CL103 COMPUTER PROGRAMMING

LAB 05

<u>CLASSES, OBJECTS AND CONSTRUCTORS</u>

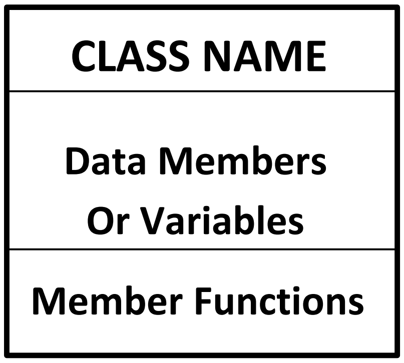
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CLASSES

A class is a programmer-defined data type that describes what an object of the class will look like when it is created. It consists of a set of variables and a set of functions.

Classes are created using the keyword **class**. A class declaration defines a new type that links code and data. This new type is then used to declare objects of that class.

In the UML, a class icon can be subdivided into compartments. The top compartment is for the name of the class, the second is for the variables of the class, and the third is for the methods of the class.



```
class class-name
{
access-specifier:
data

access-specifier:
functions

};
```

CLASS NAME
By convention, the name of a user-defined class begins with a capital letter and, for readability, each subsequent word in the class name begins with a capital letter.

DATA MEMBERS
Consider the attributes of some real world objects:

RADIO – station setting, volume setting.
CAR – speedometer readings, amount of gas in its tank and what gear it is in.

These attributes form the data in our program. The values that these attributes take (the blue color of the petals, for example) form the state of the object.

MEMBER FUNCTIONS
Consider the operations of some real world objects:

RADIO – setting its station and volume (invoked by the person adjusting the radio’s controls)
CAR – accelerating (invoked by the driver), decelerating, turning and shifting gears.

These operations form the functions in program. Member functions define the class’s behaviors.

OBJECTS

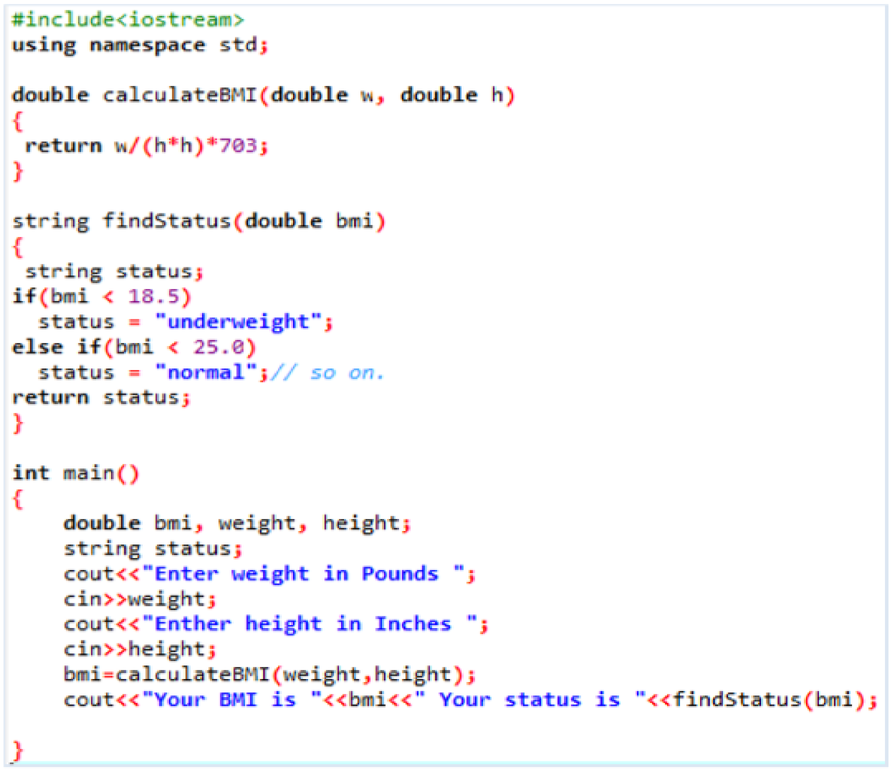
In C++, when we define a variable of a class, we call it **instantiating** the class. The variable itself is called an **instance** of the class. A variable of a class type is also called an **object**. Instantiating a variable allocates memory for the object.

