William George

CS 161 Week 9

Assignment 9

UNDERSTANDING

This week is all about classes. As it was mentioned in the book, classes are a means of hiding the inner workings of a program. Using classes makes a program seem a lot less complicated than it actually is because many of the functions being taken care of in main can be controlled by the classes. While the car lot program this week is similar to the one we did last week, it is very different because we are using classes rather than a struct. The classes give us a place where we can do all of the needed input validation and calculations away from the parent program with the program only acting on an object of the class.

I believe the emphasis placed on breaking big problems down into their smaller components throughout this course was done some with the goal of finally arriving to this week where we truly get to use C++ as an object oriented language. While the first phase of the course was tedious, now that we’ve arrived at this point I understand the reasoning behind the structure of the course. Classes are immensely valuable in breaking big problems down into their individual problems.

The project this week, while similar to that of last week, does have some key differences. It will be made up of three classes; Date, Car, and CarLot. The Date class will take the date input and validate it. Also I’d like to build a function that display’s the date as month/ day/ year. The Car class will have two different constructs, each taking the information about the car and one taking the information about the sale of the car if the car was sold. Also for the Car class there will be a member function that will return the profit from the car if it has been sold. The third class CarLot will use the vector Car as its data member and the member functions will act on the CarLot object to add cars to the lot, get the lot inventory and get the profits for a particular month.

I think the biggest issue this week will be understanding how to work with all the layers and get the data I need when I need it. I’m still not completely sure how to access the date when it’s on the most basic layer of the program. By practicing this I think it will help prepare me and the others in the class for programs that will be far more intensive and have far more layers.

DESIGN

Pseudocode: carLot2.cpp

CLASS Date

PRIVATE DECLARATIONS

int day

int month

int year

PUBLIC DECLARATIONS

Date ()

void setDate (int d, int m, int y)

IF year is invalid

re-input year

IF month is invalid

re-input month

IF day is not in month

re-input day

day = d

month = m

year = y

int getDay

return day

int getMonth

return month

int getYear

return year

CLASS Car

PRIVATE DECLARATIONS

string make

string model

int year

Date datePurchased

double purchasePrice

bool isSold

Date dateSold

double salePrice

PUBLIC DECLARATIONS

Car (string make, string model, year, Date datePurchased, double purchasePrice, bool isSold, Date dateSold, double salePrice)

Car (string make, string model, year, Date datePurchased, double purchasePrice, bool isSold)

void setMake (string m)

make = m

string getMake

return make

void setModel (string m)

model = m

string getModel

return model

void setYear (int y)

year = y

int getYear

return year

void setDatePurchased(int d, int m, int y)

datePurchased.setDate (d,m,y)

Date getDatePurchased

Date datePurchased

datePurchased.getDay

datePurchased.getMonth

datePurchased.getYear

return datePurchased

void setPurchasePrice (double p)

purchasePrice = p

double getPurchasePrice

return purchasePrice

void setIsSold (bool sold)

isSold = sold

bool getIsSold

return isSold

void setDateSold(int m, int d, int y)

dateSold.setDate(d,m,y)

Date getDateSold

dateSold.getDay

dateSold.getMonth

dateSold.getYear

return dateSold

void setSalePrice (double sale)

salePrice = sale

double getSalePrice

return salePrice

double getProfits ()

double profits

Car car

profits = getPurchasePrice() – getSalePrice()

return profits

CLASS CarLot

PRIVATE

vector carLot

PUBLIC

CarLot()

CarLot(vector carLot)

Vector getCarLot

return carLot

void addCar(vector &carLot)

void listCurrent(vector &carLot)

void getMonthProfit(vector &carLot)

void addCar (vector &carLot)

DECLARE Local Object Car newCar

DECLARE local variables: string make, string model, int year, int dayP, int dayS, int monthS, int monthP, int yearP, int yearS, double priceP, double priceS, bool isSold, char sold

OUTPUT Enter make

INPUT make

SET newCar.make

OUTPUT Enter model

INPUT model

SET newCar.model

OUTPUT Enter year

INPUT year

SET newCar.year

OUTPUT Enter Day Purchased

INPUT day purchased

OUTPUT Enter Month Purchased

INPUT month purchased

OUTPUT Enter Year Purchased

INPUT year purchased

SET newCar.datePurchase(month, day, year)

OUTPUT Enter Purchase Price

INPUT purchase price

SET newCar.purchasePrice

OUTPUT Has car been sold (y/n)

INPUT sold

IF sold = y or Y

isSold = true

SET newCar.isSold

OUTPUT Enter day sold

INPUT day sold

OUTPUT Enter month sold

INPUT month sold

OUTPUT Enter Year sold

INPUT year sold

SET newCar.dateSold(day, month, year)

