

CC-211L

Object Oriented Programming

Laboratory 04

OOP Primitives and Composition / Aggregation

Version: 1.0.0

Release Date: 02-02-2023

Department of Information Technology

University of the Punjab

Lahore, Pakistan

Contents:

- Learning Objectives
- Required Resources
- General Instructions
- Background and Overview
 - Reference to an Object
 - Const Keyword
 - Composition
 - Member Initializer List
- Activities
 - Pre-Lab Activity
 - Returning reference to private data members
 - Task 01
 - Task 02
 - In-Lab Activity
 - Default Memberwise Assignment
 - const Objects and Member functions
 - Composition
 - Task 01
 - Task 02
 - Task 03
 - Post-Lab Activity
 - Task 01
- Submissions
- Evaluations Metric
- References and Additional Material
- Lab Time and Activity Simulation Log

Learning Objectives:

- Returning Reference to private Data Members
- Default Memberwise Assignment
- const Objects
- const Member Functions
- Object Composition and Aggregation

Resources Required:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022

General Instructions:

- In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how is s/he doing, this may result in negative marking. You can **ONLY** discuss with your Teaching Assistants (TAs) or Lab Instructor.
- Your TAs will be available in the Lab for your help. Alternatively, you can send your queries via email to one of the followings.

Teachers:		
Course Instructor	Prof. Dr. Syed Waqar ul Qounain	swjaffry@pucit.edu.pk
Lab Instructor	Azka Saddiqa	azka.saddiqa@pucit.edu.pk
Teacher Assistants	Saad Rahman	bsef19m021@pucit.edu.pk
	Zain Ali Shan	bcsf19a022@pucit.edu.pk

Background and Overview:

Reference to an Object:

A reference to an object in C++ is an alias that allows you to access an object by another name. It is similar to a pointer, except that you do not need to use the dereference operator (*) to access the object. A reference is created using the & operator. References are typically used to create a more concise and efficient way of referring to an object without needing to copy the object itself. Additionally, references can be used to pass an object into a function without needing to make a copy of the object.

Const Keyword

The const keyword in C++ is used to declare a constant. It is a modifier that indicates that the value of the variable or object it is applied to cannot be changed. Constants are useful for declaring values that will not change throughout the program, such as mathematical constants or other unchanging values.

Composition

Composition in C++ is a way to combine objects or classes into a bigger class or object. This type of relationship is known as a "has-a" relationship, as the composed class "has a" reference to the other class. Composition is used to model a "part-of" relationship, where one object can contain multiple smaller objects. This allows for complex behavior with simpler parts. For example, a car class may contain a steering wheel, an engine, and a transmission as separate objects. All of these objects work together to make up the car, but can be managed and updated independently. The composition also allows for greater flexibility and reuse of code, as the objects can be changed and reused in different contexts.

Member Initializer List:

Member Initializer Lists in C++ are a way to initialize data members of a class. They are used to initialize member variables to a known state before the body of the constructor is run.

Activities:

Pre-Lab Activities:

Returning Reference to a private data member:

Returning a reference to a private data member is a way for a class to allow other classes and functions to access a private variable without making the variable public. This allows the private variable to remain protected while still providing a way for it to be accessed.

The use of the `const` keyword is of utmost importance in this regard, as it prevents external code from modifying private data members by not allowing the reference to be used to change their values. If the reference return type is declared `const`, the reference is a nonmodifiable value, meaning it cannot be used to modify the data. On the other hand, if the reference return type is not declared `const`, it can lead to subtle errors.

Example:

```

1  #include<iostream>
2  using namespace std;
3  class Time {
4  private:
5      unsigned int hour{ 0 }, minute{ 0 }, second{ 0 };
6  public:
7      void setTime(int h, int m, int s) {
8          // validate hour, minute and second
9          if ((h >= 0 && h < 24) && (m >= 0 && m < 60) && (s >= 0 && s < 60)) {
10             hour = h; minute = m; second = s;
11         }
12         else
13             throw invalid_argument("hour, minute and/or second was out of range");
14     }
15     unsigned int getHour() const {
16         return hour;
17     }
18     unsigned int& badSetHour(int hh) {
19         if (hh >= 0 && hh < 24)
20             hour = hh;
21         else
22             throw invalid_argument("hour must be 0-23");
23         return hour; // dangerous reference return
24     }
25 };
26 int main() {
27     Time t;
28     unsigned int& hourRef = t.badSetHour(20);
29     cout << "Valid hour before modification: " << hourRef;
30     hourRef = 30;
31     cout << "\nInvalid hour after modification: " << t.getHour();
32     t.badSetHour(12) = 74;
33     cout << "\n\nPOOR PROGRAMMING PRACTICE\n" << "t.badSetHour(12) as an lvalue, invalid hour:" << t.getHour() << "\n";
34 }