RADIO r;
CAR c

STRUCTURES VS. CLASSES

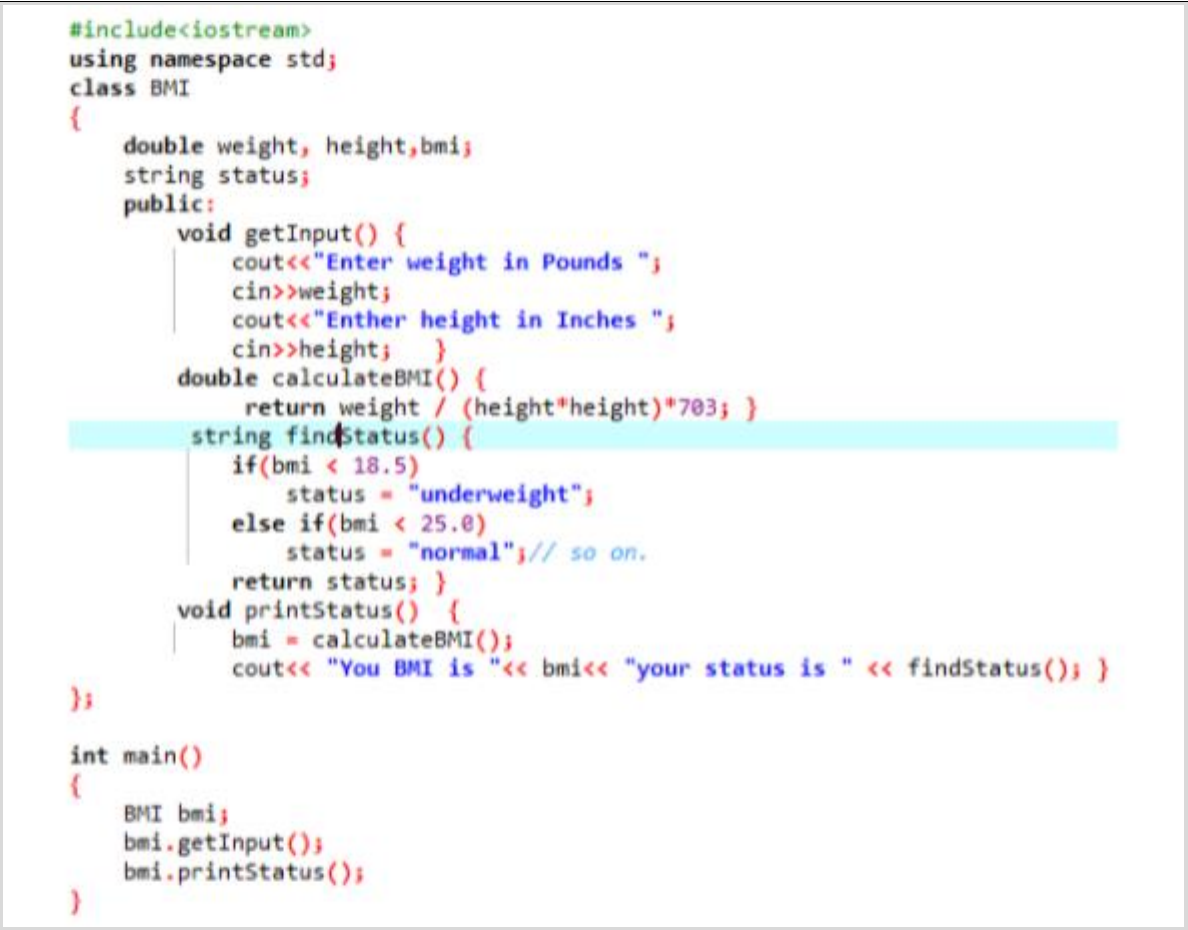
By default, all structure fields are public, or available to functions (like the main() function) that are outside the structure. Conversely, all class fields are private. That means they are not available for use outside the class. When you create a class, you can declare some fields to be private and some to be public. For example, in the real world, you might want your name to be public knowledge but your Social Security number, salary, or age to be private.

