Alexandru Majeru

CSC330

Project #1

**Vehicle Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include <string>

#include "User.h"

using namespace std;

#ifndef VEHICLE\_H

#define VEHICLE\_H

class Vehicle

{

public:

Vehicle(string m, string p, User d, double i) : model(m), plate(p), Driver(d), mileage(i) {}

//Getters

virtual string getModel(){ return model; }

virtual User getDriver(){ return Driver; }

virtual string getPlate(){ return plate; }

virtual double getMileage(){ return mileage; }

//Setters

virtual void setModel(string m){ model = m; }

virtual void setDriver(const User &d){ Driver = d; }

virtual void setPlate(string i){ plate = i; }

virtual void setMileage(double m){ mileage = m; }

virtual void print() = 0; //pure abstract function - polymorphism

private:

string model;

User Driver; //composition

string plate;

double mileage;

char type;

};

#endif

**Cargo Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include <string>

#include "Vehicle.h"

using namespace std;

#ifndef CARGO\_H

#define CARGO\_H

class Cargo : public Vehicle //public inheritance

{

public:

Cargo(string, string, User, double, double, string);

//Getters based on polymorphism

virtual string getModel(){ return Vehicle::getModel(); }

virtual User getDriver(){ return Vehicle::getDriver(); }

virtual string getPlate(){ return Vehicle::getPlate(); }

virtual double getMileage(){ return Vehicle::getMileage(); }

double getWeight(){ return weight; }

string getCargo(){ return cargo; }

//Setters based on polymorphism

virtual void setModel(string m){ Vehicle::setModel(m); }

virtual void setDriver(User d){ Vehicle::setDriver(d); }

virtual void setID(string i){ Vehicle::setPlate(i); }

virtual void setMileage(double m){ Vehicle::setMileage(m); }

void setWeight(double w){ weight = w; }

void setCarg(string c){ cargo = c; }

virtual void print();

private:

double weight;

string cargo;

char type = 'c';

};

#endif

**Cargo Member Function Definitions:**

//CSC330 Project #1

//Alexandru Majeru

#include "Cargo.h"

#include "User.h"

#include <Windows.h>

#include <iostream>

#include <string>

using namespace std;

Cargo::Cargo(string m, string p, User d, double i, double w, string c)

: Vehicle(m, p, d, i)

{

weight = w;

cargo = c;

}

void Cargo::print()

{

User temp = getDriver();

temp.print();

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << endl << endl << setw(30) << "\t| Vehicle Model | " << setw(18) << "License Plate | " << setw(10) << "Miles |" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(30) << getModel() << setw(18) << getPlate() << setw(10) << getMileage() << endl << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << setw(35) << "\t\t|\t Additional Data \t|" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(20) << getWeight() << "kg\t Cargo: " << getCargo() << endl << endl;

}

**Business File Code:**

//CSC330 Project #1

//Alexandru Majeru

#ifndef BUSINESS\_H

#define BUSINESS\_H

#include "Vehicle.h"

#include "User.h"

#include <string>

using namespace std;

class Business : public Vehicle //public inheritance

{

public:

Business(string, string, User, double, int, string);

//Getters based on polymorphism

virtual string getModel(){ return Vehicle::getModel(); }

virtual User getDriver(){ return Vehicle::getDriver(); }

virtual string getPlate(){ return Vehicle::getPlate(); }

virtual double getMileage(){ return Vehicle::getMileage(); }

int getPN(){ return passNum; }

string getPassengers() { return passengers; }

//Setters based on polymorphism

virtual void setModel(string m){ Vehicle::setModel(m); }

virtual void setDriver(const User &d){ Vehicle::setDriver(d); }

virtual void setID(string i){ Vehicle::setPlate(i); }

virtual void setMileage(double m){ Vehicle::setMileage(m); }

void setPN(int num){ passNum = num; }

void setPassenger(string p){ passengers = p; }

virtual void print();

private:

int passNum;

string passengers;

char type = 'b';

};

#endif

**Business Member Function Definitions:**

//CSC330 Project #1

//Alexandru Majeru

#include "Business.h"

