Object Oriented Programming

Lab Manual 3

Introduction

After a week of rigorous coding, Welcome back!

You have learned all about the Class Members 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. In this Lab, we will learn about something that is known as a **Constructor**.

Let's do some coding.

Default Constructor

We have learned in the previous lab manual about how to create a **class object**.

Syntax:

class object_name = new class();

The code is written inside the main function in the driver program that creates an object of the respective class in that program.

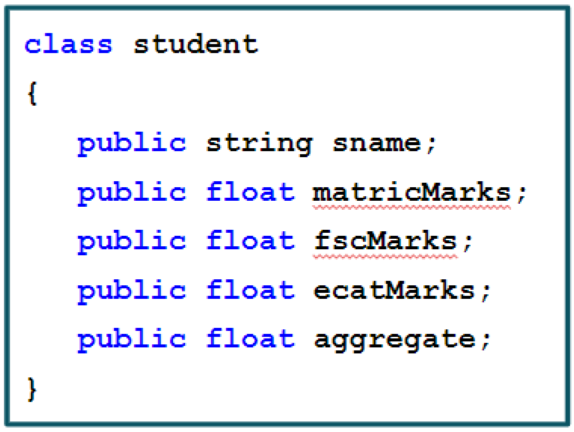
Students, **class()** is referred to as the default constructor that creates the object's memory in the heap and sets the attributes to the default values.

Task: Write a program that creates a new class of students and try printing the values of attributes without assigning them any values.

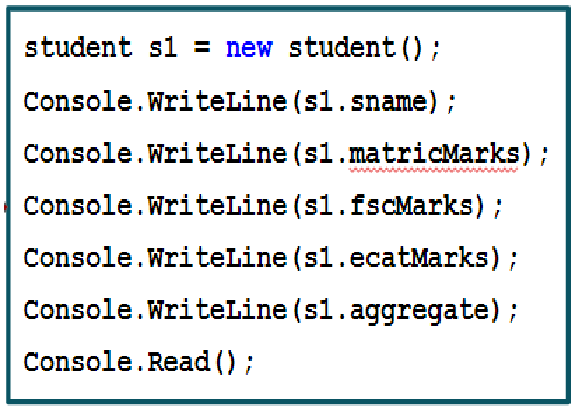
Solution:

Write the following code before the main function of the code and execute the program by clicking on the start button.

Code:

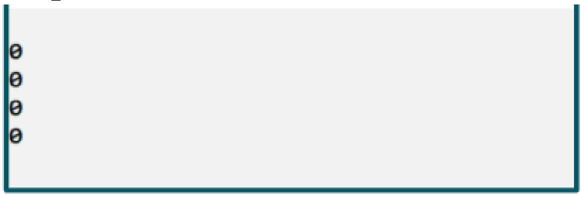


```
class student
{
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```



```
student s1 = new student();
Console.WriteLine(s1.sname);
Console.WriteLine(s1.matricMarks);
Console.WriteLine(s1.fscMarks);
Console.WriteLine(s1.ecatMarks);
Console.WriteLine(s1.aggregate);
Console.Read();
```

Output:



```
0
0
0
0
```

The code will generate a new class of students where each student would have the following properties.

- **string** type Name
- **float** type Matric marks, fsc marks, ecat marks, and aggregate

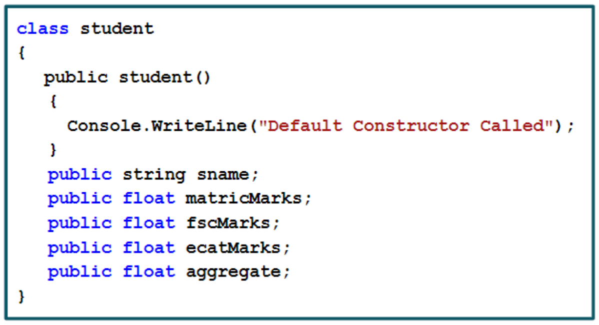
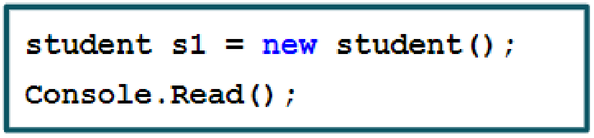
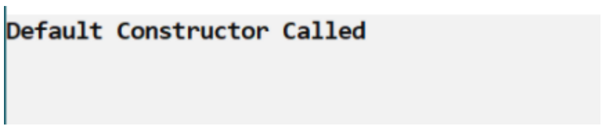
Observe the output, it reflects that C# has automatically assigned "**zero**" value to these data members of the class. However, if we want to change these default values, we can implement a **default constructor** in the class definition to override this functionality.

Creating a Default Constructor

In order to create a default constructor, you have to consider the following

- To create a default constructor, we use the **same name as the class**, followed by **parentheses** ()
- The constructor has the **same name as the class**, it is always **public**, and it does not have any **return type**.

Task: Write the code to create a default constructor that prints “Default Constructor Called”.
Solution:
Write the following code into the main function of the code and execute the program by clicking on the start button.

Code:	**Output:**
 	

