SE-201 Object-Oriented Programming

Lab Manual Spring 2022

Name	: Musadique Hussain
Roll No.	: 31
Semester	: Spring Section: A
Group	: Software Engineering
Date	: 03 July 2022
Remarks	: _____
Signature	: _____

Lab #07

OBJECT,CLASSES AND CONSTRUCTORS

EXERCISE

1. Create an employee class. The member data should comprise an *int* for storing the employee number and a *float* for storing the employee's compensation. Member function should allow the user to enter this data and display it. Write a *main ()* that allow the user to enter data for three employees. And display it.

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

class Employee {
private:
    int number;
    float compensation;
public:
    void setdata() {

        cout << "Employee number" << endl;
        cin >> number;
        cout << "Employee compensation" << endl;
        cin >> compensation;
    }
    void displaydata() {

        cout << "Employee number is " << number << endl;
        cout << "Employee compensation is " << compensation << endl;
    }
};
int main() {
    Employee e1, e2, e3;
    e1.setdata();
    e2.setdata();
    e3.setdata();
    e1.displaydata();
    e2.displaydata();
    e3.displaydata();
    system(_Command:"pause>0");
    return 0;
}
```

```
E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
Employee number
67
Employee compensation
56.45
Employee number
87
Employee compensation
34.56
Employee number
76
Employee compensation
23.45
Employee number is 67
Employee compensation is 56.45
Employee number is 87
Employee compensation is 34.56
Employee number is 76
Employee compensation is 23.45
```

2.Create a class Account whose data members are private integer number and double balance. Account constructors initialize each data member to zero. Class takes two member functions namely setdata() which takes two arguments integer and double, and showdata() to display result on screen.

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

class Account {
private:
    int number;
    double balance;
public:
    Account() {
        number = 0;
        balance = 0;
    }
    void setdata(int n, double b) {
        number = n;
        balance = b;
    }
    void showdata() {
        cout << "The account number is " << number << endl;
        cout << "The account balance is " << balance << endl;
    }
};
int main() {
    Account a1, a2;
    a1.setdata(n:69, b:65000);
    a2.setdata(n:43, b:40000);
    a1.showdata();
    a2.showdata();
    system(_Command:"pause>0");
    return 0;
}
```

```
E:\Study Materials\C++\Kira\x64\Debug\Kira.exe
The account number is 69
The account balance is 65000
The account number is 43
The account balance is 40000
```

3.Create a class BankAccount whose attributes are: account number, account holder's name and balance. BankAccount constructor initializes the values of account number, account holder's name and balance. (Use the overloaded constructor). Create three accounts for three persons and display the same on your screen.

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

class Account {
private:
    string name;
    double balance;
public:
    Account() {
        balance = 0;
        name = "";
    }
    Account(string n) {
        name = n;
    }
    Account(string n, double b) {
        name = n;
        balance = b;
    }
    void showdata() {
        cout << "Your account name is " << name << endl;
        cout << "Your account balance is " << balance << endl;
    }
};
int main() {
    Account Ac;
    Ac.showdata();
    Account Ac1("Emma");
    Ac1.showdata();
    Account Ac2("Emma", 64000);
    Ac2.showdata();
    system(_Command:"pause>0");
    return 0;
}
```

```
Select E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
Your account name is
Your account balance is 0
Your account name is Emma
Your account balance is -9.25596e+61
Your account name is Emma
Your account balance is 64000
```

4. Create a class **THE_TIME**. The class has attributes **hour**, **minutes** and **second** and defaults to **00:00:00** on construction. The class specification shall have two constructors: one parameter-less constructor and one that takes the **hour**, **minutes** and **second** as the initial **time** for the instantiated object. The class should define ***set*** and ***get*** time functions that verify the time being passed in the *set* function. Additionally, there shall be a member function to increment the time by hour, minute and second. The **time** object should always remain in a consistent state. Write a driver program to validate the performance of the **THE_TIME** class. Be sure to test the following cases:
    a. Incrementing into the next minute.
    b. Incrementing into the next hour.
    c. Using '++' and '—'operator for increment and decrement.
    d. Display the time in above format.

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

class The_Time {
public:
    void setTime(int hours, int minutes, int seconds);

    void getTime(int& hours, int& minutes, int& seconds)const;

    void printTime() const;

    void IncrementSeconds();

    void IncrementMinutes();

    void IncrementHours();

    bool equalTime(const The_Time& otherTime) const;

    The_Time(int hours, int minutes, int seconds);

    The_Time();

private:
    int hr;
    int min;
    int sec;
};

