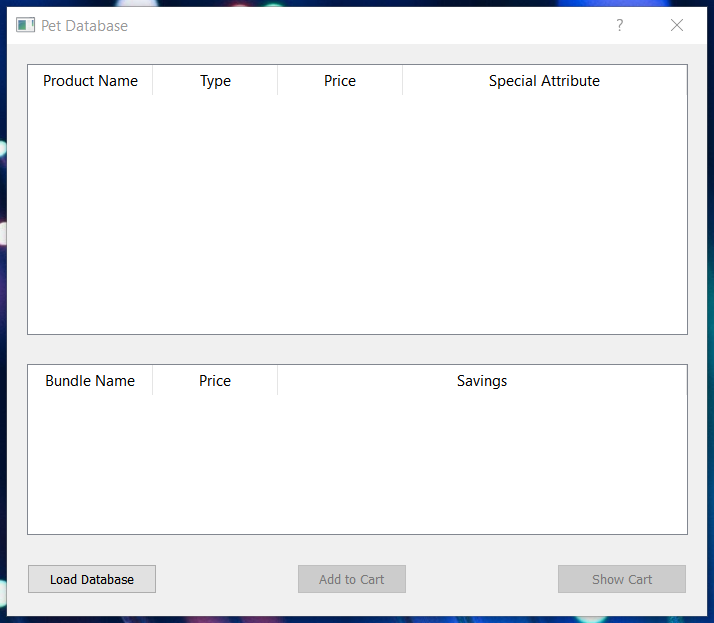
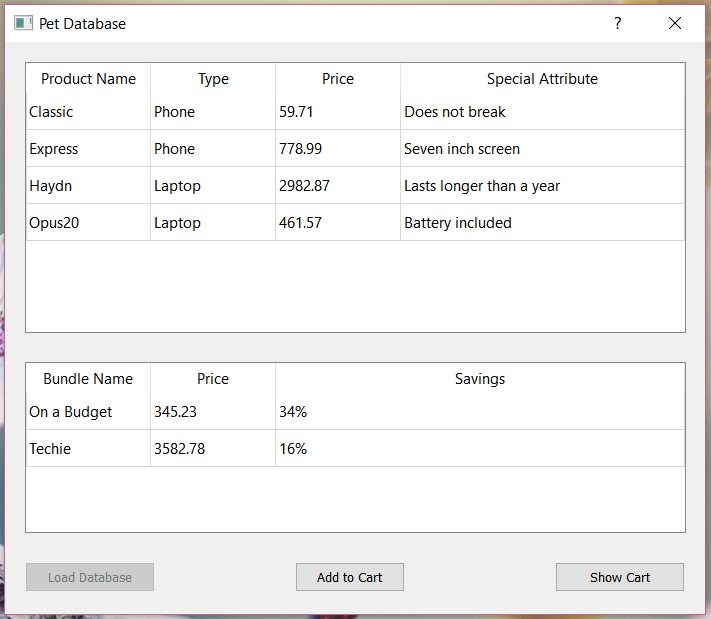
**CSE 335: Project 2**

**Project Description:**

You are tasked with making a GUI application for a technology store using the observer, builder, visitor, and mediator patterns. The software will be made using Qt. The owner of the store wants to import products from a csv file, be able to add those items to a shopping cart, and check out the cart by pressing a button. Here is an example of how this program should run. When the program starts out, there should be a window that looks like this. It’s pretty basic, just two QTableWidgets, and three buttons.

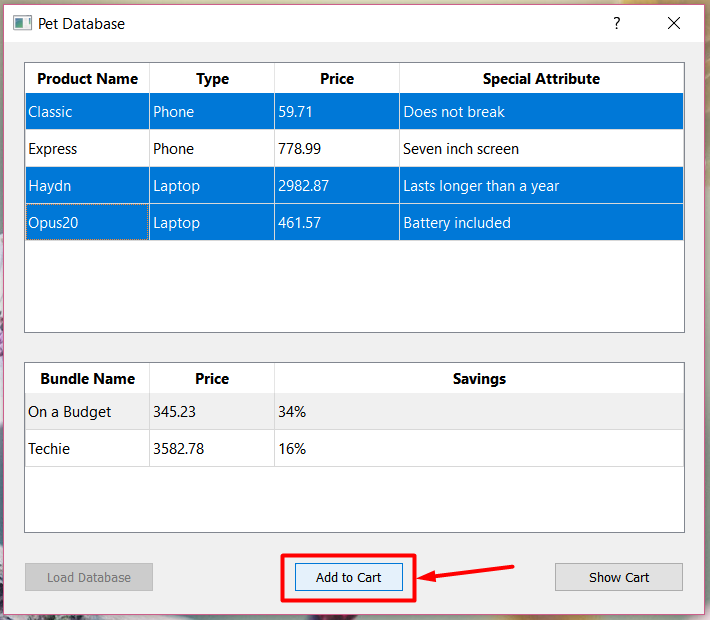


After clicking the “Load Database” button, the application should read the supplied csv files and output the data to the screen. It should look like this.

