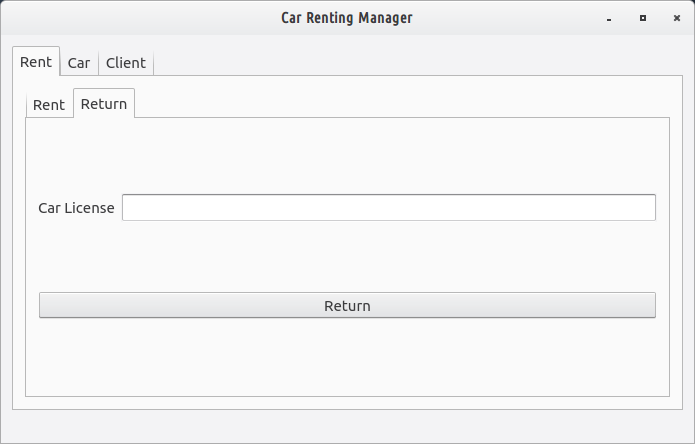
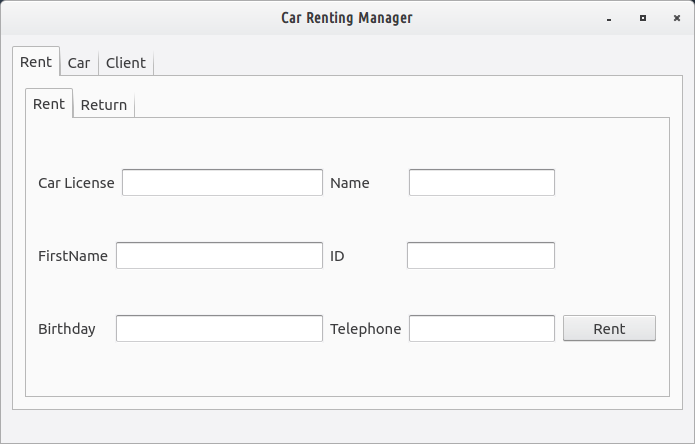
Rapport Projet C++ : Gestionnaire de location de voitures

Elaboré par : Jlidi Borhen IIA3 G1

1-Interface de l'application :

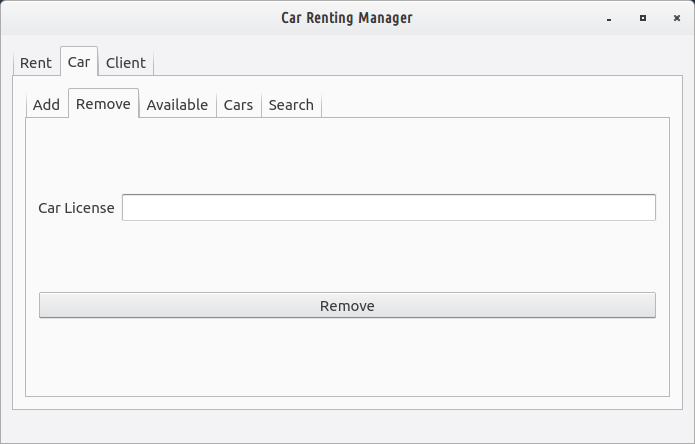
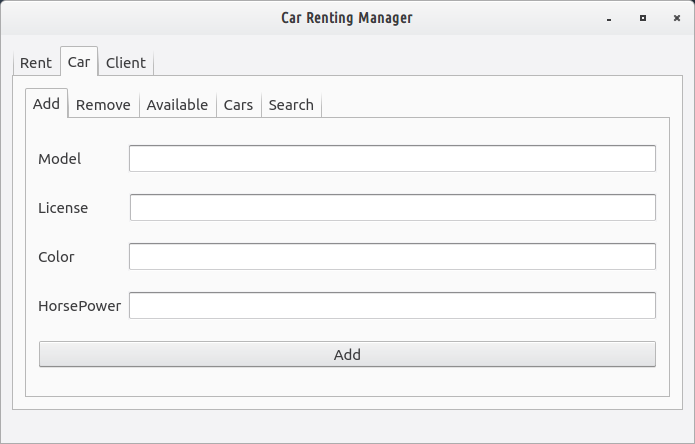
1-1-Location :

