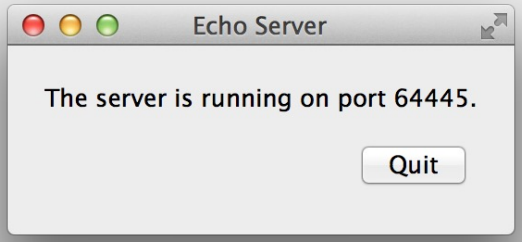
Echo 서버 : widget.h

```
#include <QLabel>
#include <QTcpServer>
#include <QWidget>

class Widget : public QWidget
{
    Q_OBJECT
public:
    Widget(QWidget *parent = 0);

private slots:
    void clientConnect( );                  /* 에코 서버 */
    void echoData( );

private:
    QLabel *infoLabel;
    QTcpServer *tcpServer;
};
```

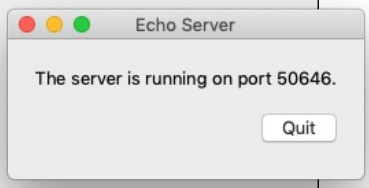


Echo 서버 : widget.cpp(1)

```
#include <QtGui>
#include <QtWidgets>
#include <QtNetwork>
#include "widget.h"

#define BLOCK_SIZE      1024

Widget::Widget(QWidget *parent) : QWidget(parent)
{
    infoLabel = new QLabel(this);
    QPushButton *quitButton = new QPushButton("Quit", this);
    connect(quitButton, SIGNAL(clicked( )), qApp, SLOT(quit( )));

    QHBoxLayout *buttonLayout = new QHBoxLayout;
    buttonLayout->addStretch(1);
    buttonLayout->addWidget(quitButton);

    QVBoxLayout *mainLayout = new QVBoxLayout(this);
```

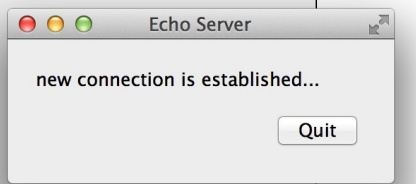
Echo 서버 : widget.cpp(2)

```
    mainLayout->addWidget(infoLabel);
    mainLayout->addLayout(buttonLayout);
    setLayout(mainLayout);

    tcpServer = new QTcpServer(this);
    connect(tcpServer, SIGNAL(newConnection( )), SLOT(clientConnect( )));
    if (!tcpServer->listen( )) {
        QMessageBox::critical(this, tr("Echo Server"), \
                              tr("Unable to start the server: %1.") \
                              .arg(tcpServer->errorString( )));
        close( );
        return;
    }

    infoLabel->setText(tr("The server is running on port %1.")
                                .arg(tcpServer->serverPort( )));
    setWindowTitle(tr("Echo Server"));
}
```



Echo 서버 : widget.cpp(3)

```
void Widget::clientConnect( )
{
    QTcpSocket *clientConnection = tcpServer->nextPendingConnection( );
    connect(clientConnection, SIGNAL(disconnected( )), \
            clientConnection, SLOT(deleteLater( )));
    connect(clientConnection, SIGNAL(readyRead( )), SLOT(echoData( )));
    infoLabel->setText("new connection is established...");
}

void Widget::echoData( )
{
    QTcpSocket *clientConnection = dynamic_cast<QTcpSocket *>(sender( ));
    if (clientConnection->bytesAvailable( ) > BLOCK_SIZE) return;
    QByteArray bytearray = clientConnection->read(BLOCK_SIZE);
    clientConnection->write(bytearray);
    infoLabel->setText(QString(bytearray));
}
```

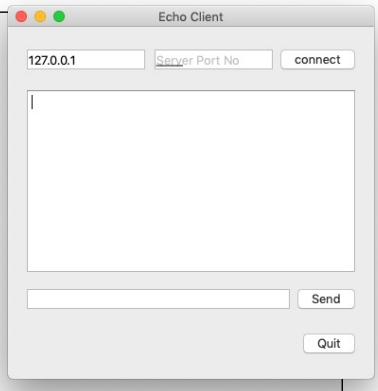


클라이언트는 텔넷(Telnet)을 사용하면 편리 : telnet IP_ADDRESS [PORT_NO]

Echo 클라이언트 : widget.h

```
#include <QWidget>
#include <QTextEdit>
#include <QLineEdit>
#include <QTcpSocket>

class Widget : public QWidget {
    Q_OBJECT
public:
    Widget(QWidget *parent = 0);
    ~Widget( );

private slots:
    void echoData( );                   // 서버에서 데이터가 올 때
    void sendData(  );                  // 서버로 데이터를 보낼 때

private:
    QTextEdit *message;                 // 서버에서 오는 메세지 표시용
    QLineEdit *inputLine;               // 서버로 보내는 메시지 입력용
    QTcpSocket *clientSocket;           // 클라이언트용 소켓
};
```


