İTÜ Computer Engineering Department
BLG252E Object Oriented Programming
2nd Homework

Due Date: April 28, 2013 23.00 PM

Consider a simple shopping interface where users can order items in 2 categories: Flower (Rose and Daisy) and Gourmet (FruitBasket and CookieBasket). In this system, all the items have following common properties:

- price of the item (double)
- calculate_price() method: to calculate the price of the item
- print() method: to print information about the item

Flower objects have a common property artificial (bool) for specifying if the ordered flower is artificial or fresh. The price of an artificial flower is 1.5 times of its fresh version. Roses have a number (int) attribute for storing the number of ordered roses and the price is 10 TL for each fresh rose. If at least 10 roses are ordered, the unit price becomes 8 TL. Daisies are only sold in a bunch (without a number attribute) and the price is 20 TL for a bunch of fresh daisies.

Gourmet objects have two attributes in common: basket_size (enum {small, medium, large}) for specifying the size of the item and promotion (bool) to determine if there is a promotion with the order or not. FruitBaskets have fruitmix_type (enum: standard, citrus, tropical) for the type of fruits and with_chocolate_sauce (bool) to specify if chocolate sauce will be added or not for an additional cost of 20 TL. The price list of a FruitBasket for different types and sizes is as follows:

	Standard	Citrus	Tropical
Small	25	40	50
Medium	50	80	100
Large	75	120	150

CookieBasket objects do not have different types and the price is determined by only using the basket size as, 30 TL for a small, 50 TL for a medium and 70 TL for a large basket of cookies. A bunch of daisies are given free as a promotion, when the price of the FruitBasket is at least 80 TL or a large CookieBasket is ordered.

In this assignment, you will design C++ classes to model different types of items in the shop (**Rose**, **Daisy**, **FruitBasket** and **CookieBasket**). Use inheritance and polymorphism effectively to be consistent with the properties mentioned above. For classes from which no object instance is created, prefer to make them **abstract**. Your design should **avoid repetition** as much as possible. The classes you provide **must** be compatible with the test code given below. While grading the homework this test code will be used as it is, so do not modify test_code.cpp. Write all your classes in **item.h** and **item.cpp** files. Write all declarations and bodies of necessary methods.

Test code (test_code.cpp):

```
#include <iostream>
#include "item.h"
using namespace std;
#define MAX_ORDER_PER_SESSION 5  // max. 5 orders are allowed per session

int main(){
    Item *orders[MAX_ORDER_PER_SESSION];   // an array of pointers to ordered items
    int item_count = 0;
    bool continue_shopping = true;
    char category, subcategory;
    // ask for order information in a loop until the user ends shopping or max limit is reached
    while(continue_shopping && item_count < MAX_ORDER_PER_SESSION){
        cout << "-------------------------------------------------------------------------" << endl;
        cout << "Select category ([f] for flower, [g] for gourmet, [e] to end shopping): ";
        cin >> category;
```

```
        switch(category){
            case 'f':
                bool is_artificial;
                cout << endl << "Fresh flower[0] or artificial flower[1]?: ";
                cin >> is_artificial;
                cout << endl << "Select subcategory ([r] for rose, [d] for daisy): ";
                cin >> subcategory;
                switch(subcategory){
                    case 'r':
                        int amount;
                        cout << endl << "Amount?: ";
                        cin >> amount;
                        orders[item_count++] = new Rose(is_artificial,amount);
                        break;
                    case 'd':
                        orders[item_count++] = new Daisy(is_artificial);
                        break;
                    default:
                        cout << endl << "Wrong choice!" << endl;
                }
                break;
            case 'g':
                int selected_size;
                cout << endl << "Size? ([0] for small/ [1] for medium / [2] for large): ";
                cin >> selected_size;
                cout << endl << "Select subcategory ([f] for fruitbasket, [c] for cookiebasket): ";
                cin >> subcategory;
                switch(subcategory){
                    case 'f':
                        int selected_type;
                        cout << endl << "Type? ([0] for standard mix/ [1] for citrus mix /
                                                [2] for tropical mix): ";
                        cin >> selected_type;
                        bool with_sauce;
                        cout << endl << "Chocolate sauce? ([0] for no/ [1] for yes): ";
                        cin >> with_sauce;
                        orders[item_count++] = new FruitBasket(basketSize(selected_size),
                                                               fruitType(selected_type),with_sauce);
                        break;
                    case 'c':
                        orders[item_count++] = new CookieBasket(basketSize(selected_size));
                        break;
                    default:
                        cout << endl << "Wrong choice!" << endl;
                }
                break;
            case 'e':
                cout << endl << "Ending the shopping session" << endl;
                continue_shopping = false;
                break;
            default:
                cout << endl << "Wrong choice!" << endl;
        }
        cout << endl;
    }
    // Calculating price and checking promotions for the ordered itens and printing out
    if(item_count > 0){
        cout << endl << "ORDER LIST" << endl;
        cout << "--------------------------------------------------------------------------------" << endl;
    }
    for(int i=0; i<item_count; i++){
        orders[i]->calculate_price();
        orders[i]->print();
        cout << endl;
    }

    // giving allocated space back
    for(int i=0; i<item_count; i++)
        delete orders[i];

    return 0;
}
```

Expected Output:

```
--------------------------------------------------------------------------
Select category ([f] for flower, [g] for gourmet, [e] to end shopping): f
Fresh flower[0] or artificial flower[1]?: 1
Select subcategory ([r] for rose, [d] for daisy): r
Amount?: 5
--------------------------------------------------------------------------
Select category ([f] for flower, [g] for gourmet, [e] to end shopping): f
Fresh flower[0] or artificial flower[1]?: 0
Select subcategory ([r] for rose, [d] for daisy): d
--------------------------------------------------------------------------
Select category ([f] for flower, [g] for gourmet, [e] to end shopping): g
Size? ([0] for small/ [1] for medium / [2] for large): 1
Select subcategory ([f] for fruitbasket, [c] for cookiebasket): f
Type? ([0] for standard mix/ [1] for citrus mix / [2] for tropical mix): 2
Chocolate sauce? ([0] for no/ [1] for yes): 1
--------------------------------------------------------------------------
Select category ([f] for flower, [g] for gourmet, [e] to end shopping): g
Size? ([0] for small/ [1] for medium / [2] for large): 2
Select subcategory ([f] for fruitbasket, [c] for cookiebasket): c
--------------------------------------------------------------------------
Select category ([f] for flower, [g] for gourmet, [e] to end shopping): e
Ending the shopping session

ORDER LIST
--------------------------------------------------------------------------
5 roses (artificial) - 75 TL
A bunch of daisies - 20 TL
Tropical fruit basket with chocalate sauce (medium) - 120 TL
Promotion: a bunch of daisies for free
A basket of cookies (large) - 70 TL
Promotion: a bunch of daisies for free
```

Submission Procedure:

1. Make sure that GNU C++ Compiler (g++) compiles your project and the application runs in Unix smoothly. This is important because we will evaluate your homework in Unix using g++.
2. Use comments wherever necessary in your code to explain what you did.
3. After you make sure that everything is compiled smoothly, archive all files into a zip file. Submit this file through www.ninova.itu.edu.tr. Ninova enables you to change your submission before the submission deadline.

Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions. Any student found guilty will get grade F.

Note: If you face with an anomaly with the assignment or given test program, contact me (responsible research assistant from this assignment) as soon as possible via e-mail (ersenm@itu.edu.tr) or in person (office no: 4308).