COMP 2404
Introduction to Software Engineering
Assignment 2

Released: Wednesday February 7, 2018, 12:00 Noon

Due: Wednesday February 21, 2018, 12:00 Noon

Toby's Auto Mechanic Information Management System

Instructions

In this assignment you will essentially convert the program you created in Assignment 1 into a program written in C++ using classes and object oriented design principles. A large part of this program has been provided to you to build from. It was designed using the model–view–controller design pattern that you should be familiar with. The various classes are described below.

Controller

The **ShopController** class acts as the controller for this program. It is the only class used by our main function (in main.cc). It maintains the model as well as the view. It prompts both the user for information and displays information via the **View** class (described below) and modifies the various model classes as appropriate.

View

The **View** class as the view for this program. All interaction with the user (including both input and output) is done using the view. These actions are prompted by the controller who then modifies the model in response.

Model

The Model is made up of a number of different classes. The **Shop** class will collect all of the objects that make up this automotive business. Currently it will only contain a list of customers (**Customer** class). Each of these customers will contain a list of vehicles (**Vehicle** class). These

collections are stored in their own respective collection classes (**CustomerArray** and **VehicleArray** classes, respectively).

Task 1: UML Diagram

You will first study the provided code to get an understanding of how all of the classes work together. Based solely on what is provided, draw a UML diagram for this program. Refer to the lecture notes for exactly what is expected to be included and omitted on a UML diagram. This diagram (a scan of a hand drawn diagram is acceptable) will be a part of your assignment submission.

Task 2: Finish the Program

Your program **must** use the provided skeleton code **without making any changes to the existing code, data structures or function prototypes.** You will, however, add your own code to it. Below are the required changes you must make to for your program.

1. **Customer and Vehicle classes**
   You are to implement all of the member functions declared in the respective header files for these classes. Some notes on what is expected for the Vehicle class:
   - The constructor simply sets all of the data members to the parameters passed in.
   - The getter functions are simple, one line getters.

   Some notes on what is expected for the Customer class:
   - The static member **nextId** is used to give all of the Customer objects unique IDs. Initialize it to 1000.
   - The constructor sets the **id** member to the current value of the static member **nextId**, increments **nextId** and then simply sets all of other data members to the parameters passed in.
   - The getter functions are simple, one line getters.
   - The **addVehicle** function simply called the **add** function in the **VehicleArray** class (described below). It returns what this **add** function returns.

2. **CustomerArray and VehicleArray classes**
   These classes are collection classes that will store multiple Customer and Vehicle objects, respectively. Implementing this in a separate class will allow us to modify the data structure used at a later date without having to make changes to other classes (as we will do on a future assignment). These classes are very similar. Some notes on what is expected for both classes:
   - The constructor simply initializes the **size** data member properly,
   - The destructor must iterate over the stored objects (which are to be dynamically allocated, discussed below), freeing the allocated memory for each one.
   - The **getSize** function is a simple, one line getter.

- The **add** function takes a pointer to a dynamically allocated object. It checks if there is room in the array of pointers for this object. If not, it returns **C_NOK** (defined in defs.h). If there is, it sets the appropriate pointer in the array to point at the same object as the parameter, increments the **size** data member and returns **C_OK** (also defined in defs.h).
- The **get** function takes an index value. If this index is not valid (ie. it is outside of the range of valid objects being stored), the function returns 0. Otherwise it returns the pointer at index i.

3. **ShopController class**
   The ShopController class contains a function called **initCustomers** which we will use to populate the program with data. This function will create **dynamically allocated** objects. It will work as follows. First it will create a **Customer** object followed by several **Vehicle** objects. You will add the **Vehicle** objects to the **Customer** using the Customer's **addVehicle** member function. Once a **Customer** object has had all of its **Vehicles** added, you will add the **Customer** to the **Shop** object using its **addCustomer** member function. You will do this until the data fill requirements have been met.

   You will add 6 **Customers** to the shop. All of your customers must have at least one vehicle registered. On top of this, you are to have at least one customer with 2 vehicles, at least one customer with 3 vehicles and at least one customer with 4 vehicles.

Once these three implementation tasks have been completed you should be able to compile and run the program. The main menu currently only has the option to print customers, which should work.

