San Francisco Bay University

CS360 - Programming in C and C++
Homework Assignment #5

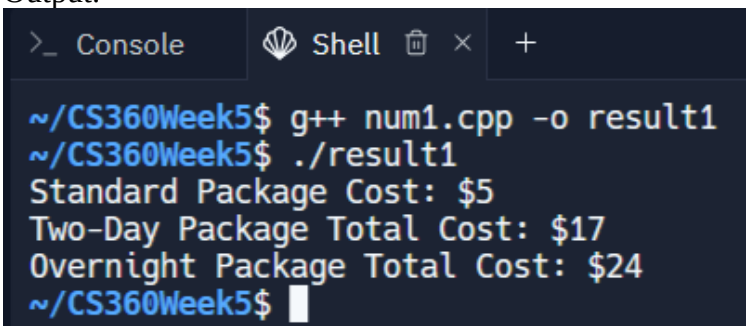
Due day: 4/04/2024

Kritika Regmi
ID:19702

1. Package-delivery services, such as FedEx®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use class *Package* as the base class of the hierarchy, then include classes *TwoDayPackage* and *OvernightPackage* that derive from *Package*.

   Base class *Package* should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. *Package*'s constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. *Package* should provide a *public* member function *calculateCost* that returns a *double* indicating the cost associated with shipping the package. *Package's calculateCost* function should determine the cost by multiplying the weight by the cost per ounce. Derived class *TwoDayPackage* should inherit the functionality of base class *Package*, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. *TwoDayPackage's* constructor should receive a value to initialize this data member. *TwoDayPackage* should redefine member function *calculateCost* so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class *Package's calculateCost* function. Class *OvernightPackage* should inherit directly from class *Package* and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. *OvernightPackage* should redefine member function *calculateCost* so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a *main* program that creates objects of each type of *Package* and tests member function *calculateCost*.
   ANSWER:

   Output:



CODE:

```cpp
#include <iostream>
#include <string>
#include <stdexcept>

using namespace std;

class Package {
protected:
    // Sender's information
    string sName;
    string sAddress;
    string sCity;
    string sState;
    string sZipCode;

    // Receiver's information
    string rName;
    string rAddress;
    string rCity;
    string rState;
    string rZipCode;

    double weightInOunces; // Weight of the package in ounces
    double costPerOunce; // Cost per ounce for shipping
```

```cpp
public:
    // Constructor to initialize package details
    Package(const string& sName, const string& sAddress, const string& sCity,
            const string& sState, const string& sZipCode, const string& rName,
            const string& rAddress, const string& rCity, const string& rState,
            const string& rZipCode, double weightInOunces, double costPerOunce)
            : sName(sName), sAddress(sAddress), sCity(sCity), sState(sState),
            sZipCode(sZipCode), rName(rName), rAddress(rAddress), rCity(rCity),
            rState(rState), rZipCode(rZipCode), weightInOunces(weightInOunces),
            costPerOunce(costPerOunce) {
        if (weightInOunces <= 0 || costPerOunce <= 0) {
            throw invalid_argument("Weight and cost per ounce must be positive values.");
        }
    }

    // Virtual function to calculate shipping cost
    virtual double calculateCost() const { return weightInOunces * costPerOunce; }
};
```

```cpp
class TwoDayPackage : public Package {
private:
    double flatFee; // Additional flat fee for two-day shipping

public:
    // Constructor to initialize two-day package details
    TwoDayPackage(const string& sName, const string& sAddress,
                  const string& sCity, const string& sState,
                  const string& sZipCode, const string& rName,
                  const string& rAddress, const string& rCity,
                  const string& rState, const string& rZipCode,
                  double weightInOunces, double costPerOunce, double flatFee)
        : Package(sName, sAddress, sCity, sState, sZipCode, rName, rAddress,
                  rCity, rState, rZipCode, weightInOunces, costPerOunce),
          flatFee(flatFee) {}

    // Override function to calculate total cost including flat fee
    double calculateCost() const override {
        return Package::calculateCost() + flatFee;
    }
};
class OvernightPackage : public Package {
private:
    double additionalFeePerOunce; // Additional fee per ounce for overnight shipping

public:
    // Constructor to initialize overnight package details
    OvernightPackage(const string& sName, const string& sAddress,
                     const string& sCity, const string& sState,
                     const string& sZipCode, const string& rName,
                     const string& rAddress, const string& rCity,
                     const string& rState, const string& rZipCode,
                     double weightInOunces, double costPerOunce,
                     double additionalFeePerOunce)
        : Package(sName, sAddress, sCity, sState, sZipCode, rName, rAddress,
                  rCity, rState, rZipCode, weightInOunces, costPerOunce),
          additionalFeePerOunce(additionalFeePerOunce) {}

    // Override function to calculate total cost including additional fee per ounce
    double calculateCost() const override {
        return (weightInOunces * costPerOunce) +
               (weightInOunces * additionalFeePerOunce);
    }
};
```

```cpp
int main() {
    try {
        // Create instances of Package, TwoDayPackage, and OvernightPackage
        Package package("John Doe", "123 Main St", "Anytown", "AnyState", "12345",
                        "Jane Smith", "456 Oak St", "Othertown", "OtherState", "67890",
                        10.0, 0.5);
        TwoDayPackage twoDayPackage("Alice Jones", "789 Elm St", "Somewhere", "SomeState", "54321",
                                    "Bob Brown", "321 Pine St", "Nowhere", "NoState", "09876",
                                    15.0, 0.6, 8.0);
        OvernightPackage overnightPackage("Eva Green", "567 Maple St", "Anywhere", "AnyState", "24680",
                                          "David White", "654 Birch St", "Nowhere", "NoState", "13579",
                                          20.0, 0.7, 0.5);

        // Display shipping costs for each package
        cout << "Standard Package Cost: $" << package.calculateCost() << endl;
        cout << "Two-Day Package Total Cost: $" << twoDayPackage.calculateCost() << endl;
        cout << "Overnight Package Total Cost: $" << overnightPackage.calculateCost() << endl;
    }
    catch (const invalid_argument& e) {
        cerr << "Error: " << e.what() << endl;
    }

    return 0;
}
```

