



# Object Oriented Programming

## Lab Manual 3



### Introduction

After a week of rigorous coding, Welcome back!

**You have learned all about the Class Attributes and the Class Constructors in the previous lab manuals. Let's move on to the next, new, and interesting concepts.**

Students, In Object-Oriented Programming, the Class is a combination of data members and member functions.

Previously, we have learnt that there are 2 types of constructors.

1. Default Constructors
2. Parameterized Constructors

### Practice Output Based Questions:

In this Lab, we will learn about something that is known as a **Copy Constructor**.

### Copy Constructor

Right now, we are initializing the attributes of an object using the Constructor (Default or Parameterized). There is another Constructor called **Copy Constructor** that creates a new object (**separate memory on the heap**) by copying variables from another object.

**Note:** The copy constructor requires a complete object as an argument.

Code:



# Object Oriented Programming

## Lab Manual 3



```
class Student
{
    public Student()
    {
        Console.WriteLine("Default Constructor");
    }
    public Student(Student s)           //Copy Constructor
    {
        sname = s.sname;
        matricMarks = s.matricMarks;
        fscMarks = s.fscMarks;
        ecacMarks = s.ecacMarks;
        aggregate = s.aggregate;
    }
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecacMarks;
    public float aggregate;
}
```

Now, as you can see it will assign all the values from the received parameter to the class attributes.

### Code: Constructor Call

```
Student s1 = new Student();
s1.sname = "Jack";
Student s2 = new Student(s1);
Console.WriteLine(s1.sname);
Console.WriteLine(s2.sname);
```

As you can see now, the copy constructor receives an object as parameter and inside the constructor function implementation, it is assigning the values of each attribute of the received object to each corresponding attribute of the newly created object.

**NOTE:** Every time you create an object using a copy constructor, it creates new memory in HEAP. UNLESS and UNTIL it is important, you are advised to not make extra copies of the class objects using this method.

**Congratulations !!!! You have just learned how to create different types of constructors for creating class objects.**



# Object Oriented Programming

## Lab Manual 3



You may take a two minutes break, as there is much more code to come.

**Behaviors of the Class:**



# Object Oriented Programming

## Lab Manual 3



### FOR EACH LOOP

We shall also cover an additional concept that would be very useful for you while handling data structures such as lists, arrays, and class objects.

The foreach loop is just like the traditional for loop that is used to execute given lines of code repetitively.

#### Syntax:

**foreach** (datatype var\_name **in** Array/List/Object Name){ }

Look at the following code snippet to have a clear understanding of this concept.

**Task:** Write a program that prints a hard-coded list on the console screen.

#### Solution

Write the following code on your computer and execute the program by clicking on the start button.

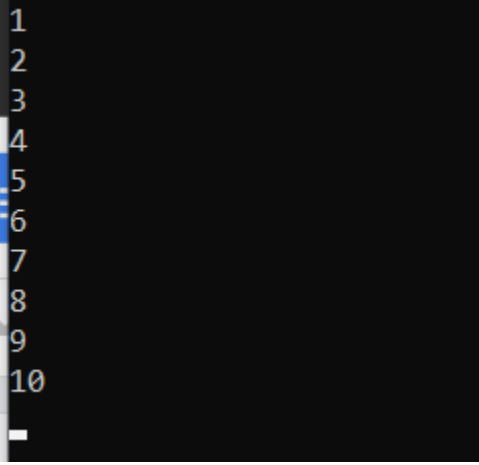
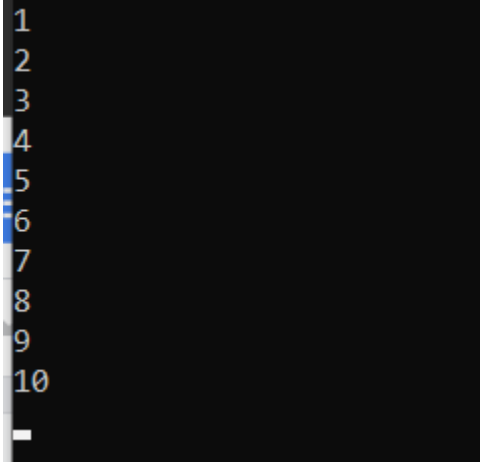
Code: With For Loop	Code: With Foreach loop
<pre>0 references static void Main(string[] args) {     List&lt;int&gt; numbers = new List&lt;int&gt;() { 1,2,3,4,5,6,7,8,9,10};     for (int i =0; i&lt;numbers.Count; i++)     {         Console.WriteLine(numbers[i]);     }     Console.Read(); }</pre>	<pre>static void Main(string[] args) {     List&lt;int&gt; numbers = new List&lt;int&gt;() { 1,2,3,4,5,6,7,8,9,10};     foreach (int i in numbers)     {         Console.WriteLine(i);     }     Console.Read(); }</pre>
Output:	Output:



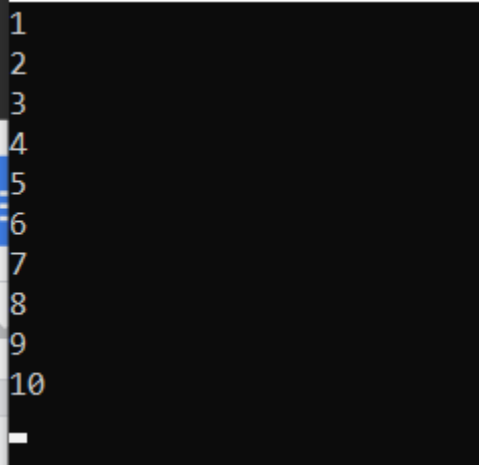
# Object Oriented Programming

## Lab Manual 3



	
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------

Here is another tip for you, the “foreach” loop allows you to even iterate a data structure even if you do not know the datatype of that data structure.

<b>Code:</b>	
<pre>static void Main(string[] args) {     List&lt;int&gt; numbers = new List&lt;int&gt;() { 1,2,3,4,5,6,7,8,9,10};     foreach (var i in numbers)     {         Console.WriteLine(i);     }     Console.Read(); }</pre>	Just write the “var” keyword as datatype and it will automatically detect the correct data type of the given variable.
	As you can observe, the program generates the same output as it did in the previous example.

**Congratulations !!!! You have just learned how to use the “foreach” loop.**



# Object Oriented Programming

## Lab Manual 3



### Task 1: ClockType Class

Suppose, that we want to define a class to implement the time of day in a program.

Because a clock gives the time of day, let us call this class clockType.

Furthermore, to represent time in computer memory, we use three int variables: one to represent the hours, one to represent the minutes, and one to represent the seconds.

Suppose these three variables are: int hr; int min; int sec;

We also want to perform the following operations on the time:

1. Set the time(seconds, minutes, hours). (default Constructor and parameterized constructors)
2. Print the time.
3. Increment the time by one second.
4. Increment the time by one minute.
5. Increment the time by one hour.
6. Compare the two times for equality.
  - With manual timings
  - With passed object's timings

### Solution:

Sr. #	Action	Description
1.	<pre>using ...  namespace testWeek3.BL {     10 references     class clockType     {         public int hours;         public int minutes;         public int seconds;     } }</pre>	Create a class with these three data members.



# Object Oriented Programming

## Lab Manual 3



2.	<pre>public clockType() {     hours = 0;     minutes = 0;     seconds = 0; }</pre>	Define a <b>default constructor</b>  <b>Note:</b> We are including this function inside the class ClockType
3.	<pre>public clockType(int h) {     hours = h; } 0 references public clockType(int h, int m) {     hours = h;     minutes = m; } 0 references public clockType(int h, int m, int s) {     hours = h;     minutes = m;     seconds = s; }</pre>	Define <b>parameterized constructors</b> with one, two, and three parameters individually.  <b>Note:</b> We are including these functions inside the class ClockType
4.	<pre>public void incrementSecond() {     seconds++; } 0 references public void incrementhours() {     hours++; } 0 references public void incrementminutes() {     minutes++; }</pre>	Define <b>member functions</b> for incrementing minutes, hours, and seconds individually.  <b>Note:</b> We are including these functions inside the class ClockType
5.	<pre>public void printTime() {     Console.WriteLine(hours + " " + minutes + " " + seconds); }</pre>	Define <b>member function</b> for printing time on screen. <b>Note:</b> We are including these functions inside the class ClockType



