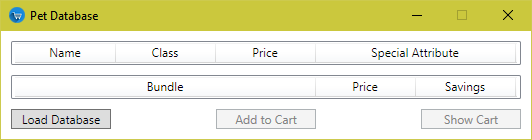
**CSE 335 Spring 2018: Project 2**

**Project Description**

In this project, you will create a GUI application based on the Pet database from project. The application must make use of Abstract Factory Pattern, Composite Pattern, Visitor Pattern, Builder Pattern, Observer Pattern and Distributed Collaboration Pattern learned in the course. The application GUI must be built in Qt. Following screenshots describe the working of the Pet database application.

1. The application’s main window is shown below. It contains five widgets: (1) “Load database” button reads data from two database files and populates widgets 2 and 3 alphabetically according to Names; (2) A table view with 4 columns (Name, Class, Price, and Special Attribute) to display pets data; (3) A table view with 3 columns (Bundle, Price and Savings) to display bundles data; (4) “Add to Cart” button to add item to a shopping cart; and (5) “Show Cart” button to display the shopping cart.

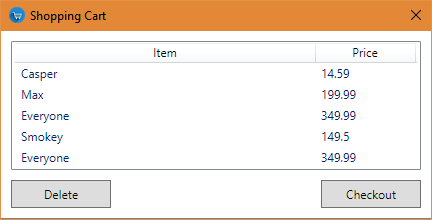


1. Two database files “Pets.csv” and “Bundles.csv” are provided with the zip file. In “Pets.csv”, each line of text represents a Pet object’s Class, Name, Type, Weight, Price and other attribute in Comma-Separated Values format. In “Bundles.csv”, each line of text represents a Bundle object’s Name, Price, and different amount of Pets within the bundle in Comma-Separated Values format. When constructing a Bundle object, the Savings are determined by first calculating the difference between bundle price and the sum of original Pet prices, then dividing by the sum. A screenshot of main window with database loaded on the next page. You will exercise Abstract Factory Pattern, Composite Pattern, Builder pattern and Visitor Pattern in database construction.

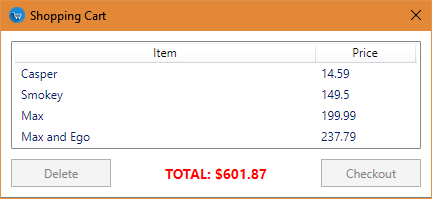


1. Any item listed in the main window may be chosen and added to a shopping cart by clicking “Add to Cart” button. The shopping cart is not shown until the “Show Cart” button is clicked. Items can be added to shopping cart regardless whether the cart is shown. The cart can hold an arbitrary number of items. On clicking “Show Cart” button, the shopping cart window is shown separately. The main database window and shopping cart window can function simultaneously, so items can be added to the shopping cart, and the shopping cart window is updated accordingly. The shopping cart window may be closed, and the same shopping cart will be shown again the next time “Show Cart” button is clicked. The same item may be added multiple times. Any item in the shopping cart can be selected and then removed by clicking on the “Delete” button. A screenshot of main window and five items in shopping cart is shown below. You will exercise Observer Pattern and Distributed Collaboration Pattern when designing these windows.

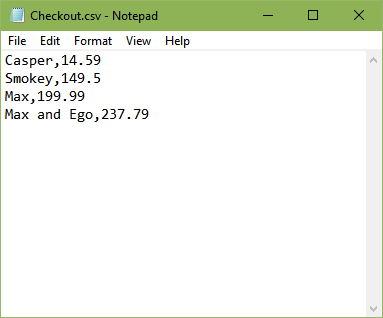




1. When the “Checkout” button is clicked, the shopping cart is first sorted by price, then its contents written to a CSV file. The total price is displayed in bold red font, and all application features are disabled after checking out. The only available option is to close the application; however, the application is not closed automatically when checking out. A screenshot of checked out shopping cart window is shown below. It contains three widgets: (1) A table view with 2 columns (Item and Price) to show cart data, (2) “Delete” button and (3) “Checkout” button. You will exercise Visitor pattern when calculating total price.



1. Checking out produces a “Checkout.csv” as shown below. A screenshot of the file containing items that were in the shopping cart at the time of checkout is shown below. You will exercise Visitor pattern and Builder pattern when writing the output file.



**Coding Requirements**

1. You should have correct abstraction of classes that separates data and algorithms.
2. Draw a UML Diagram.
3. Implement your code such that it follows proper privacy and give clients only the interfaces they need. Clients should not access data members directly.
4. Your output should match the sample output.
5. Your program should not have memory leaks.
6. Provide *Get* and *Set* functions for all necessary data members.
7. All member function that do not change values of member variable should be constant. For all calls by reference, if it does not make changes to the variable passed by reference then they should refer to const.
8. Each class should be either defined and implemented in one header file or defined in a header file and implement in a cpp file. No two classes should be defined or implemented in one file.

**Submission Guidelines:**

1. Work in teams of two to design the UML and implement the project.
2. You must have received an email about your team member. You are not allowed to change teammate. If you have issues or concerns about your project team, email TA Salman Ali at [alisalm1@msu.edu](mailto:alisalm1@msu.edu) ASAP.
3. Each team must meet at least TWO times. For each meeting, you need to fill a Team Meeting Report in detail. If your teammate does not respond or participate in your project meeting, then you need to include email proof along with your submission.
4. Each team must submit a UML along with the project code and team meeting reports.
5. You must submit a *zip* file that contain team participation report and one directory:

(a) A UML diagram (PDF document) including:

i. NetID of each group member,

ii. Full name of each group member

(b) Submit the whole Qt project directory. You need to ensure that Qt can open your project, compile and run. Your project will not be graded if it does not compile.

(c) Each team submits TWO meeting reports in their project submission.

1. This project is due via Handin (<https://secure.cse.msu.edu/handin/>) by 11:59 PM on 04/22/2018.