```
// Projekt      BROS 4 semester semesterprojekt
//
// Fil          tcp.cpp
//
// Beskrivelse  Implementering af klassen tcp
//
// Forfatter    MH
//
// Version      3.0 - oprindelig version


#include "tcp.h"
#include "server.h"
#include "enumdekrypteringskode.h"
#include <QListWidgetItem>
#include <QString>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <iostream>

using namespace std;

Tcp::Tcp(Server*s)
{

    connect(this, SIGNAL(addLogEntry(QString)), s, SLOT(addLogEntry
        (QString)));
    connect(&server, SIGNAL(newConnection()),
                this, SLOT(acceptConnection()));
    server.listen(QHostAddress::Any, 9999);

    addLogEntry("Klar til tilslutning");
    qDebug()<<"Klar til tilslutning";
}

void Tcp::acceptConnection(void)
{
    client = server.nextPendingConnection();

    connect(client, SIGNAL(readyRead()),
            this, SLOT(startRead()));
    QListWidgetItem* connected =new QListWidgetItem("Tilsluttet og klar
        til datamodtagelse");
    connected->setForeground(Qt::blue);
    addLogEntry("Skib tilsluttet");
    qDebug()<<"Socket køre";
    qDebug()<<"KI tilsluttet";

}

void Tcp::startRead(void)
{
    qDebug() <<"read";


    QString clientMSG_ = client->readLine(20);
    client->waitForReadyRead();
```

```
 qDebug()<<"Data fra skib, inden split: "<<clientMSG_;
 qDebug()<<"Data efter split: ";
 string clientMSG = clientMSG_.toStdString();


 char one_line_string[100] ;


 {

     char * writable = new char[clientMSG.size()+1];
     copy(clientMSG.begin(),clientMSG.end(),writable);
     writable[clientMSG.size()] = '\0';
     cout<<writable<<endl;
     for(int i= 0; i< clientMSG.size()+1; i++)
     {
         one_line_string[i] = writable[i];

     }

     delete[] writable;

 }


 cout<<one_line_string<<endl;

char seps[]   = " ,\t\n";
char *token;
vector<string> vec_String_Lines;
token = strtok( one_line_string, seps );

while( token != NULL )
{
   vec_String_Lines.push_back(token);
   token = strtok( NULL, seps );
}

cout<<"Data efter split: "<<endl;
cout<<"ID: " <<vec_String_Lines[0]<<endl;
cout<<"STYRBORD: " <<vec_String_Lines[1]<<endl;
cout<<"BAGBORD: " <<vec_String_Lines[2]<<endl;
cout<<"LEVEL: " <<vec_String_Lines[3]<<endl;
cout<< "TIME: "<<vec_String_Lines[4]<<endl;

QString ID = QString::fromStdString(vec_String_Lines[0]);
QString STYRBORD = QString::fromStdString(vec_String_Lines[1]);
QString BAGBORD = QString::fromStdString(vec_String_Lines[2]);
QString LEVEL = QString::fromStdString(vec_String_Lines[3]);
QString TIME = QString::fromStdString(vec_String_Lines[4]);
 *one_line_string = '\0';


 save_data tmp;
 tmp.ID_ = ID;
 tmp.STYRBORD_ = STYRBORD;
 tmp.BAGBORD_ = BAGBORD;
 tmp.LEVEL_ = LEVEL;
```

```
        tmp.TIME_ = TIME;

        //save() data
        saveData a;
        a.save(tmp);
        qDebug()<<"Data gemt";

        addLogEntry("Data gemt");

        client->close();

        addLogEntry("Klar til modtagelse af ny data");

    }
```