# Object Oriented Programming

## Lab Manual 3



6.	<pre>public bool isEqual(int h, int m, int s) {     if (hours == h &amp;&amp; minutes == m &amp;&amp; seconds == s)     {         return true;     }     else     {         return false;     } }</pre>	<p>Define <b>isEqual()</b> member function for comparing time with provided manual time.</p> <p><b>Note:</b> We are including these functions inside the class <b>ClockType</b></p>
7.	<pre>public bool isEqual(clockType temp) {     if (hours == temp.hours &amp;&amp; minutes == temp.minutes &amp;&amp; seconds == temp.seconds)     {         return true;     }     else     {         return false;     } }</pre>	<p>Define <b>isEqual()</b> member function for comparing time with provided object's time.</p> <p><b>Note:</b> We are including these functions inside the class <b>ClockType</b></p>
8.	<pre>// default constructor clockType empty_time = new clockType(); Console.WriteLine("Empty time: "); empty_time.printTime();  // Parameterized constructor (one parameter) clockType hour_time = new clockType(8); Console.WriteLine("Hour time: "); hour_time.printTime();  // Parameterized constructor (two parameters) clockType minute_time = new clockType(8, 10); Console.WriteLine("Minute time: "); minute_time.printTime();  // Parameterized constructor (three parameters) clockType full_time = new clockType(8, 10, 10); Console.WriteLine("Full time: "); full_time.printTime();</pre>	<p>Implement the <b>Main Function</b></p> <ul style="list-style-type: none"><li>• Default Constructor</li><li>• Parameterized Constructors</li><li>• Calling <b>PrintTime</b> function</li></ul>





# Object Oriented Programming

## Lab Manual 3



9.	<pre>// increment second full_time.incrementSecond(); Console.WriteLine("Full time (Increment Second): "); full_time.printTime();  // increment hours full_time.incrementhours(); Console.WriteLine("Full time (Increment hours): "); full_time.printTime();  // increment mintues full_time.incrementminutes(); Console.WriteLine("Full time (Increment Mintues): "); full_time.printTime();</pre>	Call functions with <ul style="list-style-type: none"><li>• Increment seconds</li><li>• Increment minutes</li><li>• Increment hours</li></ul>
10.	<pre>// chcek if equal bool flag = full_time.isEqual(9, 11, 11); Console.WriteLine("Flag: " + flag);  // chcek if equal with object clockType cmp = new clockType(10, 12, 1); flag = full_time.isEqual(cmp); Console.WriteLine("Object Flag: " + flag);</pre>	Call functions with <ul style="list-style-type: none"><li>• Manual timings</li><li>• Objects timings</li></ul>
11.	<pre>Empty time: 0 0 0 Hour time: 8 0 0 Minute time: 8 10 0 Full time: 8 10 10 Full time (Increment Second): 8 10 11 Full time (Increment hours): 9 10 11 Full time (Increment Mintues): 9 11 11 Flag: True Object Flag: False</pre>	The output of the Program:

You have made it through all that. Excellent work students !!!

There are a few challenges ahead. Try to solve those on your own



# Object Oriented Programming

## Lab Manual 3



### Challenge # 1:

**Task:** Write a program that performs the following:

Enhance Programming Exercise by adding functions to the class clockType so that a program that uses this class can perform the following operations:

- Returns the **elapsed time** of the day of a clock in seconds.
- Returns the **remaining time** of the day to a clock in seconds.
- Determines and outputs how far apart in time **two clocks** are. (pass one parameter of class clockType)
- Outputs the time in the form hr:min: sec.

Elapsed time = current time - 00:00:00

Remaining time = 24:00:00 - a current time

### Challenge # 2:

**Task:**

Develop a project with one class and data members that take input from the user, store it into the object, keep track of multiple objects and search the objects based on some criteria.

Miss Client wants to develop a software system for her departmental store. She wants this system to have the following 5 functionalities.

- Add Product.
- View All Product.
- Find Product with the Highest Unit Price
- View Sales Tax of All Products.
- Products to be Ordered. (less than the threshold)

### Project Requirements

Q: What kind of data is required to store for a product?

A: The following information is required

- Name of Product.
- Product Category.
- Product Price.
- Stock Quantity.
- Minimum Stock Quantity.

Note: At the moment, there are only two types of categories: Groceries and Fresh Fruit.



# Object Oriented Programming

## Lab Manual 3



Q: What is the threshold value?

A: Threshold value tells the owner when to order any product. For example, if the threshold value of a product is less than 10 the owner wants to order the product.

Q: How sales tax is calculated.

A: On All Grocery types of products, the sales tax is 10%, on all fruit types the tax is 5%, and if there is any other type the tax is 15%.

Q: How to Initialize the Products?

A: you are also required to implement default, parameterized, and copy constructors using class.

**Good Luck and Best Wishes !!**

**Happy Coding ahead :)**