**Assignment-5 CS360L.**

**#Question-1:**

#include <iostream>

#include <string>

using namespace std;

// Base class Package

class Package {

protected:

string senderName;

string senderAddress;

string senderCity;

string senderState;

string senderZIP;

string recipientName;

string recipientAddress;

string recipientCity;

string recipientState;

string recipientZIP;

double weight; // in ounces

double costPerOunce; // cost per ounce in dollars

public:

// Constructor

Package(const string &senderName, const string &senderAddress, const string &senderCity, const string &senderState, const string &senderZIP,

const string &recipientName, const string &recipientAddress, const string &recipientCity, const string &recipientState, const string &recipientZIP,

double weight, double costPerOunce)

: senderName(senderName), senderAddress(senderAddress), senderCity(senderCity), senderState(senderState), senderZIP(senderZIP),

recipientName(recipientName), recipientAddress(recipientAddress), recipientCity(recipientCity), recipientState(recipientState), recipientZIP(recipientZIP),

weight(weight), costPerOunce(costPerOunce) {

if (weight <= 0 || costPerOunce <= 0) {

cerr << "Error: Weight and cost per ounce must be positive values." << endl;

exit(1);

}

}

// Calculate shipping cost

double calculateCost() const {

return weight \* costPerOunce;

}

};

// Derived class TwoDayPackage

class TwoDayPackage : public Package {

private:

double flatFee;

public:

// Constructor

TwoDayPackage(const string &senderName, const string &senderAddress, const string &senderCity, const string &senderState, const string &senderZIP,

const string &recipientName, const string &recipientAddress, const string &recipientCity, const string &recipientState, const string &recipientZIP,

double weight, double costPerOunce, double flatFee)

: Package(senderName, senderAddress, senderCity, senderState, senderZIP,

recipientName, recipientAddress, recipientCity, recipientState, recipientZIP,

weight, costPerOunce), flatFee(flatFee) {}

// Calculate shipping cost including flat fee

double calculateCost() const {

return Package::calculateCost() + flatFee;

}

};

// Derived class OvernightPackage

class OvernightPackage : public Package {

private:

double additionalFeePerOunce;

public:

// Constructor

OvernightPackage(const string &senderName, const string &senderAddress, const string &senderCity, const string &senderState, const string &senderZIP,

const string &recipientName, const string &recipientAddress, const string &recipientCity, const string &recipientState, const string &recipientZIP,

double weight, double costPerOunce, double additionalFeePerOunce)

: Package(senderName, senderAddress, senderCity, senderState, senderZIP,

recipientName, recipientAddress, recipientCity, recipientState, recipientZIP,

weight, costPerOunce), additionalFeePerOunce(additionalFeePerOunce) {}

// Calculate shipping cost including additional fee per ounce

double calculateCost() const {

return weight \* (costPerOunce + additionalFeePerOunce);

}

};

int main() {

// Create objects of each type of Package and test calculateCost member function

Package regularPackage("John Doe", "123 Main St", "Anytown", "CA", "12345",

"Jane Smith", "456 Elm St", "Sometown", "NY", "67890",

16.5, 0.5);

TwoDayPackage twoDayPackage("John Doe", "123 Main St", "Anytown", "CA", "12345",

"Jane Smith", "456 Elm St", "Sometown", "NY", "67890",

16.5, 0.5, 5.0);

OvernightPackage overnightPackage("John Doe", "123 Main St", "Anytown", "CA", "12345",

"Jane Smith", "456 Elm St", "Sometown", "NY", "67890",

16.5, 0.5, 2.0);

cout << "Regular Package Cost: $" << regularPackage.calculateCost() << endl;