#include "User.h"

#include <Windows.h>

#include <iostream>

#include <string>

using namespace std;

Business::Business(string m, string p, User d, double i, int pn, string pass)

: Vehicle(m, p, d, i)

{

passNum = pn;

passengers = pass;

}

void Business::print()

{

User temp = getDriver();

temp.print();

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << endl << endl << setw(30) << "\t| Vehicle Model | " << setw(18) << "License Plate | " << setw(10) << "Miles |" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(30) << getModel() << setw(18) << getPlate() << setw(10) << getMileage() << endl << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << setw(35) << "\t\t|\t Additional Data \t|" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(20) << getPN() << " passenger(s): " << getPassengers() << endl << endl << endl;

}

**Loan Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include "Vehicle.h"

#include <string>

using namespace std;

#ifndef LOAN\_H

#define LOAN\_H

class Loan : public Vehicle //public inheritance

{

public:

Loan(string, string, User, double, string, string);

//Getters based on polymorphism

virtual string getModel(){ return Vehicle::getModel(); }

virtual User getDriver(){ return Vehicle::getDriver(); }

virtual string getPlate(){ return Vehicle::getPlate(); }

virtual double getMileage(){ return Vehicle::getMileage(); }

string getIC(){ return insuranceCompany; }

string getIP(){ return insurancePolicy; }

//Setters based on polymorphism

virtual void setModel(string m){ Vehicle::setModel(m); }

virtual void setDriver(const User &d){ Vehicle::setDriver(d); }

virtual void setID(string i){ Vehicle::setPlate(i); }

virtual void setMileage(double m){ Vehicle::setMileage(m); }

void setIC(string c){ insuranceCompany = c; }

void setIP(string p){ insurancePolicy = p; }

virtual void print();

private:

string insuranceCompany;

string insurancePolicy;

char type = 'l';

};

#endif

**Loan Member Function Definitions:**

//CSC330 Project #1

//Alexandru Majeru

#include "Loan.h"

#include "User.h"

#include <Windows.h>

#include <iostream>

#include <string>

using namespace std;

Loan::Loan(string m, string p, User d, double i, string IC, string IP)

: Vehicle(m, p, d, i)

{

insuranceCompany = IC;

insurancePolicy = IP;

}

void Loan::print()

{

User temp = getDriver();

temp.print();

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << endl << endl << setw(30) << "\t| Vehicle Model |" << setw(18) << "License Plate |" << setw(10) << "Miles |" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(30) << getModel() << setw(18) << getPlate() << setw(10) << getMileage() << endl << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << setw(35) << "\t\t|\t Additional Data \t|" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

cout << setw(35) << getIC() << setw(7) << "PN: " << getIP() << endl << endl << endl;

}

**User Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include <string>

#include <iostream>

#include <iomanip>

using namespace std;

#ifndef USER\_H

#define USER\_H

class User

{

public:

User(string i, string f, string l) : ID(i), fName(f), lName(l) {}

//Getters

string getFirst() const { return fName; }

string getLast() const { return lName; }

string getID() const { return ID; }

//Setters

void setFirst(string f) { fName = f; }

void setLast(string l) { lName = l; }

void setID(string i) { ID = i; }

bool operator==(string u){ return this->ID == u; }

void print() { cout <<setw(8)<< ID << setw(20) << fName << setw(20) << lName; }

private:

string fName;

string lName;

string ID;

};

#endif

**Order Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include "User.h"

#include "Loan.h"

#include "Business.h"

#include "Cargo.h"

#include "Vehicle.h"

#include <time.h>

#include <string>

using namespace std;

#ifndef ORDER\_H

#define ORDER\_H

class Order

{

public:

Order(string, string, char, Vehicle\*);

char getType() { return type; }

Vehicle\* getCar() { return car; }

void setTimeStart(string t){ timeS = t; }

void setTimeEnd(string t){ timeE = t; }

string getTimeStart() const { return timeS; }

string getTimeEnd() const { return timeE; }

bool operator<(const Order &o);

void print();

private:

string timeS;

string timeE;

char type;