Constraints

- you must use the function prototypes exactly as stated
- your program must not have any memory leaks (don't worry if valgrind reports that some memory on the heap is "still reachable")
- do not use any global variables
- your program must reuse functions everywhere possible
- your program must be thoroughly commented
- **your program must compile and run in the provided Virtual Machine**

Submission

You will submit in cuLearn, before the due date and time, one **tar** file that includes all the following:

- A UML diagram for this program
- all source code, including the code provided, if applicable
- a readme file that includes:
  - a preamble (program author, purpose, list of source/header/data files)
  - the exact compilation command
  - launching and operating instructions

Grading [out of 25 marks]

Marking components:

- 7 marks: UML diagram
- 4 marks: Customer class
    - 1 mark: initializing static member
    - 1 mark: constructor
    - 1 mark: getters
- 2 marks: Vehicle class
    - 1 mark: constructor
    - 1 mark getters
- 5 marks: CustomerArray class
    - 1 mark: constructor & getSize function
    - 2 marks: destructor
    - 1 mark: add function
    - 1 mark: get function.
- 5 marks: VehicleArray class
    - 1 mark: constructor & getSize function
    - 2 marks: destructor
    - 1 mark: add function
    - 1 mark: get function
- 2 marks: ShopController class
    - data fill done to specification, using dynamically allocated memory

Notes:

In order to get credit for a marking component, the program must prove that the marking component executes successfully. This is usually accomplished by printing out correct data.

Deductions

- Packaging errors:
    - 10% for missing readme
- Major programming and design errors:
    - 50% of a marking component that uses global variables
    - 50% of a marking component that consistently fails to use correct design principles, including separate functions
    - 50% of a marking component where unauthorized changes have been made to provided code or prototypes
- Minor programming errors:
    - 10% for consistently missing comments or other bad style
    - 10% for consistently failing to perform basic error checking
- Execution errors:
    - 100% of a marking component that can’t be tested because the code doesn’t compile or execute in the VM
    - 100% of a marking component that can’t be tested because the feature isn’t used in the code
    - 100% of a marking component that can’t be proven to run successfully because data is not printed out

Sample Console Output

```
$ ls
CustomerArray.cc  Customer.h  Makefile           ShopController.h  VehicleArray.h
View.cc
CustomerArray.h   defs.h      Shop.cc            Shop.h            Vehicle.cc
View.h
Customer.cc       main.cc     ShopController.cc  VehicleArray.cc   Vehicle.h
$ make
g++ -c main.cc
g++ -c ShopController.cc
g++ -c View.cc
g++ -c Shop.cc
g++ -c CustomerArray.cc
g++ -c VehicleArray.cc
g++ -c Customer.cc
g++ -c Vehicle.cc
g++ -o mechanicshop main.o ShopController.o View.o Shop.o CustomerArray.o
VehicleArray.o Customer.o Vehicle.o
$ ./mechanicshop



         **** Toby's Auto Mechanic Information Management System ****

                                MAIN MENU

        1. Print Customer Database

        0. Exit

Enter your selection:  1

CUSTOMERS:

Customer ID 1000

    Name:                          Maurice Maurice
    Address:                   2600 Colonel By Dr.
    Phone Number:                    (613)728-9568

    1 vehicle(s):

       1)     Red 2007       Ford Fiesta (100000km)


Customer ID 1001

    Name:                          Abigail Abigail
```

Address: 43 Carling Dr.
Phone Number: (613)345-6743

1 vehicle(s):

1) Green 2016 Subaru Forester (40000km)

Customer ID 1002

Name: Brook Brook
Address: 1 Bayshore Dr.
Phone Number: (613)123-7456

2 vehicle(s):

1) White 2018 Honda Accord (5000km)
2) White 1972 Volkswagon Beetle (5000km)

Customer ID 1003

Name: Ethan Ethan
Address: 245 Rideau St.
Phone Number: (613)234-9677

1 vehicle(s):

1) Black 2010 Toyota Camery (50000km)

Customer ID 1004

Name: Eve Eve
Address: 75 Bronson Ave.
Phone Number: (613)456-2345

3 vehicle(s):

1) Green 2013 Toyota Corolla (80000km)
2) Gold 2015 Toyota Rav4 (20000km)
3) Blue 2017 Toyota Prius (10000km)

Customer ID 1005

Name: Victor Victor
Address: 425 O'Connor St.
Phone Number: (613)432-7622

```
    4 vehicle(s):

       1)  Purple 2012          GM Envoy (60000km)
       2)   Black 2016       GM Escalade (40000km)
       3)     Red 2015         GM Malibu (20000km)
       4)  Orange 2012    GM Trailblazer (90000km)

Press enter to continue...

         **** Toby's Auto Mechanic Information Management System ****

                                MAIN MENU

        1. Print Customer Database

        0. Exit

Enter your selection:  0

$
```