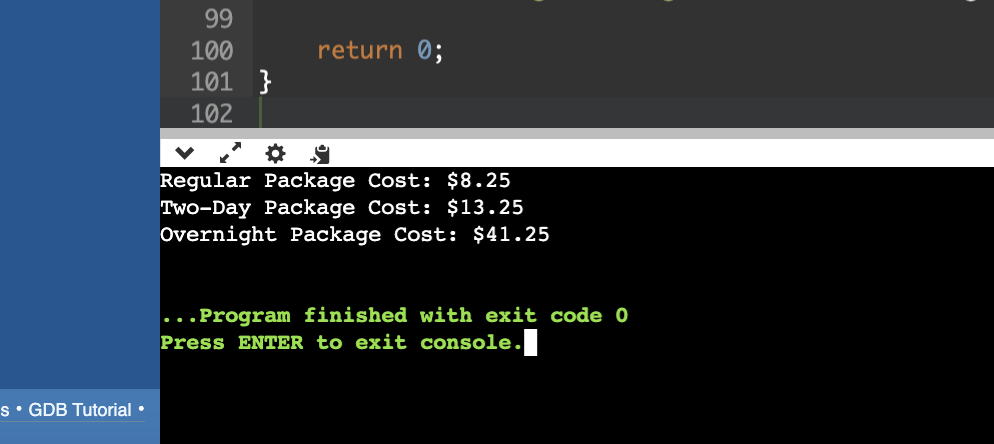
cout << "Two-Day Package Cost: $" << twoDayPackage.calculateCost() << endl;

cout << "Overnight Package Cost: $" << overnightPackage.calculateCost() << endl;

return 0;

}

Output:



**#Question-2:**

#include <iostream>

#include <string>

class Product {

protected:

long barcode;

std::string name;

public:

Product(long barcode = 0, std::string name = "") : barcode(barcode), name(name) {}

void setCode(long barcode) {

this->barcode = barcode;

}

long getCode() const {

return barcode;

}

virtual void scanner() {

std::cout << "Enter barcode for the product: ";

std::cin >> barcode;

std::cout << "Enter name for the product: ";

std::cin.ignore();

std::getline(std::cin, name);

}

virtual void printer() const {

std::cout << "Product Details:\nBarcode: " << barcode << "\nName: " << name << '\n';

}

};

class PrepackedFood : public Product {

private:

double unitPrice;

public:

PrepackedFood(long barcode = 0, std::string name = "", double unitPrice = 0.0)

: Product(barcode, name), unitPrice(unitPrice) {}

void setUnitPrice(double unitPrice) {

this->unitPrice = unitPrice;

}

double getUnitPrice() const {

return unitPrice;

}

void scanner() override {

Product::scanner();

std::cout << "Enter unit price for the prepacked food: ";

std::cin >> unitPrice;

}

void printer() const override {

Product::printer();

std::cout << "Unit Price: " << unitPrice << '\n';

}

};

class FreshFood : public Product {

private:

double weight;

double pricePerKilo;

public:

FreshFood(long barcode = 0, std::string name = "", double weight = 0.0, double pricePerKilo = 0.0)

: Product(barcode, name), weight(weight), pricePerKilo(pricePerKilo) {}

void setWeight(double weight) {

this->weight = weight;

}

double getWeight() const {

return weight;

}

void setPricePerKilo(double pricePerKilo) {

this->pricePerKilo = pricePerKilo;

}

double getPricePerKilo() const {

return pricePerKilo;

}

void scanner() override {

Product::scanner();

std::cout << "Enter weight for the fresh food: ";

std::cin >> weight;

std::cout << "Enter price per kilo for the fresh food: ";

std::cin >> pricePerKilo;

}

void printer() const override {

Product::printer();

std::cout << "Weight: " << weight << "\nPrice per Kilo: " << pricePerKilo << '\n';

}

};

int main() {

Product product1(123456, "Product1");

PrepackedFood prepackedFood1(234567, "PrepackedFood1", 10.0);

FreshFood freshFood1(345678, "FreshFood1", 2.0, 5.0);

Product product2;

PrepackedFood prepackedFood2;

FreshFood freshFood2;

std::cout << "\n--- Scanning Products ---\n";

product2.scanner();

prepackedFood2.scanner();

freshFood2.scanner();

std::cout << "\n--- Printing Products ---\n";

product1.printer();

prepackedFood1.printer();

freshFood1.printer();

product2.printer();

prepackedFood2.printer();

freshFood2.printer();

return 0;

}

**Output:**