```

Fig. 01 (Returning reference to private data member)

Output:

```

Microsoft Visual Studio Debug Console
Valid hour before modification: 20
Invalid hour after modification: 30

POOR PROGRAMMING PRACTICE
t.badSetHour(12) as an lvalue, invalid hour:74

```

Fig. 02 (Returning reference to private data member)

If we use the 'const' keyword as a return type, the compiler will prevent any modification of private data members of a class, thus preserving the concept of encapsulation while still allowing access to those members. As seen in the code snippet below, the compiler will throw an error on any lines where we attempt to change the values of private data members.

```

1  #include<iostream>
2  using namespace std;
3  class Time {
4  private:
5      unsigned int hour{ 0 }, minute{ 0 }, second{ 0 };
6  public:
7      void setTime(int h, int m, int s) {
8          // validate hour, minute and second
9          if ((h >= 0 && h < 24) && (m >= 0 && m < 60) && (s >= 0 && s < 60)) {
10             hour = h; minute = m; second = s;
11         }
12         else
13             throw invalid_argument("hour, minute and/or second was out of range");
14     }
15     unsigned int getHour() const {
16         return hour;
17     }
18     const unsigned int& badSetHour(int hh) {
19         if (hh >= 0 && hh < 24)
20             hour = hh;
21         else
22             throw invalid_argument("hour must be 0-23");
23         return hour; // dangerous reference return
24     }
25 };
26 int main() {
27     Time t;
28     const unsigned int& hourRef = t.badSetHour(20);
29     cout << "Valid hour before modification: " << hourRef;
30     hourRef = 30;
31     cout << "\nInvalid hour after modification: " << t.getHour();
32     t.badSetHour(12) = 74;
33     cout << "\n\nPOOR PROGRAMMING PRACTICE\n" << "t.badSetHour(12) as an lvalue, invalid hour:" << t.getHour()<< "\n";
34 }

```

Fig. 03 (Returning reference to private data member)

Error List:

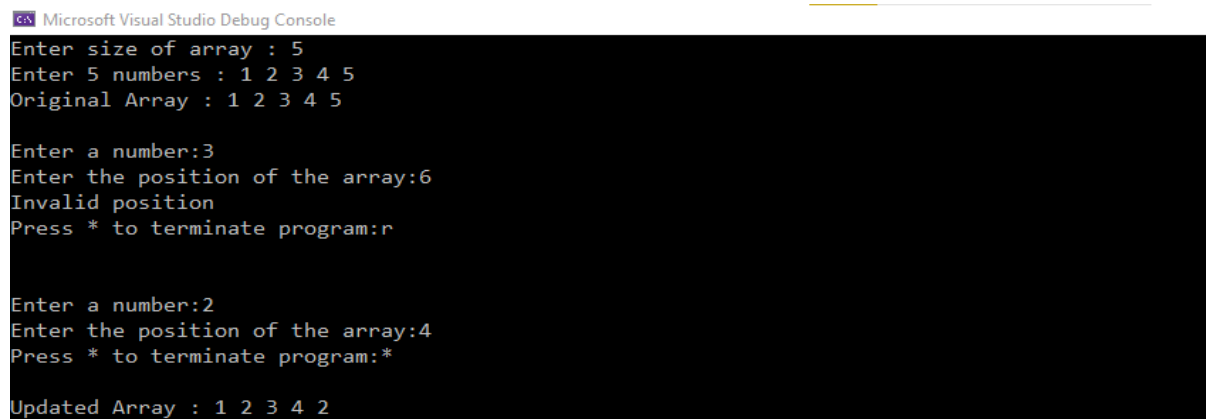
abc	E0137	expression must be a modifiable lvalue	Project10	Source.cpp	30
abc	E0137	expression must be a modifiable lvalue	Project10	Source.cpp	32

Fig. 04 (Returning reference to private data member)

Task 01: Array Updating**[Estimated time 30 minutes / 20 marks]**

Write a program that will allow the user:

- To create an array of a specified size
- Enter a number and a position within the array, and then update the array accordingly.
- Display updated array on Console.
- The program will utilize a function with a reference return type to update the array with the user's inputs.
- Do not allocate extra space for another array. You must do this by modifying the input array in-place.