TRANSFORMATION FROM PROCEDURAL TO OBJECT ORIENTED PROGRAMMING



```
#include<iostream>
using namespace std;

double calculateBMI(double w, double h)
{
 return w/(h*h)*703;
}

string findStatus(double bmi)
{
 string status;
if(bmi < 18.5)
  status = "underweight";
else if(bmi < 25.0)
  status = "normal";// so on.
return status;
}

int main()
{
    double bmi, weight, height;
    string status;
    cout<<"Enter weight in Pounds ";
    cin>>weight;
    cout<<"Enther height in Inches ";
    cin>>height;
    bmi=calculateBMI(weight,height);
    cout<<"Your BMI is "<<bmi<<" Your status is "<<findStatus(bmi);

}
```

PROCEDURE ORIENTED APPROACH

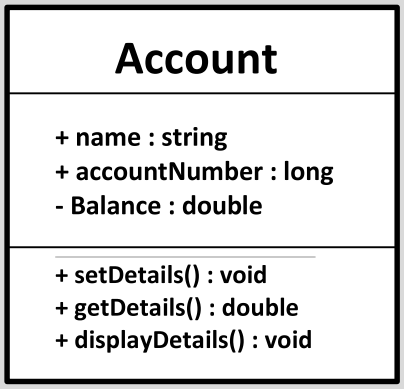


```
#include<iostream>
using namespace std;
class BMI
{
    double weight, height,bmi;
    string status;
    public:
        void getInput() {
            cout<<"Enter weight in Pounds ";
            cin>>weight;
            cout<<"Enther height in Inches ";
            cin>>height;    }
        double calculateBMI() {
             return weight / (height*height)*703; }
         string findStatus() {
            if(bmi < 18.5)
                status = "underweight";
            else if(bmi < 25.0)
                status = "normal";// so on.
            return status; }
        void printStatus()  {
            bmi = calculateBMI();
            cout<< "You BMI is "<< bmi<< "your status is " << findStatus(); }
};

int main()
{
    BMI bmi;
    bmi.getInput();
    bmi.printStatus();
}
```

OBJECT ORIENTED APPROACH

EXAMPLE PROGRAM

```
#include<iostream>

using namespace std;

class Account
{
private:
        double balance; // Account balance

public: //Public interface
        string name; // Account holder
         long accountNumber; // Account number
```

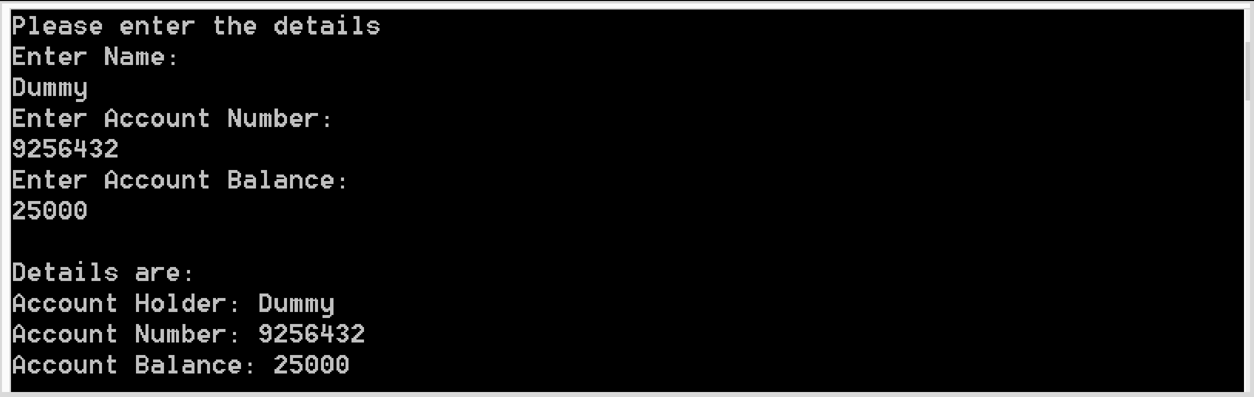


```
void setDetails(double bal)
{
        balance = bal;
}
double getDetails()
{
        return balance;
}
void displayDetails()
{
        cout<<"Details are: "<<endl;
        cout<<"Account Holder: "<<name<<endl;
        cout<<"Account Number: "<< accountNumber <<endl;
        cout<<"Account Balance: "<<getDetails()<<endl;
}

};

int main()
{
        double accBal;
        Account currentAccount;

        cout<<"Please enter the details"<<endl;
        cout<<"Enter Name:"<<endl;
        getline(cin, currentAccount.name);
        cout<<"Enter Account Number:"<<endl;
        cin>>currentAccount.accountNumber;


        cout<<"Enter Account Balance:"<<endl;
        cin>>accBal;
        currentAccount.setDetails(accBal);
        cout<<endl;

        currentAccount.displayDetails();
        return 0;
}
```

Set and get functions to manipulate private data member

Publically available data:
Assigning values in main

Private data:
Can only assign values from main



```
Please enter the details
Enter Name:
Dummy
Enter Account Number:
9256432
Enter Account Balance:
25000

Details are:
Account Holder: Dummy
Account Number: 9256432
Account Balance: 25000
```

OUTPUT OF EXAMPLE PROGRAM

CONSTRUCTOR

- A constructor is a special function that is a member of a class.
- C++ requires a constructor call for each object that's created, which helps ensure that each object is initialized properly before it's used in a program. The constructor call occurs implicitly when the object is created. This ensures that objects will always have valid data to work on.
- Normally, constructors are declared public.
- Constructors can be overloaded, just like other functions.

