Object Oriented Paradigm

Session: Fall 2022

Faculty of Information Technology

UCP Lahore, Pakistan

Lab 4: Setters are Getters Methods, Separating Interface and Implementation

Note: From now onwards, always separate the interface from implementation (that is, always create header files)

Instructions:

- Indent your code
- Comment your code
- Use meaningful variable names
- Plan your code carefully on a piece of paper before you implement it.
- Name of the program should be same as the task name. i.e. the first program should be Task_4_01.cpp

Note: In this lab, all data members must be private

Lab Tasks:

Lab Task 1:

Design and implement a class dayType that implements the day of the week in a program. The class dayType should store the day, such as Sun for Sunday. Write setter and getter functions of the class dayType. Also add member functions to perform the following operations on an object of type dayType:

a) Print the day.
c) Return the next day.
e) Return the previous day.
f) Calculate and return the day by adding certain days to the current day. For example, if the current day is Monday and we add 4 days, the day to be returned is Friday. Similarly, if today is Tuesday and we add 13 days, the day to be returned is Monday.

Lab Task 2:

Write a C++ program to represent a bank account using class and objects. Display the details such as name, account number, account type and balance in the given account number. Add setter and getter functions. User can deposit and withdrawn money from the account.

Lab Task 3:

Create a class for traffic signal. use an object of the class to turn the green , red and yellow signals on and off.

Lab Task 4:

An automobile company has serial number for engine parts starting from AA0 to FF9. The other characteristics of parts to be specified as attributes are: Year of manufacture, material and quantity manufactured.

Write getters and setters for each characteristic. Make three objects and set the values of these characteristics through setters and display them through getters.

Lab Task 5:

A class named “employee” holds information like employee code, name, date of joining. Write a program to create three objects of employee and enter some data into it through setters. Make getters and setters for all employee information. Then ask the user to enter current date. Display the names of those employees whose tenure is 2 or more than 2 years according to the given current date only using getters.

Task 6:

Write the definition for a class named **`GasPump`** used to model a pump at an automobile service station. Before you go further with this programming exercise, write down the behavior you expect from a gas pump from the point of view of the purchaser. The following are listed things a gas pump might be expected to do. If your list differs, and you think your list is as good as or better than these, then consult your instructor. You and your instructor should jointly decide what behavior you are to implement. Then implement and test the agreed upon design for a gas pump class.

- A display of the amount dispensed
- A display of the amount charged for the amount dispensed
- A display of the cost per liter
- Before use, the gas pump must reset the amount dispensed and amount charged to zero.
- Once started, a gas pump continues to dispense fuel, keep track of the amount dispensed, and compute the charge for the amount dispensed until stopped.
- A stop dispensing control of some kind is needed.

Implement the behavior of the gas pump as declarations of member functions of the gas pump class, and then write implementations of these member functions.

Task 7:

Write a program to simulate a bookstore. The bookstore has two types of customers:

- Those who are members of the bookstore and
- Those who are non-members and buy books from the bookstore only occasionally

Each member has to pay a $10 yearly membership fee and receives a 5% discount on each book purchased. For each member, the bookstore keeps track of the number of books purchased and the total amount spent. For every sixth book that a member buys, the bookstore takes the average of the total amount of the last 5 books purchased, applies this amount as a discount, and then resets the total amount spent to 0. Write a program that can process up to 10 book titles and 5 members. Your program should contain a menu that gives the user different choices to effectively run the program; in other words, your program should be user driven.

Task 8:

The equation of a line in standard form is ax + by = c, where in both a andb cannot be zero, and a, b, and c are real numbers. If b ≠ 0, then –a/b is the slope of the line. If a = 0, then it is a horizontal line, and if b = 0, then it is a vertical line. The slope of a vertical line is undefined. Two lines are parallel if they have the same slope or both are vertical lines. Two lines are perpendicular if either one of the lines is horizontal and the other is vertical or the product of their slopes is –1.

Design the class lineTypeto store a line. To store a line, you need to store the values of a (coefficient of x), b (coefficient of y), and c. Your class must contain the following operations.

a. If a line is non-vertical, then determine its slope.
b. Determine if two lines are equal. (Two lines $a_1x + b_1y = c_1$ and $a_2x + b_2y = c_2$ are equal if either $a_1 = a_2$, $b_1 = b_2$, and $c_1 = c_2$ or $a_1 = ka_2$, $b_1 = kb_2$, and $c_1 = kc_2$ for some real number k.)
c. Determine if two lines are parallel.
d. Determine if two lines are perpendicular.
e. If two lines are not parallel, then find the point of intersection.

Task 9:

Define a class called **`Pizza`** that has member variables to track the type of pizza (either deep dish, hand tossed, or pan) along with the size (either small, medium, or large) and the number of pepperoni or cheese toppings. Include mutator and accessor functions for your class. Create a function that outputs a textual description of the pizza object. Also include a function that computes the cost of the pizza and returns it as a double according to the following rules:

Small pizza = $10 + $2 per topping

Medium pizza = $14 + $2 per topping

Large pizza = $17 + $2 per topping

Write a suitable test program that creates and outputs a description and price of various pizza objects.