Sample Output:


```

Microsoft Visual Studio Debug Console
Enter size of array : 5
Enter 5 numbers : 1 2 3 4 5
Original Array : 1 2 3 4 5

Enter a number:3
Enter the position of the array:6
Invalid position
Press * to terminate program:r

Enter a number:2
Enter the position of the array:4
Press * to terminate program:*

Updated Array : 1 2 3 4 2

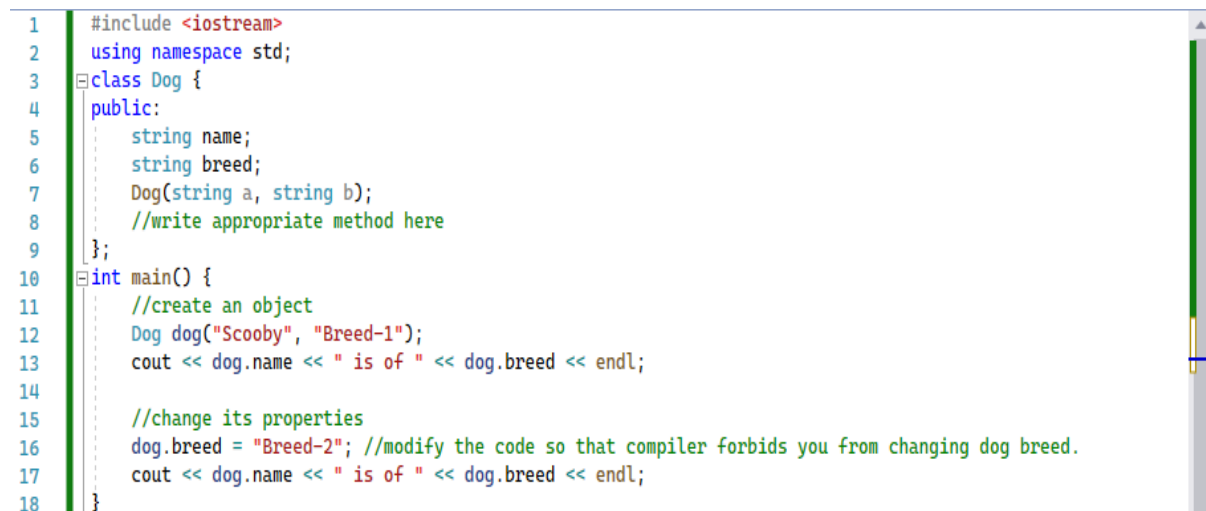
```

Fig. 05 (Pre-Lab Task)

Submit “.cpp” file named “RollNo” on Google Classroom.

Task 02: Forbid Change**[10 minutes / 10 marks]**

Update the following code so that the user won't be able to change the dog's breed once entered.



```

1  #include <iostream>
2  using namespace std;
3  class Dog {
4  public:
5      string name;
6      string breed;
7      Dog(string a, string b);
8      //write appropriate method here
9  };
10 int main() {
11     //create an object
12     Dog dog("Scooby", "Breed-1");
13     cout << dog.name << " is of " << dog.breed << endl;
14
15     //change its properties
16     dog.breed = "Breed-2"; //modify the code so that compiler forbids you from changing dog breed.
17     cout << dog.name << " is of " << dog.breed << endl;
18 }

```

Fig. 06 (Pre-Lab Task)

Submit “.cpp” file named “RollNo” on Google Classroom.

In-Lab Activities:

Default Memberwise Assignment:

The assignment operator (=) can be used to assign an object to another object of the same type. By default, such assignment is performed by **Memberwise assignment** each data member of the object on the right of the assignment operator is assigned individually to the same data member in the object on the left of the assignment operator.