```
class student
{
   public student()
   {
     Console.WriteLine("Default Constructor Called");
   }
   public string sname;
   public float matricMarks;
   public float fscMarks;
   public float ecatMarks;
   public float aggregate;
}
```

```
student s1 = new student();
Console.Read();
```

```
Default Constructor Called
```

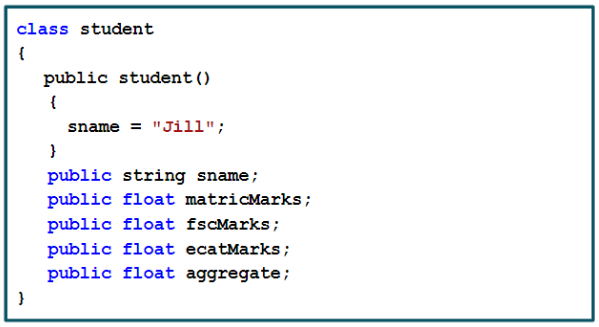
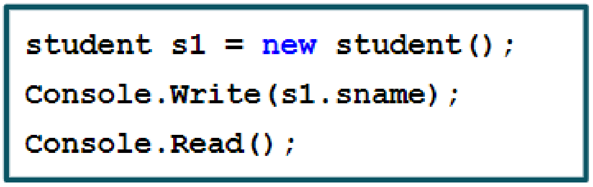
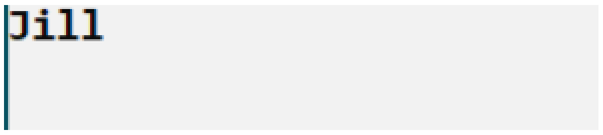
Hint: Never forget to write public before the constructor’s name.

Congratulations !!!! You have just learned how to create a default constructor.

Setting Attribute values through Default Constructor

Now, we will use the definition of default constructor to assign our desired “default” values for the class objects to be created.

Task: Write the code to create a default constructor that assigns a default value to one of the attributes.
Solution:
Write the following code into the main function of the code and execute the program by clicking on the start button.

Code:	Output:
 	

```
class student
{
   public student()
   {
     sname = "Jill";
   }
   public string sname;
   public float matricMarks;
   public float fscMarks;
   public float ecatMarks;
   public float aggregate;
}
```

```
student s1 = new student();
Console.Write(s1.sname);
Console.Read();
```

```
Jill
```

Now, all the newly created objects of class “**student**” shall be assigned the value of “**Jill**” for the “**sname**” attribute of the class.

Self Assessment Task 1(a): Create a default constructor that assigns values to all the attributes of the class.
You may use the student class from the above-mentioned example of the code snippet.

Self Assessment Task 1(b): Create multiple objects of the class and check if all the attributes in multiple objects have been assigned the default values.

Parameterized Constructor

Moving onwards, there is another type of constructor that is known as "**Parameterized Constructor**". As the name reflects, this constructor receives parameters from the user which means that the user can define the values of the attributes at the time of object creation.

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

Task: Write the code to create a class object and take the input user name at the time of object declaration.

Solution

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

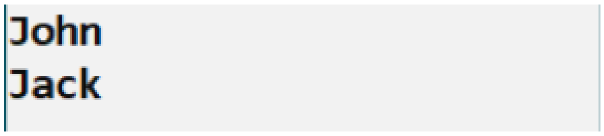
Code:

```
class student
{
    public student(string n)
    {
      sname = n;
    }
    public string sname;
    public float matricMarks;
    public float fscMarks;
    public float ecatMarks;
    public float aggregate;
}
```

// **Main Function**

```
student s1 = new student("John");
Console.WriteLine(s1.sname);
student s2 = new student("Jack");
Console.WriteLine(s2.sname);
Console.Read();
```

Output:



```
John
Jack
```

Observe that this time, the “**student constructor**” requires a single parameter from the user and assigns that value to the respective attribute of the class.

Similarly, you can create a parameterized constructor that will require a value for all the attributes of that class. In this way, however, the user would need to provide all the values that have been listed in the parameter list of the constructor function.

NOTE: You can not create an object without providing the list of parameters once you have defined a parameterized constructor. Attempting to do so would generate an error.

Self Assessment Task 1(a): Create a Parameterized constructor that assigns values to all the attributes of the class.
You may use the student class from the above-mentioned example of the code snippet.

Self Assessment Task 1(b): Create multiple objects of the class and check if all the attributes in multiple objects have been assigned the unique values.

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:

```
class student
{
   public student()
   {
      Console.WriteLine("Default Constructor");
   }
   public student(student s)             //Copy Constructor
   {
      sname = s.sname;
      matricMarks = s.matricMarks;
      fscMarks = s.fscMarks;
      ecatMarks = s.ecatMarks;
      aggregate = s.aggregate;
   }

   public string sname;
   public float matricMarks;
   public float fscMarks;
   public float ecatMarks;
   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 parameters and inside the constructor function implementation, it is assigning the values of each attribute of the received object to each corresponding attribute of newly object to be created.

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.

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

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.

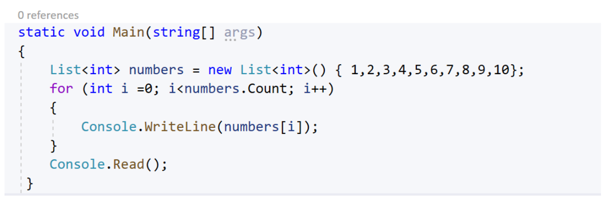
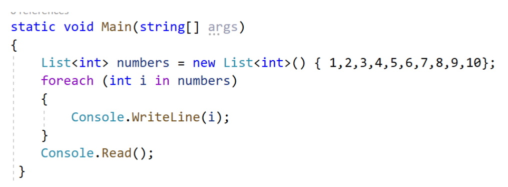
The foreach loop is just like the traditional for loop that is used to execute given lines of code repetitively.

Syntax:

foreach (datatype var_name in Array/List/Object Name){ }

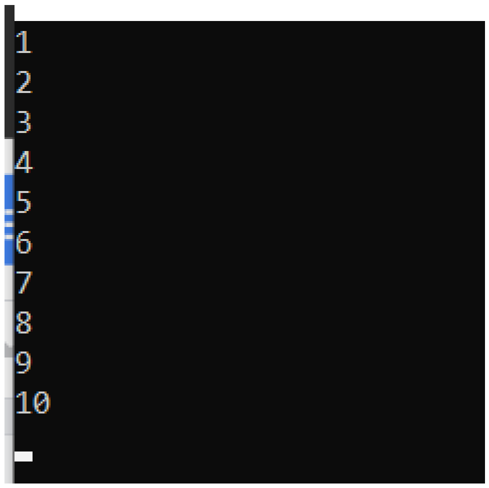
Look at the following code snippet to have a clear understanding of this concept.

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.