2. A supermarket chain has asked you to develop an automatic checkout system. All products are identifiable by means of a barcode and the product name. Groceries are either sold in packages or by weight. Packed goods have fixed prices. The price of groceries sold by weight is calculated by multiplying the weight by the current price per kilo. Develop the classes needed to represent the products first and organize them hierarchically. The *Product* class, which contains generic information on all products (barcode, name, etc.), can be used as a base class.

    a. The *Product* class contains two data members of type *long* used for storing barcodes and the product name. Define a constructor with parameters for both data members. Add default values for the parameters to provide a default constructor for the class. In addition to the access methods *setCode()* and *getCode()*, also define the methods *scanner()* and *printer()*. For test purposes, these methods will simply output product data on screen or read the data of a product from the keyboard.

```cpp
class Product {
private:
    long productCode; // Barcode of the product
    string productName;

public:
    // Constructor with default values
    Product(long code = 0, const string& name = "Unknown") : productCode(code), productName(name) {}

    // Setter and getter for product code
    void setProductCode(long code) { productCode = code; }
    long getProductCode() const { return productCode; }

    // Virtual method to scan product details
    virtual void scanner() const {
        cout << "Scanning Product:" << endl;
        cout << "Product Code: " << productCode << endl;
        cout << "Product Name: " << productName << endl;
    }

    // Virtual method to print product details
    virtual void printer() const {
        cout << "Product Details:" << endl;
        cout << "Product Code: " << productCode << endl;
        cout << "Product Name: " << productName << endl;
    }
};
```

b. The next step involves developing special cases of the *Product* class. Define two classes derived from *Product*, *PrepackedFood* and *FreshFood*. In addition to the product data, the *PrepackedFood* class should contain the unit price and the *FreshFood* class should contain a weight and a price per kilo as data members.

In both classes define a constructor with parameters providing default-values for all data members. Use both the base and member initializer.

Define the access methods needed for the new data members. Also redefine the methods *scanner()* and *printer()* to take the new data members into consideration.

```cpp
class PrepackedFood : public Product {
private:
    double unitPrice; // Price per unit for prepacked food items

public:
    // Constructor with default values
    PrepackedFood(long code = 0, const string& name = "Unknown", double price = 0.0)
        : Product(code, name), unitPrice(price) {}

    // Setter and getter for unit price
    void setUnitPrice(double price) { unitPrice = price; }
    double getUnitPrice() const { return unitPrice; }

    // Overridden scanner method to include unit price
    void scanner() const override {
        Product::scanner();
        cout << "Unit Price: $" << unitPrice << endl;
    }

    // Overridden printer method to include unit price
    void printer() const override {
        Product::printer();
        cout << "Unit Price: $" << unitPrice << endl;
    }
};
class FreshFood : public Product {
private:
    double weight; // Weight of fresh food items in kilograms
    double pricePerKg; // Price per kilogram for fresh food items

public:
    // Constructor with default values
    FreshFood(long code = 0, const string& name = "Unknown", double weightVal = 0.0, double pricePerKgVal = 0.0)
        : Product(code, name), weight(weightVal), pricePerKg(pricePerKgVal) {}

    // Setter and getter for weight
    void setWeight(double w) { weight = w; }
    double getWeight() const { return weight; }

    // Setter and getter for price per kilogram
    void setPricePerKg(double price) { pricePerKg = price; }
    double getPricePerKg() const { return pricePerKg; }

    // Overridden scanner method to include weight and price per kilogram
    void scanner() const override {
        Product::scanner();
        cout << "Weight: " << weight << " kg" << endl;
        cout << "Price per Kilogram: $" << pricePerKg << endl;
    }
```

```
84
85      // Overridden printer method to include weight and price per kilogram
86      void printer() const override {
87          Product::printer();
88          cout << "Weight: " << weight << " kg" << endl;
89          cout << "Price per Kilogram: $" << pricePerKg << endl;
90      }
91  };
```

c. Test the various classes in a *main* function that creates two objects each of the types Product, *PrepackedFood* and *FreshFood*. One object of each type is fully initialized in the object definition. Use the default constructor to create the other object. Test the *get* and *set* methods and the *scanner()* method and display the products on screen.

```
93  int main() {
94      // Testing the classes
95      Product prod1(123456789, "Yogurt");
96      Product prod2;
97      prod2.setProductCode(987654321);
98      prod2.printer();
99
100     PrepackedFood packed1(111222333, "Bagels", 2.99);
101     PrepackedFood packed2;
102     packed2.setProductCode(444555666);
103     packed2.setUnitPrice(1.49);
104     packed2.printer();
105
106     FreshFood fresh1(777888999, "Banana", 2.5, 2.99);
107     FreshFood fresh2;
108     fresh2.setProductCode(333444555);
109     fresh2.setWeight(1.8);
110     fresh2.setPricePerKg(3.49);
111     fresh2.printer();
112
113     packed1.scanner();
114     fresh1.scanner();
115
116     return 0;
117 }
```

Output:

```
Product Name: Bagels
Unit Price: $2.99
Scanning Product:
Product Code: 777888999
Product Name: Banana
Weight: 2.5 kg
Price per Kilogram: $2.99
```