CL-1004
Object Oriented Programming- Lab
Spring’ 2023
BS-SE

Lab Manual 05

Problem 1:

Build a class Sale with private member variables

- double itemCost; // Cost of the item
- int itemQuantity; //number of instance of item
- double taxRate; // Sales tax rate

and functionality mentioned below:

1. Write a default constructor to set the member variable itemCost to 0, itemQuantity to 0 and taxRate to 0.

   Sale( )

2. Write a parameterized constructor that accepts the parameter for each member variable such as cost for ItemCost, quantity for itemQuantity and rate for taxRate

   Sale(double cost, int quantity, double rate)

3. Generate getter setter for itemCost. itemQuantity and taxRate
4. Write a function getTax( ) to calculate tax i.e take a product of itemCost and itemRate.

   double getTax( )

5. Write a function getTotal( ) to calculate the total price of item i.e. take a sum of itemCost (keeping quantity in mind) and getTax( ) (calling getTax() will return the calculated tax on item)

   double getTotal( ).

Problem 2

1. Define a class Storage for storing integer values, regardless of the sequence and duplication checks. Your class should consist of the member variables.
   - int * items; // This member points to the dynamically allocated array of integers (which you will be allocating in constructor).
   - int capacity; // An integer that shows total capacity of your container
   - int counter; // A counter variable which increments upon every insertion operation; shows total number of elements inserted yet in container
2. You would need a parameterized constructor with single parameter int c; i.e. it initializes the capacity variable, showing the total capacity of the container, and also allocating memory to array. Set the value of counter to 0.
Storage(int c)
3. Function bool isFull( ), to check if the counter has reached to the max capacity of your array return true (counter = =capacity), else return false.
4. Function void insert(int val) takes single parameter, The function requires you to place that item in array, but before placing item, do call isFull( ) to check if we have not reached to the capacity. Update the counter variable as well.
5. Function bool search(int val) takes 1 parameter. Provided value has to be searched in array, note that array is not passed in the parameter list of this function; because it is already accessible i.e. array is a part of this class. Return true if you found the value.
6. Function void remove(int val) removes the value after searching it in array. But the most important thing about this function is, that suppose user provided a value which is placed at 3rd

index of array of size 5, so accomplishing the goal of removing value would require you to apply the logic of shifting the value from 4th index to 3rd and the value from 5th index to 4th. Don't forget to update counter.
7. Write a function void Print( ) to print the values of array.

Problem 3

Build a class CoffeeShots (representing a cup of coffee) having the following attributes

- type (string)
- price (double)
- volume (float)
- size (char)

1. Provide a parameterized constructor with arguments for type, price and volume. Provide a default value for the type parameter in constructor so that the user may omit providing this value when creating an instance. Initialize all data to the values provided in the argument list. However, since there is no argument for size, you will have to assign it a value by yourself. The size attribute will get a char value based on the following condition

- The size is 's' if the volume of the coffee is between 0 and 50 ml
- The size is 'm' if the volume is between 51 – 75 ml
- The size is 'l' if the volume is greater than 75 ml

2. Provide getters for each of these attributes, however, setter will be provided only for price.
3. Build a method upSize(). This method increases the volume of the coffee by 5ml and then resets the size accordingly so that the above conditions are still met. It also adds Rs.5 to the price of the coffee.
4. Provide another method spillOver(float) that takes the amount of coffee spilled (ml spilled) and then reduces the volume of coffee by that amount. For example if c is an instance of coffee and c.spillOver(3) is called, then it means that 3 ml of coffee is spilled and the volume gets reduced by this amount. Do not update the size or price.
5. Write a non-member function (not belonging to this class) createMyCoffee(). This method prompts the user for all the details required to create the coffee instance and creates a dynamic instance of the coffee with the provided data. It then returns a pointer to this coffee instance.
6. From main function, you just have to call createMyCoffee( ), calls to other functions must be handled inside this function.

Prototypes of the functions are:

- Coffeeshots(double p,float v,string t=0)
- void setPrice(double price)
- double getPrice( )
- float getVolume( )
- string getType( )
- char getSize( )
- void upSize( )

- float spillOver(float vol)
- void print( )
- Coffeeshots& createMyCoffee( )

Problem 4:

Create a class called Date that has

- int year,
- int month,
- int day

as data members.

1. Write getter setter functions for all the member variables.

2. The class should have a method to check if the date is a leap year.

   bool isLeapYear()

3. Write a method to calculate the number of days between two dates.

   int daysBetween(Date d)

Problem 5

Declare a class Block. A block as you all know is something a cubical container. It has following attributes

- Length
- Breadth
- Height

In addition to these, a block can be made of different materials e.g. wood, card, metal etc. Further, more a block can have different colors. Declare them also as member variables of class.
1. Provide a default Constructor, and a parameterized Constructor for the Block that takes all necessary values as arguments with the material as optional (if it is omitted the Block is considered to be made of Card – default value for the material).
2. Provide getters for all attributes and setters for each too except for the material (material of block cannot be changed after when it has been created!!!).
3. Provide a function getVolume() that calculates and returns the value of the volume of the Block.
4. Also provide another function getArea(), that calculates and returns the surface area of a Block.
Formula of Area = 2(lb + bh + lh)
5. Provide a print function that Prints the following about the Block

- Length:
- Breadth:
- Height:

- Material:
- Color:
- Volume:
- Area:

6. Inside main, allocate a block of memory for 4 objects (using array). Read the necessary values from the user to populate array.
7. Call functions Print() and Volume to display the data of blocks you just have saved in array.
8. Read the index and the new height from the user, ask the user to provide index of the block to change its height by creating a function update(Block b[], int size, int index, double height). Update the height and corresponding volume and area of the Block present on the index provided by user, save the updated height, volume, area right there, and print it again.