\*Louer une voiture\* \*Retourner une voiture\*

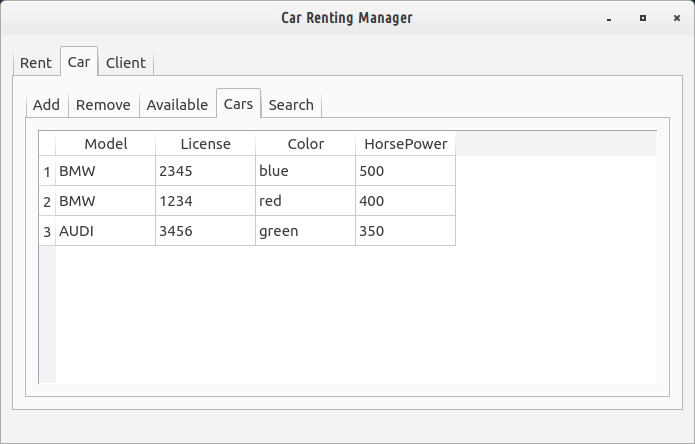
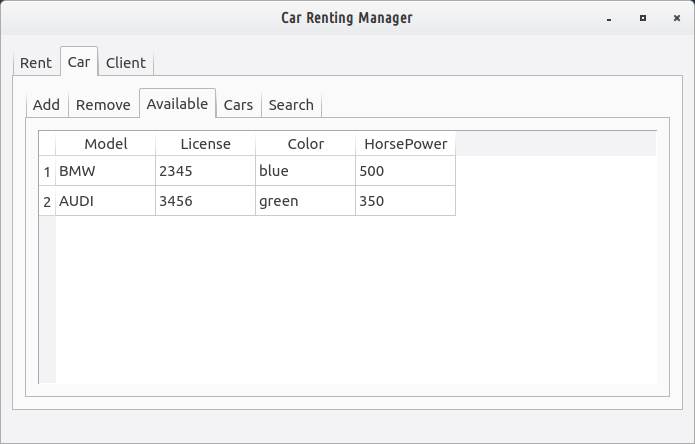


1-2-Voiture :

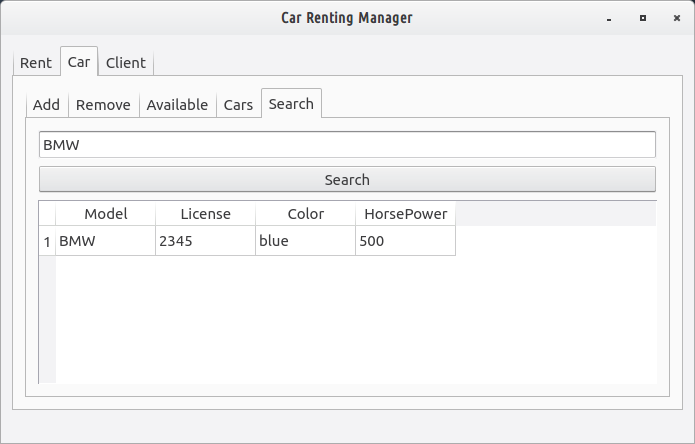
\*Ajouter une voiture\* \*Supprimer une voiture\*