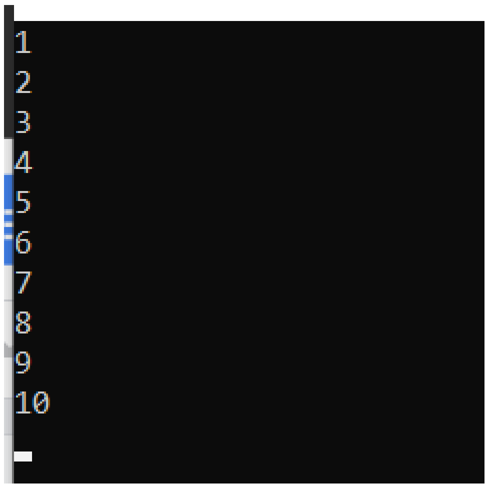
Code: With For Loop	**Code:** With Foreach loop
	
Output:	**Output:**

```
0 references
static void Main(string[] args)
{
    List<int> numbers = new List<int>() { 1,2,3,4,5,6,7,8,9,10};
    for (int i =0; i<numbers.Count; i++)
    {
        Console.WriteLine(numbers[i]);
    }
    Console.Read();
}
```

```
static void Main(string[] args)
{
    List<int> numbers = new List<int>() { 1,2,3,4,5,6,7,8,9,10};
    foreach (int i in numbers)
    {
        Console.WriteLine(i);
    }
    Console.Read();
}
```



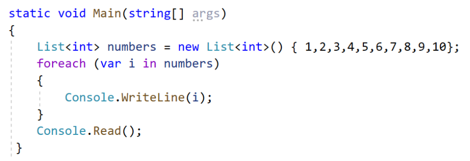
```
1
2
3
4
5
6
7
8
9
10
```



```
1
2
3
4
5
6
7
8
9
10
```

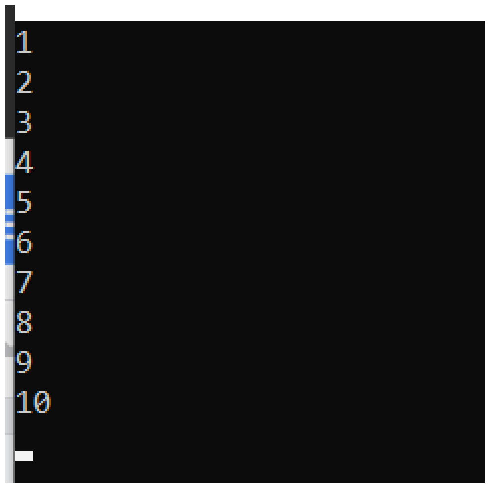
Here is another tip for you, the “foreach” loop allows you to even iterate a data structure even if you do not know the datatype of that data structure.

Code:



```
static void Main(string[] args)
{
    List<int> numbers = new List<int>() { 1,2,3,4,5,6,7,8,9,10};
    foreach (var i in numbers)
    {
        Console.WriteLine(i);
    }
    Console.Read();
}
```

Just write the “var” keyword as datatype and it will automatically detect the correct data type of the given variable.



```
1
2
3
4
5
6
7
8
9
10
```

As you can observe, the program generates the same output as it did in the previous example.

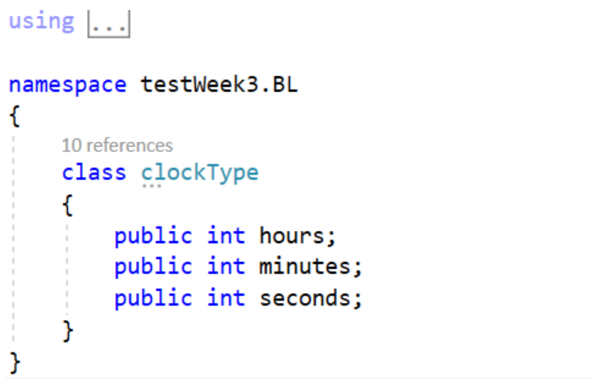
Congratulations !!!! You have just learned how to use the “foreach” loop.

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.	 `using [...]`
	
	`namespace testWeek3.BL`
	`{`
	`    10 references`
	`    class clockType`
	`    {`
	`        public int hours;`
	`        public int minutes;`
	`        public int seconds;`
	`    }`
	`}`
		Create a class with these three data members.

2.	`public clockType()`
	`{`
	`    hours = 0;`
	`    minutes = 0;`
	`    seconds = 0;`
	`}`
		Define a **default constructor**

3.	`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;`
	`}`
		Define **parameterized constructors** with one, two, and three parameters individually.

