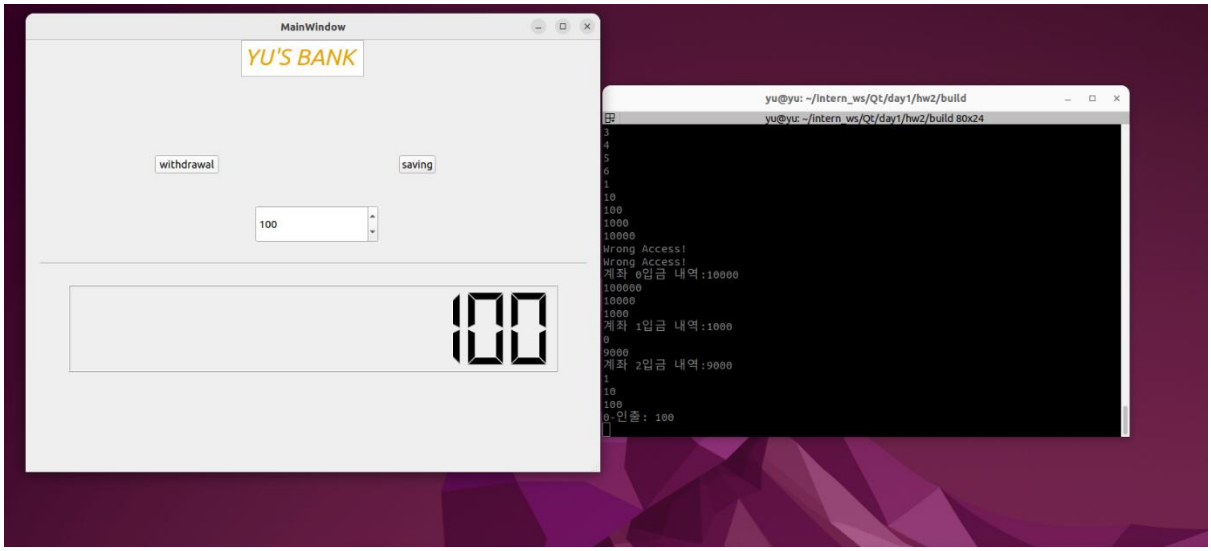
출금할 계좌가 없으므로 터미널 창에 오류 메세지가 출력됩니다.

이제 입금 버튼을 눌러보겠습니다.

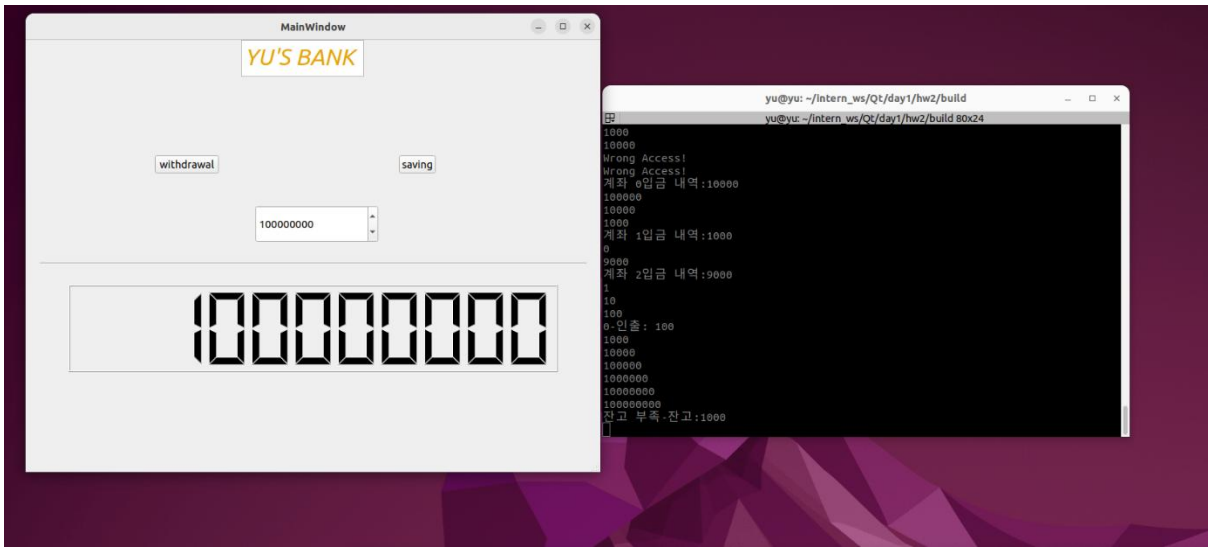


입금한 계좌 index값과 저장된 값이 터미널에 출력됩니다.

다시 인출 버튼을 눌러보겠습니다.



100원만을 인출했으므로 정상적으로 작동합니다.



만약 너무 큰 값을 인출하려고 하면 잔고 부족 메시지가 출력됩니다.