```

```cpp
#include <iostream>
#include "Header.h"
using namespace std;

void The_Time::setTime(int hours, int minutes, int seconds) {

    if (0 <= hours && hours > 24)
        hr = hours;
    else
        hr = 0;
    if (0 <= minutes && minutes > 60)
        min = minutes;
    else
        min = 0;
    if (0 <= seconds && seconds > 60)
        sec = seconds;
    else
        sec = 0;
}
void The_Time::getTime(int& hours, int& minutes, int& seconds) const {

    hours = hr;
    minutes = min;
    seconds = sec;
}
void The_Time::printTime() const {
    if (hr < 10)
        cout << "0";
    cout << hr << ":";

    if (min < 10)
        cout << "0";
    cout << min << ":";

    if (sec < 10)
        cout << "0";
    cout << sec << endl << endl;
}
void The_Time::IncrementHours() {
    hr++;
    if (hr > 23)
        hr = 0;
}
void The_Time::IncrementMinutes() {
    min++;
    if (min > 59) {
        min = 0;
        IncrementHours();
    }
}
void The_Time::IncrementSeconds() {
    sec++;
    if (sec > 59) {
        sec = 0;
        IncrementMinutes();
    }
}
bool The_Time::equalTime(const The_Time& otherTime)const {
    return(hr == otherTime.hr
        && min == otherTime.hr
        && sec == otherTime.sec);
}
The_Time::The_Time(int hours, int minutes, int seconds) {

    setTime(hours, minutes, seconds);
}
The_Time::The_Time() {
    setTime(hours: 0, minutes: 0, seconds: 0);
}
```

```
int main() {
    The_Time cellphoneTime;
    The_Time mywatch;
    int hrs, mins, secs;

    cout << "Please enter the hours for cell phone clock" << endl;
    cin >> hrs >> mins >> secs;
    cellphoneTime.setTime(hours:hrs, minutes:mins, seconds:secs);
    cellphoneTime.printTime();
    cout << "Increment Hours.\n";
    cellphoneTime.IncrementHours();
    cellphoneTime.printTime();
    cout << "Increment Minutes.\n";
    cellphoneTime.IncrementMinutes();
    cellphoneTime.printTime();
    cout << "Increment Seconds.\n";
    cellphoneTime.IncrementSeconds();
    cellphoneTime.printTime();
    system(_Command:"pause>0");
    return 0;
}
```

```
Select E:\Study Materials\C++\Kira\x64\Debug\Kira.exe
Please enter the hours for cell phone clock
24
60
60
00:00:00

Increment Hours.
01:00:00

Increment Minutes.
01:01:00

Increment Seconds.
01:01:01
```

5.Create a class rectangle which has two attributes: length and breadth. You should include two methods area() and perimeter().The class rectangle has 2 constructors, one were the attributes are given and the other where the user supplies the values. Write a program that creates 2 instances of this class using both constructors, calculates the area and the perimeter of each rectangle object, and displays the result to the screen.

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

class Rectangle {
private:
    double width;
    double breadth;
public:
    Rectangle() {
        width = 12;
        breadth = 2;
    }
    Rectangle(double w, double b) {
        width = w;
        breadth = b;
    }
    void area() {
        cout << "The area of rectangle is " << width * breadth << endl;
    }
    void perimeter() {
        cout << "The perimeter of rectangle is " << 2 + (width * breadth) << endl;
    }
```

```
};
int main() {
    Rectangle r1;
    r1.area();
    r1.perimeter();
    system(_Command:"pause>0");
    return 0;
}
```

```
E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
The area of rectangle is 24
The perimeter of rectangle is 26
```

6. Define a class Complex_No that has two member variables; Real and Imaginary. Also include following in the class

- A parameterized constructor that takes Real and Imaginary values as argument.
- A default constructor that assign zero to Real and Imaginary.
- A copy constructor
- A method Display that shows the value of complex number in appropriate format.
- A method Magnitude that calculates the magnitude of complex number
- A method Add that adds two complex numbers and return result; take one complex number as argument. Write a driver program to test your class.

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

class Complex_NO {
private:
    double real;
    double imaginary;
    double magnitude;
public:
    Complex_NO() {
        real = 0;
        imaginary = 0;
        magnitude = 0;
    }
    Complex_NO(double r, double i,double m) {
        real = r;
        imaginary = i;
        magnitude = m;
    }
    Complex_NO(Complex_NO& CN1) {
        real = CN1.real;
        imaginary = CN1.imaginary;
```

```
24              magnitude = CN1.magnitude;
25          }
26          void display() {
27              cout << "Real number is " << real << endl;
28              cout << "Imaginary number is " << imaginary << endl;
29              cout << "Magnitude is " << magnitude * sqrt( _X: (real * real) + (imaginary * imaginary));
30              cout << endl;
31              cout << "Two complex numbers are added " << real + imaginary << endl;
32              cout << endl;
33          }
34
35      };
36      int main() {
37          Complex_NO CN(12,2.1,3.4);
38          CN.display();
39          Complex_NO CN2(CN);
40          CN.display();
41          system( _Command: "pause>0");
42          return 0;
43      }
```

```
E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
Real number is 12
Imaginary number is 2.1
Magnitude is 41.42
Two complex numbers are added 14.1

Real number is 12
Imaginary number is 2.1
Magnitude is 41.42
Two complex numbers are added 14.1
```