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
- double taxRate; // Sales tax rate

and functionality mentioned below:

1. Write a default constructor to set the member variable itemCost 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 and rate for taxRate
   Sale(double cost, double rate)
3. Generate only accessors for itemCost 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 and getTax( ) (calling getTax() will return the calculated tax on item) .
double getTotal( ).

Problem 2:

Create a class called BankAccount that has

- int balance,
- int interest

as a data member. Write getter setter for these variables.

1.The class should have methods to deposit money.

void deposit(double amount)

2. Another method is of withdraw money from the account.

void withdraw(double amount)

3. A method to calculate the interest earned on the balance over a certain period of time. The interest rate should be 0.01%.

double calculateInterest(int days)

4. A method that transfers the given amount from the current account to another BankAccount object specified by the other parameter.

void transfer(double amount, BankAccount& other)

Problem 3:

The absence of array bounds checking in C++ is a source of potential hazard. Write a class which will perform bounds checking on integer array.

Write a class IntegerList with private member variables as:

- int *list; // A pointer to an int . This member points to the dynamically allocated array of integers (which you will be allocating in constructor).
- int numElements; // An integer that holds the number of elements in the dynamically allocated array.

And public member functions

1. IntegerList(int) // The class constructor accepts an int argument that is the number of elements to allocate for the array. The array is allocated, and all elements are set to zero.
2.~IntegerList(); // Destructor to unallocated memory of list array.
3. bool isValid(int); // This function validates a subscript into the array. It accepts a subscript value as an argument and returns boolean true if the subscript is in the range 0 through numElements - 1. If the value is outside that range, boolean false is returned.
4. void setElement(int, int); // The setElement member function sets a specific element of the list array to a value. The first argument is the element subscript, and the second argument is the value to be stored in that element. The function uses isValid to validate the subscript. If subscript is valid, value is stored at that index, if an invalid subscript is passed to the function, the program aborts.
5. int getElement(int); // The getElement member function retrieves a value from a specific element in the list array. The argument is the subscript of the element whose value is to be retrieved. The function should use isValid function to validate the subscript. If the subscript is valid, the value is returned. If the subscript is invalid, the program returns -1.

Problem 4:

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

- Length
- Width
- 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, 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 getSurfaceArea(), that calculates and returns the surface area of a Block.
5. Provide a print function that Prints the following about the Block
    - Length:

- Width:
- Height:
- Material:
- Color:
- Volume:
- Surface Area:

6. Inside main, allocate a block of memory for 5 objects (using array). Read the necessary values from the user to populate array.
7. Call functions Print() and Volume to display the data of blockes 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 of the Block present on the index provided by user, save the updated height right there, and print it again.

Problem 5:

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( )