\*Voitures disponibles\* \*Tous les voitures\*



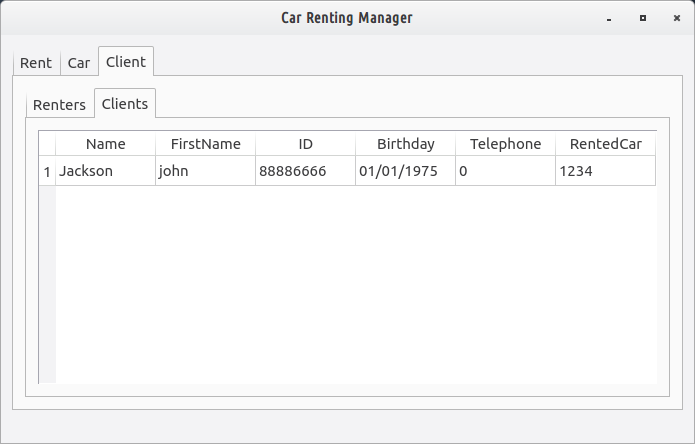
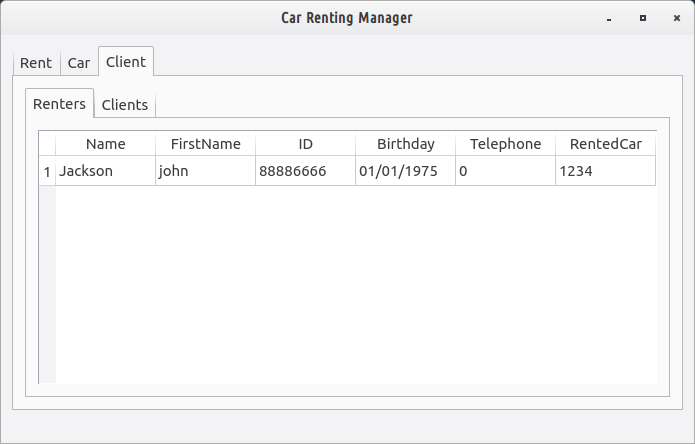
\*Voitures disponibles pour une marque donnée\*



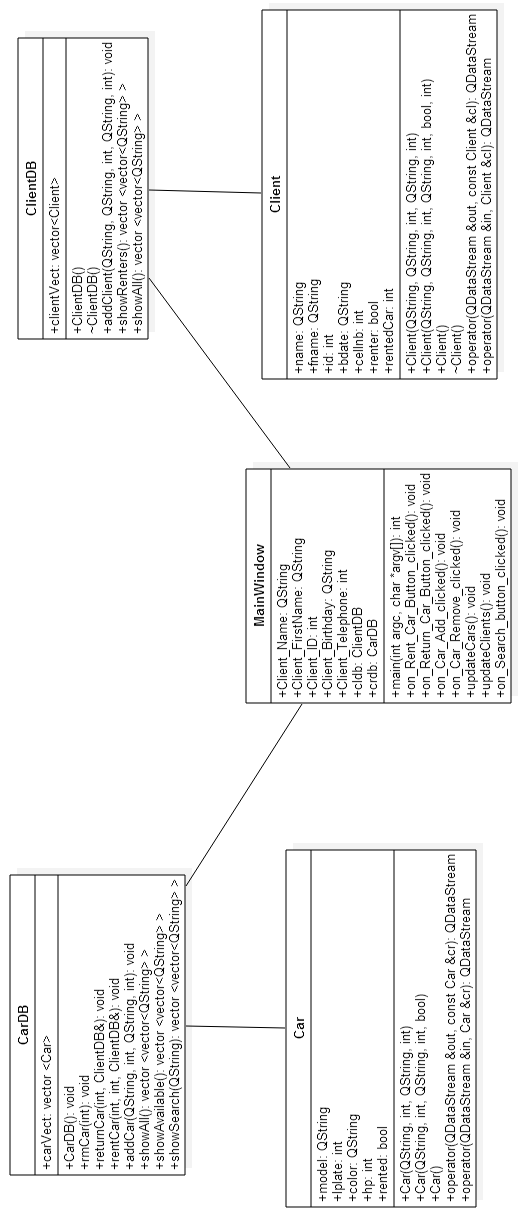
1-3-Client :

\*Les clients qui n'ont pas rendu

leurs voitures\* \*Tous les clients\*



2-Diagramme de classes :



3-Principe(explication brève):

-La classe ClientDB a comme attribut clientVect qui contient tous les clients.

