I have been a team leader for the last 5 years and specialize in helping companies design software architects, set up process deployment, set up servers in the cloud, and train new members.

My strength is good logical thinking, good at teamwork, and always keeping a CAN-DO attitude.

What I am looking for is a company that I could add value to, where I could join a strong team.

Is this what CMC is looking for?

<https://www.w3schools.com/cs/cs_oop.php>

**O**bject-**O**riented **P**rogramming is about creating objects that contain both data and methods.

It is faster and easier to execute.

It provides a clear structure for the programs.

It makes the code easier to maintain, modify and debug.

It makes it possible to create full reusable applications with less code and shorter development time.

A class is a template for objects, and an object is an instance of a class.

A variable is declared directly in a class, it is often referred to as a field (or attribute).

Fields and methods inside classes are often referred to as "class members".

A static method can be accessed without creating an object of the class, while public methods can only be accessed by objects.

Access modifier **public**: the code is accessible for all classes.

Access modifier **private**: the code is only accessible within the same class.

Access modifier **protected**: the code is accessible within the same class, or in a class that is inherited from that class.

Access modifier **internal**: the code is only accessible within its assembly, but not from another assembly.

**Encapsulation** makes sure that "sensitive" data is hidden from users.

Better control of class members.

Fields can be made read-only, or write-only.

Increased security of data.

Declare fields as private

Provide public get and set methods, through properties, to access and update the value of a private field.

A property is like a combination of a field and a method,

**Inheritance** (Derived and Base Class)

It is useful for code reusability: reuse fields and methods of an existing class when you create a new class.

Derived Class (child) - the class that inherits from another class

Base Class (parent) - the class being inherited from

**Polymorphism** means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

It uses those methods to perform different tasks. This allows us to perform a single action in different ways.

**Abstraction** is the process of hiding certain details and showing only essential information to the user. An interface is a completely "abstract class", which can only contain abstract methods and properties (with empty bodies)

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

https://refactoring.guru/design-patterns/singleton

**Design patterns** are typical solutions to common problems

in software design. Each pattern is like a blueprint

that you can customize to solve a particular

design problem in your code.

Design patterns define a common language that you and your teammates can use to communicate more efficiently. You can say, “Oh, just use a Singleton for that,” and everyone will understand the idea behind your suggestion. No need to explain what a singleton is if you know the pattern and its name.

Three main groups of patterns

* **Creational patterns** provide object creation mechanisms that increase flexibility and reuse of existing code.
* **Structural patterns** explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.
* **Behavioral patterns** take care of effective communication and the assignment of responsibilities between objects.

**Abstract Factory** is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.

**Prototype** is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

**Builder** is a creational design pattern that lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code.

**Singleton** is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.

Test-driven development approach is writing unit tests before writing code.

When you do unit testing, you write a test after you write your code.

You can do unit testing without doing test-driven development. However, you can't do test-driven development without using unit tests.

5. .NET

5.1. EAGER/LAZY LOADING

Lazy loading delays the initialization of a resource. Related entities are loaded when the navigation property is accessed.

Eager loading initializes or loads a resource as soon as the code is executed. Related entities are loaded as part of the initial query.

Explicit loading - Related entities are loaded explicitly, not as part of the initial query, but at a later point in time.

5.2. IENUMERABLE/IQUERYABLE

The main difference between IEnumerable and IQueryable in C# is that IQueryable queries out-of-memory data stores, while IEnumerable queries in-memory data.

Moreover, IQueryable is part of .NET's System.LINQ namespace, while IEnumerable is in System.Collections namespace.

5.3. ASYNC/AWAIT

When we are dealing with UI, and button clicks, we use a long-running method like reading a large file or something else which will take a long time.

In that case, the entire application must wait to complete the whole task. In other words, if any process is blocked in a synchronous application,

the whole application gets blocked, and our application stops responding until the whole task completes.

Asynchronous programming is very helpful in this condition. By using Asynchronous programming,

the Application can continue with the other work that does not depend on the completion of the entire task.