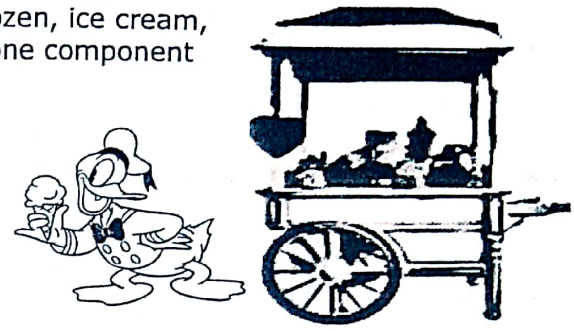
Objective:

- Exploring an Application of Polymorphism.

Challenge: F18 - CS & SE - Dessert Shoppe (51)

Thanks To: [https://www.csc.ncsu.edu/people/spbalik]

In this Laboratory, you will be writing software in support of a Dessert Shoppe, which sells candy by the pound, cookies by the dozen, ice cream, and sundaes (ice cream with a topping). You will develop one component of this software i.e. checkout system. The interfacing will be dealt by another software development team so you don't have to worry about it.



To do this, you will implement an inheritance hierarchy of classes derived from a `DessertItem` abstract base class. The `Candy`, `Cookie`, and `IceCream` classes will be derived from the `DessertItem` class.
The `Sundae` class will be derived from the `IceCream` class.
You will also write a `Checkout` class which maintains a list (a resizable array) of `DessertItem's`.

The DessertItem Class

The `DessertItem` class is an *abstract base class* from which specific types of `DessertItems` can be derived. It contains only one data member, a name. It also defines a number of methods. All of the `DessertItem` class methods except the `getCost()` are defined in a generic way in the class, `DessertItem.cpp`. The `getCost()` method is an *abstract method* that is not defined in the `DessertItem` class because the method of determining the costs varies based on the type of item.

The DessertShoppe Class

It contains constants such as the tax rate as well the name of the store, the width used to display the costs of the items and item name width to be displayed on the receipt. Your code should use these constants wherever necessary! The `DessertShoppe` class also contains the `centsToDollars` method, which takes an integer number of cents and returns it as a String formatted in dollars and cents. For example, 105 cents would be returned as "1.05".

The Derived Classes

- The `Candy` class should be derived from the `DessertItem` class. A `Candy` item has a *weight* and a *price per pound*, which are used to determine its *cost*. For example, 2.30 lbs. of fudge @ .89 $/lb. = 205 cents.
- The `Cookie` class should be derived from the `DessertItem` class. A `Cookie` item has a *number* and a *price per dozen* which are used to determine its *cost*. For example, 4 cookies @ 399 cents /dz. = 133 cents.
- The `IceCream` class should be derived from the `DessertItem` class. An `IceCream` item simply has a *cost*.
- The `Sundae` class should be derived from the `IceCream` class. The *cost* of a Sundae is the *cost of the IceCream* plus the *cost of the topping*.

The Checkout Class

The `Checkout` class behaves as a point of sale, provides methods to enter dessert items into the cash register, clear the cash register, maintains the number of items, maintains the total cost of the items (before tax), maintains the total tax for the items, and maintains a string representing a receipt for the dessert items. The `Checkout` class should use an array to store the `DessertItem's` and their count of orders for a particular item.

Assumptions and General Guidelines

- C++ string class has been used in few places of code provided in this document, Not good for you ☺. You must use CString class or char *.
- All type/classes of dessert items will store their price in cents but while printing the bill, the cost will be converted and displayed in dollars.
- All the item prices (including total tax and total bill) on receipt will be printed as rounded to nearest cent. But total bill and tax will be calculated on actual values to get the maximum possible accuracy. (see the sample output of receipt in this regard.)
- Other than what has been specified, you are free to decide about adding any private **data member(s)** or any **ctor/dtor** or any **member functions** needed for your classes but you got to justify your decision to the TA, failing to give some acceptable/reasonable justification will result in less or zero marks. Note: This liberty is not for `CString` and `Array` class.
- You may assume that arguments passed to functions will be valid.
- You may assume that the items stored in the checkout will be unique.
- Just to save printing paper, I have defined function in header files but you must do and should already know, where to put those function definitions.
- You have mainly three tasks to implement in this lab:
  - Implement the hierarchy of classes for dessert items.
  - Checkout class and DessertShoppe class
  - Bill printing which is part of checkout class. You strictly got to follow the format as shown in the sample run.
  - You don't need to worry about the interface that how items will be scanned on point of sale and will be added to checkout class. This part is assigned to another development team which will use your classes. For your help, I have given a sample run which should produce the output desired, which will also help you understand the purpose of classes.