DIFFERENCE BETWEEN CONSTRUCTORS AND OTHER FUNCTIONS

- A constructor must be defined with the same name as the class.
- Constructors do not possess a return type (not even void).

DEFAULT CONSTRUCTOR

- A constructor without parameters is referred to as a default constructor.
- A constructor with parameters having default values is also called Default Constructor
- If a class does not contain a constructor definition, the compiler will create a minimal version of the default constructor as a public member. However, this constructor will not perform initialization. An uninitialized variable typically contains a "garbage" value.

By contrast, if a class contains at least one constructor, a default constructor must be defined explicitly.

CONSTRUCTOR OVERLOADING

When two or more functions share the same name, the function name is said to be overloaded. Any class member function may be overloaded, including the constructor. One constructor might take an integer argument, for example, while another constructor takes a double.

EXAMPLE CODE (No, One, Two Arguments Constructors and Destructor)

```
#include<iostream>
#include<string>
using namespace std;

class Account
{
private:
double balance; // Account balance

public: //Public interface
string name; // Account holder
long accountNumber; // Account number
```

OVERLOADING

```
Account()
{
        cout<<endl<<"Dafault constructor has been called"<<endl;
        name="Dummy";
        setDetails(85000);
}
```

Default/
No Argument
Constructor

```
Account(double balance)
{
        cout<<endl<<"One argument constructor has been called"<<endl;
        name = "Dummy1";
        setDetails(balance);
}
```

One Argument
Constructor

```
Account(string accHolder, double balance)
{
        cout<<endl<<"Two arguments constructor has been called"<<endl;
        name = accHolder;
        setDetails(balance);
}
```

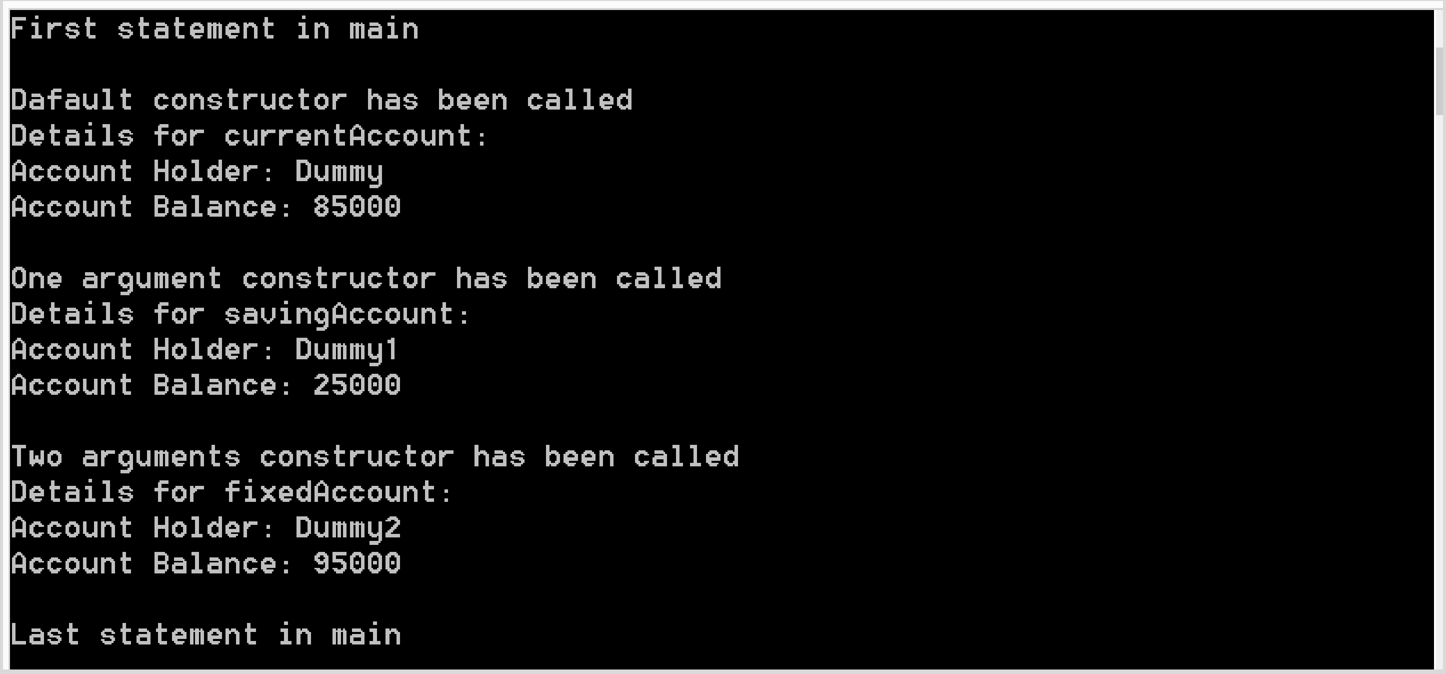
Two Arguments
Constructor

```
void setDetails(double balance)
{
this->balance = balance;
}
```