-La classe CarDB a comme attribut carVect qui contient tous les voitures.

-Un objet de type ClientDB contient tous les clients.

-Un objet de type CarDB contient tous les voitures.

-Si une voiture est louée , le client correspondent devient un “renter” (renter=true) et son “rentedCar” recoit la matricule de la voiture louée et la voiture dont la matricule est égale à la matricule de la voiture louéé ,son “rented” devient “true”.

-Si on retourne une voiture , Client : “renter” devient false, Car: "rented” devient false et “renteCar” devient 0

-La classe MainWindow comporte tous les "Handlers" des évenements (les clicks sur les boutons).

-Lors de l'ouverte de l'application , elle charge les données enregistrées dans deux fichiers Clients.backup et Cars.backup .Et lors de la fermeture , tous les objets seront enregistrées dans ces deux fichiers.

4-Source Code :

MainWindow.cpp :

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QtCore>

#include <QtGui>

#include <QtWidgets/QMessageBox>

#include <fstream>

#include <iostream>

#include <QDataStream>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

QFile clientsBackupFile("clients.backup");

QFile carsBackupFile("cars.backup");

clientsBackupFile.open(QIODevice::ReadOnly);

carsBackupFile.open(QIODevice::ReadOnly);

QDataStream clientsBackup(&clientsBackupFile);

QDataStream carsBackup(&carsBackupFile);

while (true) {

if (clientsBackup.atEnd() == true) { break; }

Client cl;

clientsBackup >> cl;

cldb.clientVect.push\_back(cl);

}

while (true) {

if (carsBackup.atEnd() == true) { break; }

Car cr;

carsBackup >> cr;

crdb.carVect.push\_back(cr);

}

clientsBackupFile.close();

carsBackupFile.close();

updateCars();

updateClients();

}

MainWindow::~MainWindow()

{

QFile clientsBackupFile("clients.backup");

QFile carsBackupFile("cars.backup");

clientsBackupFile.open(QIODevice::WriteOnly);

carsBackupFile.open(QIODevice::WriteOnly);

QDataStream clientsBackup(&clientsBackupFile);

QDataStream carsBackup(&carsBackupFile);

for (size\_t i = 0; i < cldb.clientVect.size(); i++) {

clientsBackup << cldb.clientVect[i];

}

for (size\_t i = 0; i < crdb.carVect.size(); i++) {

carsBackup << crdb.carVect[i];

}

clientsBackupFile.close();

carsBackupFile.close();

delete ui;

}

void MainWindow::on\_Rent\_Car\_Button\_clicked()

{

QString Client\_Name = ui->Rent\_Name->text();

QString Client\_FirstName = ui->Rent\_FirstName->text();

int Client\_ID = (ui->Rent\_ID->text()).toInt();

QString Client\_Birthday = ui->Rent\_Birthday->text();

int Client\_Telephone = (ui->Rent\_Telephone->text()).toInt();

cldb.addClient(Client\_Name,Client\_FirstName,Client\_ID,Client\_Birthday,Client\_Telephone);

int Car\_Rent\_License = (ui->Rent\_Car\_License->text()).toInt();

crdb.rentCar(Car\_Rent\_License,Client\_ID,cldb);

updateCars();

updateClients();

}

void MainWindow::on\_Return\_Car\_Button\_clicked()

{

int Car\_Return\_License = (ui->Return\_Car\_License->text()).toInt();

crdb.returnCar(Car\_Return\_License,cldb);

updateCars();

updateClients();

}

void MainWindow::on\_Car\_Add\_clicked()

{

QString Car\_Model = ui->Car\_Add\_Model->text();

int Car\_License = (ui->Car\_Add\_License\_->text()).toInt();