OUTPUT Enter Sale Price

INPUT sale price

SET newCar.salePrice

ELSE IF sold = n

isSold = false

SET newCar.isSold

ADD newCar to Vector &carLot

END function

void listCurrentInventory(vector &carLot)

int carsInLot

FOR EACH car in carLot

IF carLot.isSold = false

OUTPUT Make

OUTPUT Model

OUTPUT Date Purchased

OUTPUT Purchase Price

ADD to carsInLot counter

RETURN carsInLot

END function

Double getMonthProfit (vector &carLot)

DECLARE LOCAL VARIABLES: Date purchaseDate, Date saleDate, double totalPurchasePrice, double totalSales, double monthProfits, int month, int day, int year

OUTPUT please enter a year

INPUT year

OUTPUT please enter a month

INPUT month

VALIDATE month

FOR EACH car in carLot

GET purchaseDate

GET soldDate

IF purchaseDate.year = year AND purchaseDate.month = month

ADD purchase price to totalPurchasePrice

IF soldDate.year = year AND soldDate.month = month

ADD sale price to totalSales

OUTPUT totalSales

OUTPUT totalPurchases

OUTPUT totalProfits

OUTPUT cars sold AND profits on each car sold

END FUNCTION

PROTOTYPE FUNCTIONS

displayMenu

getChoice

Main Function

DECLARE VARIABLES

CarLot object

Vector car

Int choice

Char ch

DO

CALL displayMenu function

Choice = CALL getChoice function

Switch choice

Case 1 CALL addCar(car)

Case 2 CALL listCurrentInv(car)

Case 3 CALL getMonthProfit(car)

Case 4 Quit

WHILE choice doesn’t equal 4

END Main

DISPLAY menu function

OUTPUT menu choices

CHOICE 1 Add Car

CHOICE 2 List Current Inventory

CHOICE 3 Get Month Profits

CHOICE 4 Quit

END function

INT getChoice function

DECLARE int choice

INPUT choice

WHILE choice is invalid

REINPUT choice

RETURN choice

END function

TESTING

|  |  |  |
| --- | --- | --- |
| INPUT | EXPECTED OUTPUT | ACTUAL OUTPUT |
| Menu input: 5 | Please enter a number between 1 and 4 | Expected |
| Menu input: F | Please enter a valid choice | Expected |
| Menu Input: 1 | Enter Make | Expected |
| Year Input 2017 | Enter a valid year | Expected |
| Year input 4 | Enter a valid year | Expected |
| Year input ffff | Enter a valid year | Expected |
| Year input 1920 | Enter day | Expected |
| Day input: 31 | Enter month | Expected |
| Month input: 9 | There are not that many days in that month | Expected |
| Year input : fff | Enter a valid year | Expected |
| Year input 1982 | Enter purchase price | Expected |
| Price input:ff | Invalid price | Expected |
| Price input 12.02 | Has the car been sold? Enter Y or N | Expected |
| Y or N input f | Please enter y or n | Expected |
| Y or N input Y | Enter day sold | Expected |
| Day: 31 | Enter month | Expected |
| Month: 9 | Day is not in month re-enter | Expected |
| Year 1980 | Enter sale price |  |
| Sale price 200 | Press enter to return to menu | Expected |
| Menu input: 2 | No values | Expected |
| Menu input 3 | Enter a year | Expected |
| Year: 1980 month 9 | Total sales $200.00  Total profit $200.00 | Expected |
| Menu input: 4 | Good-bye | Expected |

REFLECTION

As I mentioned previously my biggest issue was with accessing and setting all of the data on each of the different layers of the program. I specifically had problems with the date, because setting the date meant first sending the completed date to the Car class then sending the different components of the date to the Date class to then verify that the date was valid. Retrieving the date also was challenging for the same reasons. I understand the reasoning behind having classes and objects is to have related data and functions to be grouped together, but this can be a challenge to handle.

Beyond the different layers of classes I really didn’t have any other major problems with the program except with how to build the constructors and the parameters of the constructors. I really had to play around with the syntax of the constructors to get the program to work, I think there may have been a better way of doing so but as of now I haven’t figured it out.

Testing while coding was a bit different than what I have done throughout the course as well. Throughout the rest of the course I have been placing phrases in the project to see if data is being passed properly and to see if certain parts of the program are being called properly, with the use of the classes I had to wait until everything was built, including main, to start debugging. It made the process of debugging more tedious because there were so many more errors than there had been previously. Luckily the other projects leading up to this one prepared me for how to handle these errors and the program eventually got to work exactly how I anticipated. One key difference between this program and the others I’ve done previously is that the main program is far cleaner than others because all the functions are being called from somewhere else.