Vehicle \*car;

};

#endif

**Order Member Function Definitions:**

//CSC330 Project #1

//Alexandru Majeru

#include "Order.h"

#include "User.h"

#include "Loan.h"

#include "Business.h"

#include "Cargo.h"

#include "Vehicle.h"

#include <Windows.h>

#include <time.h>

#include <string>

#include <iostream>

#include <iomanip>

using namespace std;

Order::Order(string timeS, string timeE, char t, Vehicle \*c)

{

this->timeS = timeS;

this->timeE = timeE;

type = t;

car = c;

}

void Order::print()

{

for (int i = 0; i < 80; i++)

cout << "\_";

cout << endl << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 2);

cout << "Order Start Time: " << timeS << "\n"

<< "Order End Time: " << timeE << "\n\n";

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << setw(20) << "V. Type | " << setw(10) << "Empl. ID |" << setw(20) << "First Name |" << setw(20) << "Last Name | " << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

if (type == 'c')

cout << setw(17) << "Cargo";

else if (type == 'b')

cout << setw(17) << "Business";

else if (type == 'l')

cout << setw(17) << "Loan";

car->print();

}

bool Order::operator<(const Order &o)

{

return timeS < o.getTimeStart();

}

**Menu Header File Code:**

//CSC330 Project #1

//Alexandru Majeru

#include "Order.h"

#include <list>

#include <string>

using namespace std;

#ifndef MENU\_H

#define MENU\_H

class Menu

{

public:

Menu() : currUser("i", "f", "l") {}

int welcome();

void getEmployees();

void getHistory();

int options();

string chooseType(int);

void displayEmployees();

User addEmployee();

void displayHistory(string);

void placeOrder(string);

string properDate(string date);

bool isProperDate(string date);

private:

list<Order> History; //composition, STL

list<User> Employees; //composition, STL

User currUser; //composition

};

#endif

**Menu Member Function Definitions:**

//CSC330 Project #1

//Alexandru Majeru

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Menu.h"

#include <windows.h>

#include <iostream>

#include <iomanip>

#include <sstream>

#include <fstream>

#include <time.h>

#include <string>

#include "User.h"

using namespace std;

int Menu::welcome() //Welcome method - reads the data

{

getEmployees(); //from external files, saves it

getHistory(); //runs the search for the employee

cout << "Welcome to the Vehicle Management System!" << endl;

currUser = addEmployee();

system("CLS");

cout << "Welcome, " << currUser.getFirst() << " " << currUser.getLast() << endl;

return 0;

}

void Menu::getEmployees() //gets Employee file to write

{ //into <list> of Employees

ifstream readit;

readit.open("Employees.txt");

if (!readit) //if cannot open file

{ //create new one

ofstream write("Employees.txt");

getEmployees(); //recursion

write.close();

}

else

while (readit.good()) //if open file, parsing

{ //for employee ID and Name

string i, f, l;

getline(readit, i, '|');

getline(readit, f, '|');

getline(readit, l);

User temp(i, f, l);

Employees.push\_back(temp); //save employee into list

}

readit.close(); //close file

}

void Menu::getHistory()

{

ifstream readit;

readit.open("History.txt");

if (!readit) //if cannot open file

{ //create new one

ofstream write("History.txt");

getHistory(); //recursion

}

else

while (readit.good())

{

string tS,tE, c, m, p, i, f, l; //parsing the file

double mile;

getline(readit, tS, '|');

getline(readit, tE, '|');

getline(readit, c, '|');

getline(readit, m, '|');

getline(readit, p, '|');

getline(readit, i, '|');

getline(readit, f, '|');

getline(readit, l, '|');

if (c == "Cargo") //depending on vehicle type

{ //read different data

double weight;

string cargo;

readit >> mile >> weight;

getline(readit, cargo);

User temp(i, f, l);

Vehicle \*car = new Cargo(m, p, temp, mile, weight, cargo);

tE = properDate(tE);

Order o(tS, tE, 'c', car);