***** *Classes* *****

DessertItem.h

```
class DessertItem
{
    CString name ;
public:
    DessertItem(CString = "");
    CString getName() const;
    virtual double getCost() const = 0;
    virtual double getTax() const;
};
```

DessertShoppe.h

```
class DessertShoppe
{
    static double _TAX_Rate;
    static const CString _STORE_Name;
    static const int _MAX_ITEM_NAME_WIDTH;
public:
    static string centsToDollars(int cents)//for your help: may be a cattywampus
    {
        string s = "";

        if (cents < 0)
        {
            s += "-";
            cents *= -1;
        }
        int dollars = cents/100;
        cents = cents % 100;

        if (dollars > 0)
            s += dollars;
```

```
        s +=".";

        if (cents < 10)
            s += "0";

        s += cents;
        return s;
    }
};
```

Class Checkout

Maintains a list of DessertItem references. There is no limit to the number of DessertItem's in the list

```
class Checkout
{
    DessertItem * * list;  // list of different Dessert Items
    Array countPerItem;    //count of items at list[i] is stored at countPerItem[i]
    int itemsCount=0;
    int listCapacity;
public:
    Checkout();    //Initializes the object with some fix size of DessertItem array
    void clear();  //Clears the Checkout to begin checking out a new set of items i.e. the next
                   //customer in line.
    void enterItem(const DessertItem & item, int cnt=1);    //A deep copy of DessertItem is
                                                            //added to the end of the list of
                                                            //items with default count of 1.
    CString toString();
                /*Returns a String representing a receipt for the current list of items. i.e. a
                String representing a receipt for the current list of DessertItem's with the
                name of the Dessert store, the items purchased, the tax, and the total cost. See
                the sample run output for further detail.*/
    double totalCost();    //Returns total cost of items
    double totalTax();     //Returns total tax on items
};
```

Sample Run

```
//Driver.cpp
int main()
{
    DessertShoppe::setTaxRate(17.5);
    Checkout checkout;
    checkout.enterItem (Candy(2.25, 399, "Fall 18: Special Platter of Candies : Pop Rocks
Candy+Nerds Candy+Swedish Fish Candy+Candy Corn"),3);
    checkout.enterItem (IceCream(105, "Vanilla Ice Cream"), 10);
    checkout.enterItem (Sundae(145, 50, "Chocolate Chip Ice Cream with Pinapple and Almonds"),
2);
    checkout.enterItem (Cookiee(399, 15, "Oatmeal Raisin Cookies"), 2);

    cout<<checkout.toString();

    checkout.clear();

    checkout.enterItem(IceCream(145, "Strawberry Ice Cream"));
    checkout.enterItem(Sundae(105, 50, "Caramel"));
    checkout.enterItem(Candy(1.33, 89, "Gummy Worms"));
    checkout.enterItem(Cookiee(44, 30, "Chocolate Chip Cookies"));
    checkout.enterItem(Candy(1.55, 209, "Salt Water Taffy"));
    checkout.enterItem(Candy(3.0, 109, "Candy Corn"));

    cout<<"\n\n"<<checkout.toString();

    return 0;
}
```

Sample Run Output

```
   Fall-18 : CS & SE : Dessert Shoppe
-----------------------------------------
Fall 18: Special Platter.....26.94 = 3 x 8.98
 of Candies : Pop Rocks
Candy+Nerds Candy+Swedis
h Fish Candy+Candy Corn
Vanilla Ice Cream.....10.50 = 10 x 1.05
Chocolate Chip Ice Cream.....3.90 = 2 x 1.95
 with Pinapple and Almon
ds
Oatmeal Raisin Cookies.....9.98 = 2 x 4.99
Tax~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~8.98
Total Cost~~~~~~~~~~~~~~~~~~~~~~~~~~~~~60.29

   Fall-18 : CS & SE : Dessert Shoppe
-----------------------------------------
Strawberry Ice Cream.....1.45 = 1 x 1.45
Caramel.....1.55 = 1 x 1.55
Gummy Worms.....1.18 = 1 x 1.18
Chocolate Chip Cookies.....1.10 = 1 x 1.10
Salt Water Taffy.....3.24 = 1 x 3.24
Candy Corn.....3.27 = 1 x 3.27
Tax~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~2.06
Total Cost~~~~~~~~~~~~~~~~~~~~~~~~~~~~~13.86
```

"The hardest job kids face today is learning good manners without seeing any."

-- Fred Astaire --

"Trying to improve software quality by increasing the amount of testing is like trying to lose weight by weighing yourself more often. What you eat before you step onto the scale determines how much you will weigh, and the software-development techniques you use determine how many errors testing will find."

-- Steve McConnell, Code Complete --