Echo 클라이언트 : widget.cpp(1)




```
#include "widget.h"

#include <QtGui>
#include <QtWidgets>
#include <QtNetwork>

#define BLOCK_SIZE      1024

Widget::Widget(QWidget *parent) : QWidget(parent) {
    // 연결한 서버 정보 입력을 위한 위젯들
    QLineEdit *serverAddress = new QLineEdit(this);
    serverAddress->setText("127.0.0.1");
//    serverAddress->setInputMask("999.999.999.999;_");
    QRegularExpression re("^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\."
                          "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\."
                          "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\."
                          "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$|");
    QRegularExpressionValidator validator(re);
    serverAddress->setPlaceholderText("Server IP Address");
```

Echo 클라이언트 : widget.cpp(2)

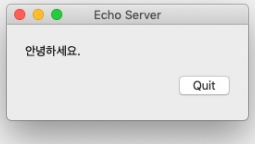
```
QLineEdit *serverPort = new QLineEdit(this);
serverPort->setInputMask("00000;_");
serverPort->setPlaceholderText("Server Port No");

QPushButton *connectButton = new QPushButton("connect", this);
connect(connectButton, &QPushButton::clicked,
        [=]{ clientSocket->connectToHost(serverAddress->text( ),
                                         serverPort->text( ).toInt( )); } );

QHBoxLayout *serverLayout = new QHBoxLayout;
serverLayout->addStretch(1);
serverLayout->addWidget(serverAddress);
serverLayout->addWidget(serverPort);
serverLayout->addWidget(connectButton);

message = new QTextEdit(this);                    // 서버에서 오는 메시지 표시용
message->setReadOnly(true);

// 서버로 보낼 메시지를 위한 위젯들
inputLine = new QLineEdit(this);
QPushButton *sentButton = new QPushButton("Send", this);
connect(sentButton, SIGNAL(clicked( )), SLOT(sendData( )));
```

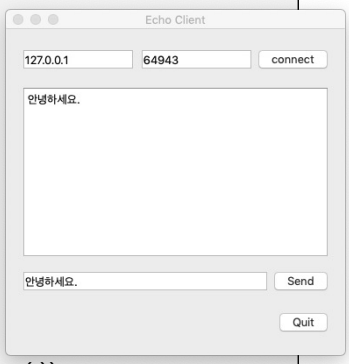

Echo 클라이언트 : widget.cpp(3)

```
QHBoxLayout *inputLayout = new QHBoxLayout;
inputLayout->addWidget(inputLine);
inputLayout->addWidget(sentButton);

// 종료 기능
QPushButton *quitButton = new QPushButton("Quit", this);
connect(quitButton, SIGNAL(clicked( )), qApp, SLOT(quit( )));

QHBoxLayout *buttonLayout = new QHBoxLayout;
buttonLayout->addStretch(1);
buttonLayout->addWidget(quitButton);

QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->addLayout(serverLayout);
mainLayout->addWidget(message);
mainLayout->addLayout(inputLayout);
mainLayout->addLayout(buttonLayout);

setLayout(mainLayout);
```

Echo 클라이언트 : widget.cpp(4)

```
    clientSocket = new QTcpSocket(this);                    // 클라이언트 소켓 생성
    connect(clientSocket, &QAbstractSocket::errorOccurred,
                          [=]{ qDebug( ) << clientSocket->errorString( ); });
    connect(clientSocket, SIGNAL(readyRead( )), SLOT(echoData( )));
    setWindowTitle(tr("Echo Client"));
}

Widget::~Widget( )
{
    clientSocket->close( );
}

void Widget::echoData( )
{
    QTcpSocket *clientSocket = dynamic_cast<QTcpSocket *>(sender( ));
    if (clientSocket->bytesAvailable( ) > BLOCK_SIZE) return;
    QByteArray bytearray = clientSocket->read(BLOCK_SIZE);
    message->append(QString(bytearray));
}
```





Echo 클라이언트 : widget.cpp(5)

```
void Widget::sendData(  )
{
    QString str = inputLine->text( );
    if(str.length( )) {
        QByteArray bytearray;
        bytearray = str.toUtf8( );
        clientSocket->write(bytearray);
    }
}
```

- 과제 : 위의 에코 서버를 이용해서 여러 명이 채팅을 할 수 있는 채팅 서버를 완성하시오.
  - 아이디를 사용해서 사용자를 구분