The 'this' pointer is used to retrieve the object's balance hidden by the local variable balance

```
double getDetails()
{
        return balance;
}
```

```
void displayDetails()
{
        cout<<"Account Holder: "<<name<<endl;
        cout<<"Account Balance: "<<balance<<endl;
}

};

int main()
{
        cout<<"First statement in main"<<endl;

        Account currentAccount;                     //calls default constructor
        cout<<"Details for currentAccount: "<<endl;
        currentAccount.displayDetails();

        Account savingAccount(25000);               //calls one argument constructor
        cout<<"Details for savingAccount: "<<endl;
        savingAccount.displayDetails();

        Account fixedAccount("Dummy2",95000);       //calls two arguments constructor
        cout<<"Details for fixedAccount: "<<endl;
        fixedAccount.displayDetails();

        cout<<endl<<"Last statement in main"<<endl;

        return 0;

}
```



```
First statement in main

Dafault constructor has been called
Details for currentAccount:
Account Holder: Dummy
Account Balance: 85000

One argument constructor has been called
Details for savingAccount:
Account Holder: Dummy1
Account Balance: 25000

Two arguments constructor has been called
Details for fixedAccount:
Account Holder: Dummy2
Account Balance: 95000

Last statement in main
```

Output: EXAMPLE CODE (No, One, Two Arguments Constructors and Destructor)

EXAMPLE CODE (DEFAULT CONSTRUCTOR WITH DEFAULT VALUES)

```
class Car
{
private:
      double tyres, engine;

public:
                  Car(double t=4, double e=1)// this is also default constructor having default values for all parameters.

                  {
                           tyres=t;
                           engine=e;
                  }
};

int main()
{
          Car car;
}
```

LAB 05 EXERCISES

INSTRUCTIONS:

NOTE: Violation of any of the following instructions may lead to the cancellation of your submission.

1) Create a folder and name it by your student id (k16-1234).
2) Paste the .cpp file for each question with the names such as Q1.cpp, Q2.cpp and so on into that folder.
3) Submit the zipped folder on slate.

QUESTION#1

Create a class Rectangle with attributes length and width, each of which defaults to 1. Provide member functions that calculate the perimeter and the area of the rectangle. Also, provide set and get functions for the length and width attributes. The set functions should verify that length and width are each floating-point numbers larger than 0.0 and less than 20.0.

QUESTION#2

Create a class called Employee that includes three pieces of information as data members—a first name (type char*), a last name (type string) and a monthly salary (type int). Your class should have a constructor that initializes the three data members. Provide a set and a get function for each data member. If the monthly salary is not positive, set it to 0.Write a test program that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again.

QUESTION#3

Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four data members—a part number (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type float). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a float value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. Write a test program that demonstrates class Invoice's capabilities.

QUESTION#4

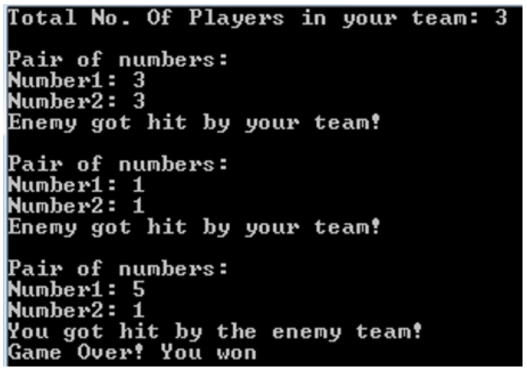
Write C++ code to represent a hitting game. The details are as follows:
This game is being played between two teams (i.e. your team and the enemy team).
The total number of players in your team is randomly generated and is passed as argument to the constructor.
The constructor generates a pair of numbers and matches each pair. If the numbers get matched, the following message is displayed:

"Enemy got hit by your team!"