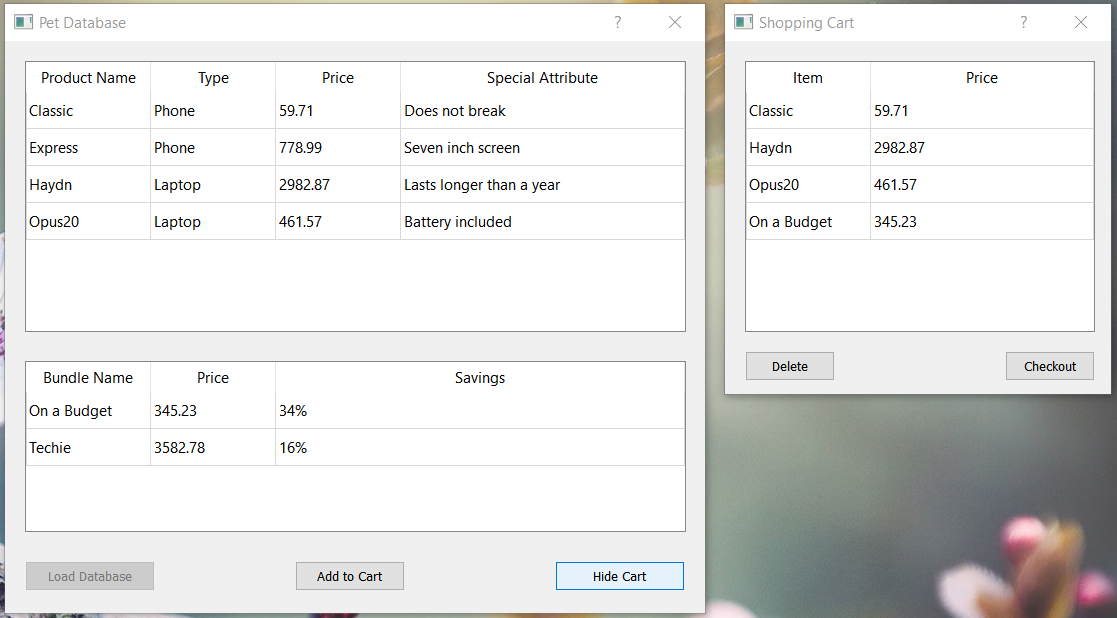


There are two files provided: one for each table. The first file, “Technology.csv” will contain items separated by commas. There are 6 items in each line: technology type, product name, company name, weight, price, and a fun fact. The second file will contain bundles of items. First, there is the name of the bundle, then the price for everything, and all items after that are what the bundle consists of. When displaying, you should calculate and display the percent saved by buying the bundle too.

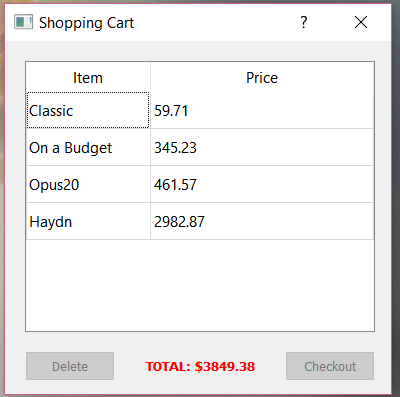
You should be able to click items listed in the tables, and add them to your shopping cart. Anything in either section can be added to the cart. You should also be able to add from both tables at the same time as in the example below.



Here is what the shopping cart should look like at this point. If you made a mistake, you should also be able to delete items from your shopping list.



And you should be able to close out your purchase, and see the total price of the items.



Notice that some buttons change as the state of the program changes. To begin with, only “Load Database” is pressable. After that, it is no longer pressable, but the other two buttons are. Buttons may also change after being pressed as is the case with “Show Cart” and “Hide Cart”.

**Suggestions:**

Start this project off with a UML. Just as you don’t build a skyscraper without a plan, you shouldn’t start coding without a design. UML is the blueprint for your code. Each pattern has a specific job it does. Try to figure that job in the context of this project, and then work them all together.

Reading from a csv file in Qt can be challenging because of how the pathing works for executables. I recommend getting this working sooner than later. To help you out, here is a little test you can run. I put it in the function that would be loading the data from the CSV.

“”””””””””””

#include <QDebug>

#include <QDir>

#include <QFile>

#include <QList>

//This is the value for me to be able to read CSV files. Maybe you'll need to modify it a bit

QString backPath = "../../project2/";

QFile test(backPath + "Technology.csv");

if(!test.open(QIODevice::ReadOnly))

{

// gives the reason opening the file didn’t work. Usually because it wasn’t found.

qDebug() << test.errorString();

// gives the current path the program’s .exe file is running at

qDebug() << QDir::currentPath();

throw test.error();

}

else

{

qDebug() << "File Read!";

}

“”””””””””””””

**Coding Requirements:**

1. You should have correct abstraction of classes that separates data and algorithms.
2. Draw a UML diagram.
3. Your program should run as described in the project scenario above.
4. Implement your code such that it follows proper privacy and give clients only the interfaces they need. Clients should not access data members directly.
5. Your program should not have memory leaks.
6. Provide *Get* and *Set* functions for all necessary data members.
7. For 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 teammates. If you have issues or concerns about your project team, email TA Daniel Shumaker at shumak37@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 contains your UML, team participation reports, and one directory:
   1. A UML diagram (PDF document) including:
      1. NetID of each group member,
      2. Full name of each group member
   2. Each team submits TWO meeting reports in their project submission.
   3. The whole 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.
6. This project is due via Handin ([https://handin.cse.msu.edu/login.php/](https://secure.cse.msu.edu/handin/)) by 11:59 PM on 04/22/2019.