History.push\_back(o); //push data into list

}

else if (c == "Business")

{

string passengers;

int passNum;

readit >> mile >> passNum;

getline(readit, passengers);

User temp(i, f, l);

Vehicle \*car = new Business(m, p, temp, mile, passNum, passengers);

tE = properDate(tE);

Order o(tS, tE, 'b', car);

History.push\_back(o);

}

else if (c == "Loan")

{

string IC, IP;

readit >> mile;

getline(readit, IC, '|');

getline(readit, IP);

User temp(i, f, l);

Vehicle \*car = new Loan(m, p, temp, mile, IC, IP);

tE = properDate(tE);

Order o(tS, tE, 'l', car);

History.push\_back(o);

}

}

readit.close();

}

int Menu::options() //display possible options to the user

{

cout << "Please, choose one of the options below: " << endl;

cout << "1. Add Vehicle Use Case \n2. Employees \n3. History \n4. Exit\n";

string i;

cin >> i;

if (i == "1") //add vehicle to history

{

system("CLS");

cout << "Please, select the type of vehicle: \n";

string choice = chooseType(1);

system("CLS");

if (choice == "4")

return 1;

placeOrder(choice);

return 1;

}

else if (i == "2") //display employees

{

system("CLS");

displayEmployees();

return 1;

}

else if (i == "3") //display history

{

system("CLS");

cout << "Please, select the type of history: \n";

string choice = chooseType(2);

system("CLS");

if (choice == "5")

return 1;

displayHistory(choice);

return 1;

}

else if (i == "4") //exit program

{

return 0;

}

else

{

system("CLS");

cout << "Invalid option!\n";

i = "";

return options();

}

return 0;

}

string Menu::chooseType(int i) //display a type of menu

{ //either for adding data to history

string c; //or for displaying history

if (i == 1)

{

cout << "1. Cargo Transportation \n2. Business Trips \n3. Personal Use \n4. Back\n";

cin >> c;

//loop until user inputs proper value

while (c != "1" && c != "2" && c != "3" && c != "4")

{

system("CLS");

cout << "Invalid option!\n";

c = chooseType(1);

}

return c;

}

else if (i == 2)

{

cout << "1. Cargo Transportation \n2. Business Trips \n3. Personal Use \n4. View All History\n5. Back\n";

cin >> c;

while (c != "1" && c != "2" && c != "3" && c != "4" && c != "5")

{

system("CLS");

cout << "Invalid option!\n";

c = chooseType(2);

}

return c;

}

return 0;

}

void Menu::displayEmployees() //using an iterator, display contents of Employee list

{

list<User>::iterator it; //STL iterator

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 3);

cout << setw(8) << "ID" << setw(20) << "First Name" << setw(20) << "Last Name" << endl;

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 7);

for (it = Employees.begin(); it != Employees.end(); it++)

{

it->print();

cout << endl;

}

system("Pause");

system("CLS");

}

void Menu::displayHistory(string i) //using interator, display contents of History list

{

list<Order>::iterator it; //STL iterator

char c;

if (i == "1")

c = 'c';

else if (i == "2")

c = 'b';

else if (i == "3")

c = 'l';

History.sort();

if (History.size() == 0)

cout << "There is no order history present!\n";

else if (i == "4")

{

for (it = History.begin(); it != History.end(); it++)

it->print();

}

else //print specific vehicle type history

{

int count = 0;

for (it = History.begin(); it != History.end(); it++)

{

if (it->getType() == c)

{

it->print();

count++;

}

}

if (count == 0)

cout << "No orders of this type present!" << endl;

}

system("Pause");

system("CLS");

}

void Menu::placeOrder(string c) //create a new order

{ //ask user to input data

ofstream write;

string m, p, ID, f, l, tStart;

double miles;

time\_t clock = time(0); //get current time

string tEnd = ctime(&clock); //change to string

tEnd.pop\_back();

tEnd.erase(16, 3);

write.open("History.txt", ofstream::app); //open file to write into

if (!write)

{

cout << "An error has occured, please contact support.";

system("Pause");

throw logic\_error("There was a problem reading the file!");