4.	`public void incrementSecond()`
	`{`
	`    seconds++;`
	`}`
	0 references
	`public void incrementhours()`
	`{`
	`    hours++;`
	`}`
	0 references
	`public void incrementminutes()`
	`{`
	`    minutes++;`
	`}`
		Define **member functions** for incrementing minutes, hours, and seconds individually.

5.	`public void printTime()`
	`{`
	`    Console.WriteLine(hours + " " + minutes + " " + seconds);`
	`}`
		Define **member function** for printing time on screen.

6.	
	```
	public bool isEqual(int h, int m, int s)
	{
	    if (hours == h && minutes == m && seconds == s)
	    {
	        return true;
	    }
	    else
	    {
	        return false;
	    }
	}
	```
		Define **isEqual() member function** for comparing time with provided manual time.

7.	
	```
	public bool isEqual(clockType temp)
	{
	    if (hours == temp.hours && minutes == temp.minutes && seconds == temp.seconds)
	    {
	        return true;
	    }
	    else
	    {
	        return false;
	    }
	}
	```
		Define **isEqual() member function** for comparing time with provided object's time.

8.	
	```
	// default constructor
	clockType empty_time = new clockType();
	Console.Write("Empty time: ");
	empty_time.printTime();

	// Parameterized constructor (one parameter)
	clockType hour_time = new clockType(8);
	Console.Write("Hour time: ");
	hour_time.printTime();

	// Parameterized constructor (two parameters)
	clockType minute_time = new clockType(8, 10);
	Console.Write("Minute time: ");
	minute_time.printTime();

	// Parameterized constructor (three parameters)
	clockType full_time = new clockType(8, 10, 10);
	Console.Write("Full time: ");
	full_time.printTime();
	```
		Implement the **Main Function**
		● Default Constructor
		● Parameterized Constructors
		● Calling **PrintTime** function

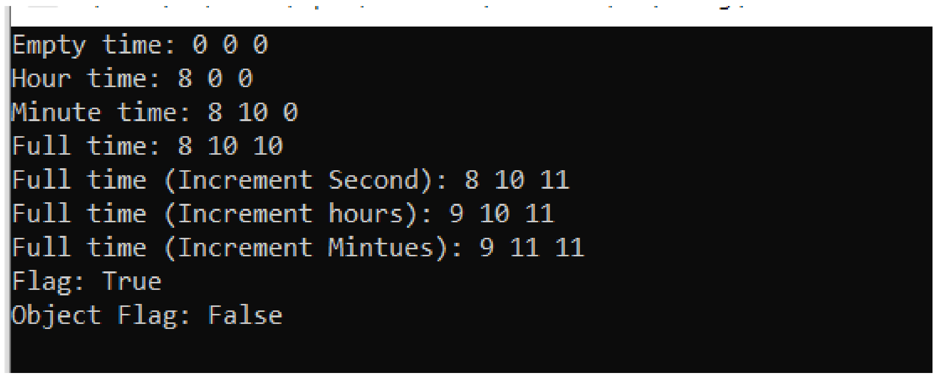
9.	
	```
	// increment second
	full_time.incrementSecond();
	Console.Write("Full time (Increment Second): ");
	full_time.printTime();

	// increment hours
	full_time.incrementhours();
	Console.Write("Full time (Increment hours): ");
	full_time.printTime();

	// increment mintues
	full_time.incrementminutes();
	Console.Write("Full time (Increment Mintues): ");
	full_time.printTime();
	```
		Call functions with
		● Increment seconds
		● Increment minutes
		● Increment hours

10.	
	```
	// 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);
	```
		Call functions with
		● Manual timings
		● Objects timings

11.	
	```
	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
	```
		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

Challenge # 1:
Task: Write a program that performs the following:

Add functions to this class so that a program that uses this class can set only the hours (another constructor), minutes(another constructor), or seconds(another constructor) and retrieve only the hours, minutes, or seconds. Also, write a driver program (main) to test your class.

Enhance Programming Exercise by adding functions to the class clockType so that a program that uses this class can perform the following operations:
a. Returns the elapsed time of the day of a clock in seconds.
b. Returns the remaining time of the day to a clock in seconds.
c. Determines and outputs how far apart in time two clocks are.
d. Outputs the time in the form hr:min: sec.

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.
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.

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 :)