QString Car\_Color = ui->Car\_Color\_Add->text();

int Car\_HP = (ui->Car\_HP\_Add->text()).toInt();

crdb.addCar(Car\_Model,Car\_License,Car\_Color,Car\_HP);

updateCars();

}

void MainWindow::on\_Car\_Remove\_clicked()

{

int Car\_License\_Remove = (ui->Car\_License\_Remove->text()).toInt();

crdb.rmCar(Car\_License\_Remove);

updateCars();

}

void MainWindow::on\_Search\_button\_clicked()

{

QString Searched\_Car\_Name=ui->Search\_field->text();

vector <vector<QString> > listSC =crdb.showSearch(Searched\_Car\_Name);

int rowsSC=listSC.size();

QStandardItemModel \*modelSC = new QStandardItemModel(rowsSC,4,this);

modelSC->setHorizontalHeaderItem(0, new QStandardItem(QString("Model")));

modelSC->setHorizontalHeaderItem(1, new QStandardItem(QString("License")));

modelSC->setHorizontalHeaderItem(2, new QStandardItem(QString("Color")));

modelSC->setHorizontalHeaderItem(3, new QStandardItem(QString("HorsePower")));

for(int i=0;i<listSC.size();i++){

for(int j=0;j<4;j++){

QStandardItem \*dataSC = new QStandardItem(\*&listSC[i][j]);

modelSC->setItem(i,j,dataSC);

}

}

ui->Car\_Search\_Results->setModel(modelSC);

}

void MainWindow::updateCars()

{

vector <vector<QString> > listCA =crdb.showAll();

int rowsCA=listCA.size();

QStandardItemModel \*modelCA = new QStandardItemModel(rowsCA,4,this);

modelCA->setHorizontalHeaderItem(0, new QStandardItem(QString("Model")));

modelCA->setHorizontalHeaderItem(1, new QStandardItem(QString("License")));

modelCA->setHorizontalHeaderItem(2, new QStandardItem(QString("Color")));

modelCA->setHorizontalHeaderItem(3, new QStandardItem(QString("HorsePower")));

for(int i=0;i<listCA.size();i++){

for(int j=0;j<4;j++){

QStandardItem \*dataCA = new QStandardItem(\*&listCA[i][j]);

modelCA->setItem(i,j,dataCA);

}

}

ui->all\_cars->setModel(modelCA);

vector <vector<QString> > listCAV =crdb.showAvailable();

int rowsCAV=listCAV.size();

QStandardItemModel \*modelCAV = new QStandardItemModel(rowsCAV,4,this);

modelCAV->setHorizontalHeaderItem(0, new QStandardItem(QString("Model")));

modelCAV->setHorizontalHeaderItem(1, new QStandardItem(QString("License")));

modelCAV->setHorizontalHeaderItem(2, new QStandardItem(QString("Color")));

modelCAV->setHorizontalHeaderItem(3, new QStandardItem(QString("HorsePower")));

for(int i=0;i<listCAV.size();i++){

for(int j=0;j<4;j++){

QStandardItem \*dataCAV = new QStandardItem(\*&listCAV[i][j]);

modelCAV->setItem(i,j,dataCAV);

}

}

ui->available\_cars->setModel(modelCAV);

}

void MainWindow::updateClients()

{

vector <vector<QString> > listR =cldb.showRenters();

int rowsR=listR.size();

QStandardItemModel \*modelR = new QStandardItemModel(rowsR,6,this);

modelR->setHorizontalHeaderItem(0, new QStandardItem(QString("Name")));

modelR->setHorizontalHeaderItem(1, new QStandardItem(QString("FirstName")));

modelR->setHorizontalHeaderItem(2, new QStandardItem(QString("ID")));

modelR->setHorizontalHeaderItem(3, new QStandardItem(QString("Birthday")));

modelR->setHorizontalHeaderItem(4, new QStandardItem(QString("Telephone")));

modelR->setHorizontalHeaderItem(5, new QStandardItem(QString("RentedCar")));

for(int i=0;i<listR.size();i++){

for(int j=0;j<6;j++){

QStandardItem \*dataR = new QStandardItem(\*&listR[i][j]);

modelR->setItem(i,j,dataR);