Example:

```

1  #include <iostream>
2  using namespace std;
3  class Test {
4      int x;
5  public:
6      Test(int i)
7      {
8          x = i;
9      }
10     void print() { cout << x << endl; }
11 };
12 // Driver Code
13 int main()
14 {
15     Test t1(10);
16     Test t2(20);
17     t2 = t1;
18     t1.print();
19     t2.print();
20     return 0;
21 }

```

Fig. 07 (Memberwise Assignment)

Output:



```

Microsoft Visual Studio Debug Console
10
10

```

Fig. 08 (Memberwise Assignment)

const Objects and const Member functions:

Some objects need to be modifiable, and some do not. You may use const to specify that an object is not modifiable and that any attempt to modify the object should result in a compilation error.

const Time noon {12,0,0};

The statement above declares a const object noon of class Time and initializes it to 12 noon. It's possible to instantiate const and non-const objects of the same class.

Constant member functions are those functions that are denied permission to change the values of the data members of their class. To make a member function constant, the keyword "const" is appended to the function prototype and to the function definition header.

Example:

```

1  #include<iostream>
2  using namespace std;
3  class Demo
4  {
5      int x;
6  public:
7      void set_data(int);
8      int get_data() const;
9
10 };
11 void Demo::set_data(int a)
12 {
13     x = a;
14 }
15 int Demo::get_data() const
16 {
17     x++;
18     return x;
19 }
20 int main()
21 {
22     Demo d;
23     d.set_data(10);
24     cout << endl << d.get_data();
25     return 0;
26 }

```

Fig. 09 (const Objects & Member functions)

Error List:

abc E0137	expression must be a modifiable lvalue	Project10	Source.cpp	17
-----------	--	-----------	------------	----

Fig. 10 (const Objects & Member functions)

Composition:

In real-life complex objects are often built from smaller and simpler objects. For example, a car is built using a metal frame, an engine some tires, a transmission system, a steering wheel, and many other parts. This process of building complex objects from simpler ones is called C++ composition. It is also known as object composition.

A class can have one or more objects of other classes as members. A class is written in such a way that the object of another existing class becomes a member of the new class. this relationship between classes is known as C++ Composition. It is also known as containment, part-whole, or has-a relationship. In C++ Composition, an object is a part of another object. The object that is a part of another object is known as a sub-object. When a C++ Composition is destroyed, then all its sub-objects are destroyed as well. Such as when a car is destroyed, then its motor, frame, and other parts are also destroyed with it. It is a do or die relationship.

Example:

```

1  #include <iostream>
2  using namespace std;
3  class X
4  {
5  private:
6      int num;
7  public:
8      void set_value(int k){
9          num = k;
10     }
11     void show_sum(int n){
12         cout << "Sum of " << num << " and " << n << " = " << num + n << endl;
13     }
14 };
15 class Y
16 {
17 public:
18     X a; //Object of Class X
19     void print_result(){
20         a.show_sum(10);
21     }
22 };
23 int main()
24 {
25     Y b;
26     b.a.set_value(50);
27     b.a.show_sum(50);
28     b.print_result();
29 }

```

Fig. 11 (Composition)

Output:


```

Microsoft Visual Studio Debug Console
Sum of 50 and 50 = 100
Sum of 50 and 10 = 60

```

Fig. 12 (Composition)

Explanation:

In this program, class X has one data member 'num' and two member functions 'set_value()' and 'show_sum()'. The set_value() function is used to assign value to 'num'.

The show_sum() function uses an integer type parameter. It adds the value of parameter with the value of 'num' and displays the result on the Console.

Another class Y is defined after the class X. the class Y has an object of class X that is the C++ Composition relationship between classes X and Y. This class has its own member function print_result().

In the main() function, an object 'b' of class Y is created. The member function set_value() of object 'a' that is the sub-object of object 'b' is called by using two dot operators. One dot operator is used to access the member of the object 'b' that is object 'a', and second is used to access the member function set_value() of sub-object 'a' and 'num' is assigned a value 20.

In the same way, the show_sum() member function is called by using two dot operators. The value 50 is also passed as a parameter. The member function print_result of object 'b' of class Y is also called for execution. In the body of this function, the show_sum() function of object 'a' of class X is called for execution by passing value 10.

