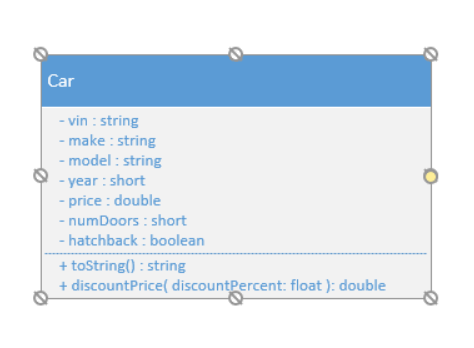
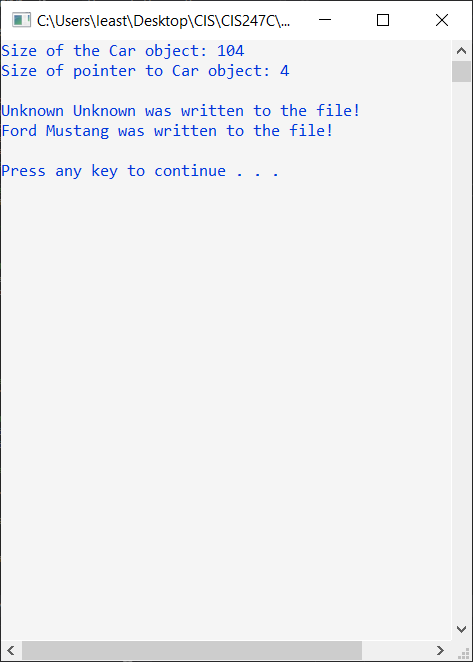
**UML DIAGRAM:**

****

**SCREENSHOT:**

****

**SOURCE:**

/\*

Leah Rieger

11/04/2019

CIS247C Course Project

\*/

#include <iostream>

#include <string>

#include <iomanip> // setw, setprecision

#include <fstream> // required to read/write to files

#include "Car.h";

using namespace std;

//prototypes

void saveToFile(Car\* ptrCar);

/// Entry point for the application

int main()

{

// create 2 Car obj -- use default & parameterized constructors

Car firstCar;

Car secondCar("X2345678901234567", "Ford", "Mustang", 2010, 8500.0, 2, false);

//check the size

cout << "Size of the Car object: " << sizeof(firstCar) << endl;

cout << "Size of pointer to Car object: " << sizeof(&firstCar) << endl;

cout << endl; //blank line

//delete file if it exists

remove("data.txt");

//call the method to write your Car to the file

saveToFile( &firstCar ); //notice the & meaning 'address of'

saveToFile( &secondCar );

// pause

cout << endl;

system("pause");

return 0;

}

void saveToFile(Car\* ptrCar)

{

// open the pipe to the file

ofstream outToFile("data.txt");

// if the file is open, write data to file

if (outToFile.is\_open())

{

//notice that we DEREFERENCE the Car pointer using (->)

outToFile << ptrCar->getVin() << endl;

outToFile << ptrCar->getMake() << endl;

outToFile << ptrCar->getModel() << endl;

outToFile << ptrCar->getYear() << endl;

outToFile << ptrCar->getPrice() << endl;

outToFile << ptrCar->getNumDoors() << endl;

outToFile << ptrCar->getHatchback() << endl;

//close the pipe

outToFile.close();

//tell user that the data was written

cout << ptrCar->getMake() << " " << ptrCar->getModel() << " was written to the file!" << endl;

}

}

**CAR.CPP**

#include "Car.h"

// constructor

Car::Car()

{

vin = "Unknown";

make = "Unknown";

model = "Unknown";

year = 0;

price = 0.0;

numDoors = 0;

hatchback = false;

}

/// Setting all of the car information

Car::Car(string vin, string make, string model, short year, double price,

short numDoors, bool hatchback)

{

setVin(vin);

setMake(make);

setModel(model);

setYear(year);

setPrice(price);

setNumDoors(numDoors);

setHatchback(hatchback);

}

// destructor

Car::~Car()

{

}

// returning all information on car

string Car::toString()

{

return "Vin: " + vin + ", Make: " + make + ", Model: " + model + ", Year: "

+ to\_string(year) + ", Price: " + to\_string(price) + ", Number of Doors: "

+ to\_string(numDoors) + ", Hatchback: " + to\_string(hatchback);

}

// calculating the price of the car with a discount

double Car::discountPrice(float discountPercent)

{

double newPrice = price - (price \* discountPercent);

return newPrice;

}

// vin number of the car

string Car::getVin()

{

return vin;

}

void Car::setVin(string vin)

{

if (vin.length() == 17)

{

this->vin = vin;

}

else

{

this->vin = "Unknown";

}

}

// make of the car

string Car::getMake()

{

return make;

}

void Car::setMake(string make)

{

if (make.length() > 0)

{

this->make = make;

}

else

{

this->make = "Unknown";

}

}

// model of the car

string Car::getModel()

{

return model;

}

void Car::setModel(string model)

{

if (model.length() > 0)

{

this->model = model;

}

else

{

this->model = "Unknown";

}

}

// year of the car

short Car::getYear()

{

return year;

}

void Car::setYear(short year)

{

if (year > 1980)

{

this->year = year;

}

else

{

this->year = 0;

}

}

// price of the car

double Car::getPrice()

{

return price;

}

void Car::setPrice(double price)

{

if (price > 0.0 && price < 1000000.0)

{

this->price = price;

}

else

{

this->price = 0.0;

}

}

// the number of doors on the car

short Car::getNumDoors()

{

return numDoors;

}

void Car::setNumDoors(short numDoors)

{

if (numDoors > 0 && numDoors < 5)

{

this->numDoors = numDoors;

}

else

{

this->numDoors = 0;

}

}

//Does the car have hatchback

bool Car::getHatchback()

{

return hatchback;

}

void Car::setHatchback(bool isHatchback)

{

this->hatchback = isHatchback;

}

**CAR.H**

#pragma once

#include <string>

using namespace std;

class Car

{

private:

//attributes

string vin;

string make;

string model;

short year;

double price;

short numDoors;

bool hatchback;

public:

//constructors & destructors

Car();

Car(string vin, string make, string model, short year, double price,

short numDoors, bool hatchback);

~Car();

//behaviors in the public section!

string toString();

double discountPrice(float discountPercent);

// accessors & mutators

string getVin();

void setVin(string vin);

string getMake();

void setMake(string make);

string getModel();

void setModel(string mmodel);

short getYear();

void setYear(short year);

double getPrice();

void setPrice(double price);

short getNumDoors();

void setNumDoors(short numDoors);

bool getHatchback();

void setHatchback(bool isHatchback);

};