}

}

ui->renters\_clients->setModel(modelR);

vector <vector<QString> > listA =cldb.showAll();

int rowsA=listA.size();

QStandardItemModel \*modelA = new QStandardItemModel(rowsA,6,this);

modelA->setHorizontalHeaderItem(0, new QStandardItem(QString("Name")));

modelA->setHorizontalHeaderItem(1, new QStandardItem(QString("FirstName")));

modelA->setHorizontalHeaderItem(2, new QStandardItem(QString("ID")));

modelA->setHorizontalHeaderItem(3, new QStandardItem(QString("Birthday")));

modelA->setHorizontalHeaderItem(4, new QStandardItem(QString("Telephone")));

modelA->setHorizontalHeaderItem(5, new QStandardItem(QString("RentedCar")));

for(int i=0;i<listA.size();i++){

for(int j=0;j<6;j++){

QStandardItem \*dataA = new QStandardItem(\*&listA[i][j]);

modelA->setItem(i,j,dataA);

}

}

ui->all\_clients->setModel(modelA);

}

QString Client\_Name;

QString Client\_FirstName;

int Client\_ID;

QString Client\_Birthday;

int Client\_Telephone;

ClientDB cldb;

CarDB crdb;

int i;

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

ClientDB cldb;

CarDB crdb;

return a.exec();}

MainWindow.h :

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "ClientDB.h"

#include "CarDB.h"

#include "Car.h"

#include "Client.h"

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

friend void ClientDB::addClient(QString, QString, int, QString, int);

QString Client\_Name;

QString Client\_FirstName;

int Client\_ID;

QString Client\_Birthday;

int Client\_Telephone;

ClientDB cldb;

CarDB crdb;

friend int main(int argc, char \*argv[]);

private slots:

void on\_Rent\_Car\_Button\_clicked();

void on\_Return\_Car\_Button\_clicked();

void on\_Car\_Add\_clicked();

void on\_Car\_Remove\_clicked();

void updateCars();

void updateClients();

void on\_Search\_button\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

Client.cpp :

#include "Client.h"

#include <QDataStream>

Client::Client(QString name,QString fname,int id,QString bdate,int cellnb)

{

this->name=name;

this->fname=fname;

this->id=id;

this->bdate=bdate;

this->cellnb=cellnb;

this->renter=false;

this->rentedCar=0;

}

Client::Client(QString name,QString fname,int id,QString bdate,int cellnb, bool renter,int rentedCar)

{

this->name=name;

this->fname=fname;

this->id=id;

this->bdate=bdate;

this->cellnb=cellnb;

this->renter=renter;

this->rentedCar=rentedCar;

}

Client::Client()

{

this->name="";

this->fname="";

this->id=0;

this->bdate="";

this->cellnb=0;

this->renter=false;

this->rentedCar=0;

}

Client::~Client(){}

QDataStream &operator<<(QDataStream &out, const Client &cl)

{

out << cl.name << cl.fname << cl.id << cl.bdate << cl.cellnb << cl.renter << cl.rentedCar;

return out;

}

QDataStream &operator>>(QDataStream &in, Client &cl)

{

QString name;

QString fname;

int id;

QString bdate;

int cellnb;

bool renter;

int rentedCar;

in >> name >> fname >> id >> bdate >> cellnb >> renter >> rentedCar;

cl = Client(name,fname,id,bdate,cellnb,renter,rentedCar);

return in;

}

Client.h :

#ifndef CLIENT\_H

#define CLIENT\_H

#include <QString>

class Client

{

public:

QString name;

QString fname;

int id;

QString bdate;

int cellnb;

bool renter;

int rentedCar;

Client(QString,QString,int,QString,int);

Client(QString,QString,int,QString,int,bool,int);

Client();

~Client();

friend class CarDB;

};

QDataStream &operator<<(QDataStream &out, const Client &cl);

QDataStream &operator>>(QDataStream &in, Client &cl);

#endif // CLIENT\_H

Car.cpp :

#include "Car.h"

#include <QDataStream>

QString model;

int lplate;

QString color;

int hp;

bool rented;

Car::Car(QString model,int lplate,QString color,int hp)

{

this->color=color;

this->hp=hp;

this->lplate=lplate;

this->model=model;

this->rented=false;

}

Car::Car(QString model,int lplate,QString color,int hp,bool rented)

{

this->color=color;

this->hp=hp;

this->lplate=lplate;

this->model=model;

this->rented=rented;

}

Car::Car(){}

QDataStream &operator<<(QDataStream &out, const Car &cr)

{

out << cr.model << cr.lplate << cr.color << cr.hp << cr.rented ;

return out;

}

QDataStream &operator>>(QDataStream &in, Car &cr)

{

QString model;

int lplate;

QString color;

int hp;

bool rented;

in >> model >> lplate >> color >> hp >> rented;

cr = Car(model,lplate,color,hp,rented);

return in;

}

Car.h :

#ifndef CAR\_H

#define CAR\_H

#include <QString>

class Car

{

public:

QString model;

int lplate;

QString color;

int hp;

bool rented;

Car(QString,int,QString,int);

Car(QString,int,QString,int,bool);

Car();

};

QDataStream &operator<<(QDataStream &out, const Car &cr);

QDataStream &operator>>(QDataStream &in, Car &cr);

#endif // CAR\_H

ClientDB.cpp :

#include "ClientDB.h"

ClientDB::ClientDB(){}

ClientDB::~ClientDB(){}

void ClientDB::addClient(QString name,QString fname,int id,QString bdate,int cellnb)

{

bool redundency=false;

for(int i=0;i<clientVect.size();i++){

if (clientVect[i].name==name &&

clientVect[i].fname==fname &&

clientVect[i].bdate==bdate &&

clientVect[i].cellnb==cellnb ){

redundency=true;

break;

}}

if(redundency==false){

Client c= Client(name,fname,id,bdate,cellnb);

this->clientVect.push\_back(c);

}

}

vector <vector<QString> > ClientDB::showRenters(){

using namespace std;

int i;

QString id,cellnb,rentedcar;

vector <vector<QString> > list;

for(i=0;i<clientVect.size();i++){

if (clientVect[i].renter==true){

vector<QString> values;

values.push\_back(clientVect[i].name);

values.push\_back(clientVect[i].fname);

id=QString::number(clientVect[i].id);

values.push\_back(id);

values.push\_back(clientVect[i].bdate);

cellnb=QString::number(clientVect[i].cellnb);

values.push\_back(cellnb);

rentedcar=QString::number(clientVect[i].rentedCar);

values.push\_back(rentedcar);

list.push\_back(values);

}

}

return list;

}

vector <vector<QString> > ClientDB::showAll(){

using namespace std;

int i;

QString id,cellnb,rentedcar;

vector <vector<QString> > list;

for(i=0;i<clientVect.size();i++){

vector<QString> values;

values.push\_back(clientVect[i].name);

values.push\_back(clientVect[i].fname);

id=QString::number(clientVect[i].id);

values.push\_back(id);

values.push\_back(clientVect[i].bdate);

cellnb=QString::number(clientVect[i].cellnb);

values.push\_back(cellnb);

rentedcar=QString::number(clientVect[i].rentedCar);

values.push\_back(rentedcar);

list.push\_back(values);

}

return list;

}

ClientDB.h :