Otherwise, the following message is displayed:

"You got hit by the enemy team!"

The number of hits should be equal to the number of players in your team.
The program should tell the final result of your team by counting the hits of both the teams.
Consider the following sample output:



```
Total No. Of Players in your team: 3

Pair of numbers:
Number1: 3
Number2: 3
Enemy got hit by your team!

Pair of numbers:
Number1: 1
Number2: 1
Enemy got hit by your team!

Pair of numbers:
Number1: 5
Number2: 1
You got hit by the enemy team!
Game Over! You won
```

QUESTION#5

MyJava Coffee Outlet runs a catalog business. It sells only one type of coffee beans. The company sells the coffee in 2-lb bags only and the price of a single 2-lb bag is $5.50 when a customer places an order, the company ships the order in boxes. The boxes come in 3 sizes with 3 different costs:

	Large Box	Medium Box	Small Box
Capacity	20 Bags	10 Bags	5 Bags
Cost	$1.80	$1.00	$0.60

The order is shipped using the least number of boxes. For example, the order of 52 bags will be shipped in 2 boxes: 2 large boxes, 1medium and 1 small. Develop an application that computes the total cost of an order.

```
Number of Bags Ordered: 52
The Cost of Order: $ 286.00
Boxes Used:
2 Large - $3.60
1 Medium - $1.00
1 Small - $0.60
Your total cost is: $ 291.20
```

QUESTION#6

Write a class named Vehicle that can represent both the Rickshaw and Bike on the basis of number of wheels it has. Each vehicle has the following details

- **year**. An int that holds the vehicle’s model year.
- **manufacturer**. A string that holds the manufacturer name of that vehicle.
- **speed**. An int that holds the vehicle’s current speed.

In addition, the class should have the following member functions.

- **accelerate**. The accelerate function should add 5 to the speed member variable each time it is called.
- **brake**. The brake function should subtract 5 from the speed member variable each time it is called.

Demonstrate the class in a program that creates a Vehicle object for a Rickshaw and for a Bike both, and then calls the accelerate function five times. After each call to the accelerate function, get the current speed of the car and display it. Then, call the brake function two times. After each call to the brake function, get the current speed of the car and display it.

QUESTION#7

Define a class for a type called CounterType. An object of this type is used to count things, so it records a count that is a nonnegative whole number. Include a mutator function that sets the counter to a count given as an argument. Include member functions to increase the count by one and to decrease the count by one. Also, include a member function that returns the current count value and one that outputs the count.

QUESTION#8

Write a program that uses a class named CorpData to store the following information on a company division:

- Division name (such as East, West, North, or South)
- First quarter sales
- Second quarter sales
- Third quarter sales
- Fourth quarter sales
- Total annual sales
- Average quarterly sales

The program should create four objects of this class, each representing one of the following corporate divisions: East, West, North, and South. Each object should be passed in turn to a function that calculates and stores the total sales and average quarterly sales for that division. Once this has been done for each division, each object should be passed in turn to a function that displays the division name, total sales, and quarterly average.

QUESTION#9

Design a class for a widget manufacturing plant. Assuming that 10 widgets may be produced each hour, the class object will calculate how many days and hours it will take to produce any number of widgets. (The plant operates two shifts of eight hours each per day.) Write a program that asks the user for the number of widgets that have been ordered and then displays the number of days and hours it will take to produce them.

QUESTION#10

You are a programmer for the Standard Charted Bank assigned to develop a class that models the basic workings of a bank account. The class should perform the following tasks:

- Save the account balance.
- Save the number of transactions performed on the account.
- Allow deposits to be made to the account.
- Allow with drawls to be taken from the account.
- Calculate interest for the period.
- Report the current account balance at any time.
- Report the current number of transactions at any time.

Program Output

```
Menu
        a) Display the account balance
        b) Display the number of transactions
        c) Display interest earned for this period
        d) Make a deposit
        e) Make a withdrawal
         f) Add interest for this period
        g) Exit the program
```