}

else

{

cin.ignore(); //ignore necessary for getline accuracy

cin.sync();

cout << "Please, input the starting date and time of the trip\n(YYYY-MM-DD HH:MM(24hr)): \n";

getline(cin, tStart);

while (!isProperDate(tStart))

getline(cin, tStart);

cout << "Please, input the model of the vehicle: \n";

getline(cin, m);

cout << "Please, input the license plate of the vehicle: \n";

getline(cin, p);

cout << "Please, input the mileage of the trip: \n";

cin >> miles;

while (!cin)

{

cin.clear();

cin.ignore();

cout << "The mileage value must be a number!\n Please, input the mileage of the trip: \n";

cin >> miles;

}

cin.ignore();

cin.sync();

User tempUser = addEmployee(); //check if Employee already in system

if (c == "1")

{

double w;

string cargo;

cout << "Please, input the weight of the cargo in kg: \n";

cin >> w;

while (!cin)

{

cin.clear();

cin.ignore();

cout << "The value must be a number!\n Please, input the weight of the cargo in kg: \n";

cin >> w;

}

cin.clear();

cin.ignore();

cout << "Please, input the cargo list, separating items with a comma (','): \n";

getline(cin, cargo);

Vehicle \*car = new Cargo(m, p, tempUser, miles, w, cargo);

write << tStart << "|" << tEnd << "|" << "Cargo" << "|" << m << "|" << p << "|" << tempUser.getID()

<< "|" << tempUser.getFirst() << "|" << tempUser.getLast() << "|"

<< miles << " " << w << " " << cargo << endl;

tEnd = properDate(tEnd);

Order temp(tStart, tEnd, 'c', car);

History.push\_back(temp); //push order into list and print into file

}

else if (c == "2")

{

int pN;

string passengers;

cout << "Please, input the number of passengers on the trip: \n";

cin >> pN;

while (!cin)

{

cin.clear();

cin.ignore();

cout << "The value must be a number!\n Please, input the number of passengers on the trip: \n";

cin >> pN;

}

cin.clear();

cin.ignore();

cin.ignore();

cin.sync();

if (pN == 0)

{

passengers = "None";

}

else

{

cout << "Please, input the list of passengers, separating items with a comma (','): \n";

getline(cin, passengers);

}

Vehicle \*car = new Business(m, p, tempUser, miles, pN, passengers);

write << tStart << "|" << tEnd << "|" << "Business" << "|" << m << "|" << p << "|" << tempUser.getID()

<< "|" << tempUser.getFirst() << "|" << tempUser.getLast() << "|"

<< miles << " " << pN << " " << passengers << endl;

tEnd = properDate(tEnd);

Order temp(tStart, tEnd, 'b', car);

History.push\_back(temp); //push order into list and print into file

}

else if (c == "3")

{

string IC, IP;

cout << "Please, input the name of insurance company: \n";

getline(cin, IC);

cout << "Please, input the insurance policy: \n";

getline(cin, IP);

Vehicle \*car = new Loan(m, p, tempUser, miles, IC, IP);

write << tStart << "|" << tEnd << "|" << "Loan" << "|" << m << "|" << p << "|" << tempUser.getID()

<< "|" << tempUser.getFirst() << "|" << tempUser.getLast() << "|"

<< miles << " " << IC << "|" << IP << endl;

tEnd = properDate(tEnd);

Order temp(tStart, tEnd, 'l', car);

History.push\_back(temp);

}

system("CLS");

cout << "Case was added to history!\n";

}

write.close();

}

User Menu::addEmployee() //checks if Employee ID

{ //is already in the file

list<User>::iterator it;

string ID, f, l;

cout << "Please, enter Employee's ID: \n";

getline(cin, ID);

it = find(Employees.begin(), Employees.end(), ID); //search for employee

if (it != Employees.end()) //in Employee list

{

return \*it; //if found, return

}

else //if not found, add

{

ofstream write;

write.open("Employees.txt", ofstream::app);

if (!write)