#ifndef CLIENTDB\_H

#define CLIENTDB\_H

#include "Client.h"

#include <vector>

#include <QString>

using namespace std;

class ClientDB

{

public:

vector<Client> clientVect;

void addClient(QString,QString,int,QString,int);

vector <vector<QString> >showRenters();

vector <vector<QString> >showAll();

ClientDB();

~ClientDB();

};

#endif // CLIENTDB\_H

CarDB.cpp :

#include "CarDB.h"

#include "ClientDB.h"

CarDB::CarDB(){}

void CarDB::addCar(QString model,int lplate,QString color,int hp)

{

Car c= Car(model,lplate,color,hp);

carVect.push\_back(c);

}

void CarDB::rmCar(int lplate)

{ int i;

for (i=0;i<this->carVect.size();i++){

if (carVect[i].lplate==lplate) break;

}

carVect.erase(carVect.begin()+i);

}

void CarDB::rentCar(int lpp,int idd,ClientDB &cldb){

int i=0;

for ( i=0;i<cldb.clientVect.size();i++){

if (cldb.clientVect[i].id==idd)break;

}

cldb.clientVect[i].renter=true;

cldb.clientVect[i].rentedCar=lpp;

for (i=0;i<carVect.size();i++){

if (carVect[i].lplate==lpp) break;

}

carVect[i].rented=true;

}

void CarDB::returnCar(int lpp,ClientDB& cldb){

int i;

for (i=0;i<carVect.size();i++){

if (carVect[i].lplate==lpp) break;

}

carVect[i].rented=false;

for (i=0;i<cldb.clientVect.size();i++){

if (cldb.clientVect[i].rentedCar==lpp) break;

}

cldb.clientVect[i].rentedCar=0;

cldb.clientVect[i].renter=false;

}

vector <vector<QString> > CarDB::showAll(){

using namespace std;

int i;

QString lplate,hp;

vector <vector<QString> > list;

for(i=0;i<carVect.size();i++){

vector<QString> values;

values.push\_back(carVect[i].model);

lplate=QString::number(carVect[i].lplate);

values.push\_back(lplate);

values.push\_back(carVect[i].color);

hp=QString::number(carVect[i].hp);

values.push\_back(hp);

list.push\_back(values);

}

return list;

}

vector <vector<QString> > CarDB::showAvailable(){

using namespace std;

int i;

QString lplate,hp;

vector <vector<QString> > list;

for(i=0;i<carVect.size();i++){

if(carVect[i].rented==false){

vector<QString> values;

values.push\_back(carVect[i].model);

lplate=QString::number(carVect[i].lplate);

values.push\_back(lplate);

values.push\_back(carVect[i].color);

hp=QString::number(carVect[i].hp);

values.push\_back(hp);

list.push\_back(values);

}

}

return list;

}

vector <vector<QString> > CarDB::showSearch(QString model){

using namespace std;

int i;

QString lplate,hp;

vector <vector<QString> > list;

for(i=0;i<carVect.size();i++){

if(carVect[i].model==model && carVect[i].rented==false ){

vector<QString> values;

values.push\_back(carVect[i].model);

lplate=QString::number(carVect[i].lplate);

values.push\_back(lplate);

values.push\_back(carVect[i].color);

hp=QString::number(carVect[i].hp);

values.push\_back(hp);

list.push\_back(values);

}

}

return list;

}

CarDB.h :

#ifndef CARDB\_H

#define CARDB\_H

#include "Car.h"

#include "ClientDB.h"

#include <vector>

using namespace std;

class CarDB

{

public:

vector <Car> carVect;

vector <vector<QString> > showAll();

vector <vector<QString> > showAvailable();

vector <vector<QString> > showSearch(QString);

void addCar(QString,int,QString,int);

void rentCar(int,int,ClientDB&);

void returnCar(int,ClientDB&);

void rmCar(int);

CarDB();

};

#endif // CARDB\_H