Task 01: Composition with two classes**[Estimated 30 minutes / 20 marks]**

Design a Line class to be implemented using composition with the Point class. The program should allow users to create Lines using two Points, and output the coordinates of each Point. Create constructors and getter/setter methods for each class, and write the logic in main () to generate the Lines and display the coordinates of the two Points.



Fig. 13 (In-Lab Task)

Task 02: Rational Numbers**[Estimated 40 minutes / 30 marks]**

Create a class **Rational** for performing arithmetic with fractions. Write a program to test your class. Use an integer variable to represent the Private data of the class

- The numerator
- The denominator

Provide a constructor that enables an object of this class to be initialized when it is declared. Create getter/setters for each data member. The getters must be constant. Provide public member functions that perform each of the following tasks:

- Add two rational numbers
- Subtract two Rational Numbers
- Multiply two Rational Number
- Print Rational Number in reduced form i.e., 2/4 should be displayed as 1/2.

Task 03: Course Composition**[Estimated 20 minutes / 10 marks]**

Provide the implementation of code given below. Declare an array of 5 objects of type section and set their values:

- Course number: 100 for all sections
- Credit hours: 3 for all sections
- Section number: give each section a unique number from 1-5

```
1  #include<iostream>
2  using namespace std;
3  class course
4  {
5      int courseNumber;
6      int creditHours;
7      public:
8          void setCourse(int x, int y);
9  };
10 class section
11 {
12     int secNumber;
13     course c;//composition
14     public:
15         void setSection(int, int, int);
16 };
17
```

Fig. 14 (In-Lab Task)

Post-Lab Activities:**Task 01: Composition with three classes****[Estimated 40 minutes / 20 marks]**

Write a program that uses composition to define classes, Time, Date and Event, which are used to store and print the details related to an event. The program should allow the user to enter the details of the event, such as the hour, day, month, year and name of the event. There should be a function named `printEventData()` in the event class. In `main()`, when this function is called with an argument like the name of the event or date of the event, all the details entered by the user for that event should be printed on screen. For example, if the user enters the date 25/12/2020, and there are two objects with this date, the output should be:

Christmas occurs on 25/12/2020 at 06:00

Quaid's Birthday occurs on 25/12/2020 at 01:15

And make sure to display when constructors and destructors are called for each class.

Note: Objects are constructed from the inside out (contained classes will be constructed first) and destructed in the reverse order, from the outside in (container class will be destructed first).

Submit “.cpp” file named your “**RollNo**” on Google Classroom.

Submissions:

- For In-Lab Activity:
 - Save the files on your PC.
 - TA's will evaluate the tasks offline.
- For Pre-Lab & Post-Lab Activity:
 - Submit the .cpp file on Google Classroom and name it to your roll no.

Evaluations Metric:

- All the lab tasks will be evaluated offline by TA's
- **Division of Pre-Lab marks:** [30 marks]
 - Task 01: Array Updating [20 marks]
 - Task 02: Forbid Change [10 marks]
- **Division of In-Lab marks:** [60 marks]
 - Task 01: Composition with two classes [20 marks]
 - Task 02: Rational Numbers [30 marks]
 - Task 03: Course Composition [10 marks]
- **Division of Post-Lab marks:** [20 marks]
 - Task 01: Composition with three classes [20 marks]

References and Additional Material:

- **Return By Reference**
<https://www.scaler.com/topics/cpp-return-reference/>
- **Const object and data members**
https://www.linuxtopia.org/online_books/programming_books/thinking_in_c++/Chapter08_024.html
- **Composition**
<https://www.geeksforgeeks.org/object-composition-delegation-in-c-with-examples/>

Lab Time Activity Simulation Log:

- Slot – 01 – 00:00 – 00:15: Class Settlement
- Slot – 02 – 00:15 – 00:40: In-Lab Task
- Slot – 03 – 00:40 – 01:20: In-Lab Task
- Slot – 04 – 01:20 – 02:20: In-Lab Task
- Slot – 05 – 02:20 – 02:45: Evaluation of Lab Tasks
- Slot – 06 – 02:45 – 03:00: Discussion on Post-Lab Task