{

cout << "An error has occured, please contact support.";

system("Pause");

throw logic\_error("There was a problem reading the file!");

}

else

{

cout << "Please, enter First Name: \n";

getline(cin, f);

for (int i = 0; i<f.length();++i) //check for numbers in name

if (isdigit(f[i]))

{

cout << "Invalid name. No numbers allowed" << endl;

cout << "Please, enter First Name: \n";

getline(cin, f);

i = 0;

}

cout << "Please, enter Last Name: \n";

getline(cin, l);

for (int i = 0; i<l.length(); ++i) //check for numbers in name

if (isdigit(l[i]))

{

cout << "Invalid name. No numbers allowed" << endl;

cout << "Please, enter First Name: \n";

getline(cin, l);

i = 0;

}

for (int i = 0; i<l.length(); ++i)

if (isdigit(l[i]))

{

cout << "Invalid name. No numbers allowed" << endl;

system("Pause");

exit(EXIT\_FAILURE);

}

User temp(ID, f, l);

Employees.push\_back(temp);

write << ID << "|" << f << "|" << l << endl; //write into file

write.close();

return temp;

}

}

}

string Menu::properDate(string date)

{

string s, y, m, d, h, min;

if (date.length() <= 1)

return "";

y = date.substr(17, 4);

y.append("-");

m = date.substr(4, 3);

d = date.substr(8, 2);

d.append(" ");

h = date.substr(11, 2);

h.append(":");

min = date.substr(14, 2);

min.append(" ");

if (m == "Jan")

m = "01-";

else if (m == "Feb")

m = "02-";

else if (m == "Mar")

m = "03-";

else if (m == "Apr")

m = "04-";

else if (m == "May")

m = "05-";

else if (m == "Jun")

m = "06-";

else if (m == "Jul")

m = "07-";

else if (m == "Aug")

m = "08-";

else if (m == "Sep")

m = "09-";

else if (m == "Oct")

m = "10-";

else if (m == "Nov")

m = "11-";

else if (m == "Dec")

m = "12-";

s.append(y);

s.append(m);

s.append(d);

s.append(h);

s.append(min);

return s;

}

bool Menu::isProperDate(string date)

{

string s, y, m, d, h, min, a;

if (date.length() != 16)

{

cout << "Invalid date or time! Check the format:\nYYYY-MM-DD HH:MM (24hr)\n";

return false;

}

y = date.substr(0, 4);

m = date.substr(5, 2);

d = date.substr(8, 2);

h = date.substr(11, 2);

min = date.substr(14, 2);

if (y > "2018" && y<"2000" && m>"12" && m < "1" && d<"1" && d>"12" && h<"00" && h>"24" && m<"00" && m>"59")

{

cout << "Invalid date or time! Check the format:\nYYYY-MM-DD HH:MM (24hr)\n";

return false;

}

time\_t clock = time(0); //get current time

string tEnd = ctime(&clock); //change to string

tEnd.pop\_back();

tEnd.erase(16, 3);

tEnd = properDate(tEnd);

if (date > tEnd)

{

cout << "Invalid date or time! Check the format:\nYYYY-MM-DD HH:MM (24hr)\n";

return false;

}

return true;

}

**Driver Code:**

//CSC330 Lab #3

//Alexandru Majeru

#define \_CRT\_SECURE\_NO\_WARNINGS

#define \_WIN32\_WINNT 0x500

#include <Windows.h>

#include <fstream>

#include <iostream>

#include <string>

#include <list>

#include <algorithm>

#include "time.h"

#include "Vehicle.h"

#include "User.h"

#include "Business.h"

#include "Cargo.h"

#include "Loan.h"

#include "Order.h"

#include "Menu.h"

using namespace std;

int main()

{

Menu VMS;

int i = 1;

//this code sets the size of the screen for the user

HWND console = GetConsoleWindow();

RECT r;

GetWindowRect(console, &r);

MoveWindow(console, r.left, r.top, 800, 600, TRUE);

VMS.welcome();

while (i == 1) //run until user decides the exit the program

{

i = VMS.options();

}

return 0;

}