|  |
| --- |
| **Computer Engineering Department - ITU** |
| **CE101L: Object Oriented Programming Lab** |

|  |  |
| --- | --- |
| **Course Instructor: Usama Bin Shakeel** | **Dated: 01/06/2022** |
| **Teaching Assistant: Aqsa Khalid** | **Semester: Spring 2022** |
| **Lab Engineer: Nadir Abbas** | **Batch: BSCE2021** |

# **Lab 12A. Problem Based Learning through Open Ended Questions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Roll number** | **Report**  **(out of 100)** | **Scaled to 10** | **Total**  **(out of 10)** |
| Muhammad Abubakar Saif | BSCE21017 |  |  |  |

Checked on: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **Objective**

The objective of this lab is to observe the basic knowledge of programming classes in C++.

## **Equipment and Component**

|  |  |  |
| --- | --- | --- |
| **Component Description** | **Value** | **Quantity** |
| Computer | Available in lab | 1 |

## **Conduct of Lab**

1. Students are required to perform this experiment individually.
2. In case the lab experiment is not understood, the students are advised to seek help from the course instructor, lab engineers, assigned teaching assistants (TA) and lab attendants.

## **Theory and Background**

**Open-ended problem** is a problem that has several or many correct answers, and several ways to the correct answer(s). The Open-Ended Approach provides students with "experience in finding something new in the process"(Shimada 1997). It is basically facilitating the development of creative problem solving skills.

Diagram

Description automatically generated

Figure 1: \*What is Open Ended Problem Solving??

**Lab Task**

**Task A: [Marks: 40]**

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 test program that creates objects of each type of Package and tests member function calculateCost.

|  |
| --- |
| (Header File):  #include **<string>  class** Person { **public**:  std::string name, address, city, state;  **int** ZIP;  **void** setter(); };  **class** Package { **private**:  **double** weight; *//weight in ounces* **static double** costPerOunce; **protected**:  Person Sender, Recipient; **public**:  Package();  **double** calculateCost(); };  **class** TwoDayPackage : **public** Package { **private**:  **static double** flat\_fee; **public**:  TwoDayPackage();  **double** calculateCost(); };  **class** OvernightPackage : **public** Package { **private**:  **static double** fee\_per\_ounce; **public**:  OvernightPackage();  **double** calculateCost(); };  (Courier.cpp):  #include **"Courier.h"** #include **<iostream>  using namespace** std;  **void** Person::setter() {  cout << **"Enter Name: "**;  cin.ignore();  cin >> name;  cout << **"Enter Address: "**;  cin >> address;  cout << **"Enter City : "**;  cin >> city;  cout << **"Enter Province/State : "**;  cin >> state;  cout << **"Enter ZIP Code : "**;  cin >> ZIP; }  Package::Package() {  **do** {  cout << **"Enter the Weight of the Package (in ounces): \n"**;  cin >> weight;  } **while** (weight < 0 &&  cout << **"Real Life Error, Weight of Package cannot be set to a negative value. Try Again \n"**); }  **double** Package::calculateCost() { *// cout<<"Base Class \n";* **double** cost;  cost = weight \* costPerOunce;  **return** cost; }  **double** TwoDayPackage::calculateCost() { *// cout<<"Child Class \n";* **double** sum = Package::calculateCost() + flat\_fee;  **return** sum; }  **double** OvernightPackage::calculateCost() {  **double** sum = Package::calculateCost() + fee\_per\_ounce;  **return** sum; }  TwoDayPackage::TwoDayPackage() {  cout << **"\nEnter the Details of Sender: \n"**;  Sender.setter();  cout << **"Enter the Details of Recipient: \n"**;  Recipient.setter(); }  OvernightPackage::OvernightPackage() {  cout << **"\nEnter the Details of Sender: \n"**;  Sender.setter();  cout << **"Enter the Details of Recipient: \n"**;  Recipient.setter(); }  (MAIN FUNCTION):  #include **<iostream>** #include **"Courier.h"  using namespace** std;  **double** Package::costPerOunce = 0.20; *//initialization of static members* **double** TwoDayPackage::flat\_fee = 6.95; *//initialization of static members* **double** OvernightPackage::fee\_per\_ounce = 8.90; *//initialization of static members* **int** main() {  **int** choice; *//declaring integer variable* string rep; *//declaring string variable* again: *//"GOTO" label* cout << **"Welcome to TRANSPAK® Couriers \n"**;  cout << **"Select your Package Type: \n"**;  cout << **"1. Two Day Package: \n"**;  cout << **"2. Overnight Package: \n"**;  **do** {  cout << **"Enter your choice: "**;  cin >> choice; *//takes input from the user* } **while** (choice < 1 || choice > 2);  **switch** (choice) { *//switch the cases based upon user choice* **case** 1: {  TwoDayPackage parcel; *//declare instance manually declared class* cout << **"Your Total Shipment Charges are: $"** << parcel.calculateCost(); *//member function calling* }  **break**;  **case** 2: {  OvernightPackage nightMail; *//declare instance manually declared class* cout << **"Your Total Shipment Charges are: $"** << nightMail.calculateCost()<<**"\n"**; *//member function calling* }  **break**;  }  cout << **"Please pay as soon as possible, so that your parcel will be dispatched to your destination \n"**;  cout << **"Do you want to use the program again? (Y/N): "**;  cin >> rep; *//takes input from user* **if** (rep == **"N" or** rep == **"n" or** rep == **"no" or** rep == **"NO" or** rep == **"No" or** rep == **"nO"**) {  **return** 0; *//exits the process with exit code 0* } **else goto** again; *//restart the program flow* } |

**Output:**

**Two Day Package:**

Text

Description automatically generated

**Overnight Package:**

Text

Description automatically generated

#### **Assessment Rubric for Lab**

**Method for assessment:**

Lab reports and instructor observation during lab sessions. Outcome assessed:

a. Ability to conduct experiments, as well as to analyze and interpret data (P) b. Ability to function on multi-disciplinary teams (A)

c. Ability to use the techniques, skills, and modern engineering tools necessary for engineering practice (P)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance metric** | **Task** | **CLO** | **Description** | **Max marks** | **Exceeds expectation** | **Meets expectation** | **Does not meet expectation** | **Obtained marks** |
| 1. Realization of experiment (a) | 1 | 1 | Functionality | 40 | Executes without errors excellent user prompts, good use of symbols, spacing in output. Through testing has been completed (35-40) | Executes without errors, user prompts are understandable, minimum use of symbols or spacing in output. Some testing has been completed (20-34) | Does not execute due to syntax errors, runtime errors, user prompts are misleading or non-existent. No testing has been completed (0-19) |  |
| 2. Teamwork (b) | 1 | 3 | Group Performance | 5 | Actively engages and cooperates with other group member(s) in effective manner (4-5) | Cooperates with other group member(s) in a reasonable manner but conduct can be improved (2-3) | Distracts or discourages other group members from conducting the experiment (0-1) |  |
| 3. Conducting experiment (a, c) | 1 | 1 | On Spot Changes | 10 | Able to make changes (8-10) | Partially able to make changes (5-7) | Unable to make changes (0-4) |  |
| 1 | 1 | Viva | 10 | Answered all questions (8-10) | Few incorrect answers (5-7) | Unable to answer all questions (0-4) |  |
| 4. Laboratory safety and disciplinary rules (a) | 1 | 3 | Code commenting | 5 | Comments are added and does help the reader to understand the code (4-5) | Comments are added and does not help the reader to understand the code (2-3) | Comments are not added (0-1) |  |
| 5. Data collection (c) | 1 | 3 | Code Structure | 5 | Excellent use of white space, creatively organized work, excellent use of variables and constants, correct identifiers for constants, No line-wrap (4-5) | Includes name, and assignment, white space makes the program fairly easy to read. Title, organized work, good use of variables (2-3) | Poor use of white space (indentation, blank lines) making code hard to read, disorganized and messy (0-1) |  |
| 6. Data analysis (a, c) | 1 | 4 | Algorithm | 20 | Solution is efficient, easy to understand, and maintain (15-20) | A logical solution that is easy to follow but it is not the most efficient (6-14) | A difficult and inefficient solution (0-5) |  |
| 7. Computer use (c) | 1 | 2 | Documentation & GitHub Submissions | 5 | Timely (4-5) | Late (2-3) | Not done (0-1) |  |
|  | Max Marks (total): | | | 100 | Obtained Marks (total): | | |  |

Lab Engineer Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_