1. **Difference between interfaces and abstract classes**

**What is an Abstract Class?**

An abstract class is a special kind of class that cannot be instantiated. So the question is why we need a class that cannot be instantiated? An abstract class is only to be sub-classed (inherited from). In other words, it only allows other classes to inherit from it but cannot be instantiated. The advantage is that it enforces certain hierarchies for all the subclasses. In simple words, it is a kind of contract that forces all the subclasses to carry on the same hierarchies or standards.

**What is an Interface**?

An interface is not a class. It is an entity that is defined by the word Interface. An interface has no implementation; it only has the signature or in other words, just the definition of the methods without the body. As one of the similarities to abstract class, it is a contract that is used to define hierarchies for all subclasses or it defines specific set of methods and their arguments. The main difference between them is that a class can implement more than one interface but can only inherit from one abstract class. Since C# doesn’t support multiple inheritance, interfaces are used to implement multiple inheritance.

**DIFFERENCE BETWEEN AN ABSTRACT CLASS AND AN INTERFACE:**

1. An Abstract class doesn't provide full abstraction but an interface does provide full abstraction; i.e. both a declaration and a definition is given in an abstract class but not so in an interface.
2. Using Abstract we cannot achieve multiple inheritance but using an Interface we can achieve multiple inheritance.
3. We cannot declare a member field in an Interface.
4. We cannot use any access modifier i.e. public, private, protected, internal etc. because within an interface by default everything is public.
5. An Interface member cannot be defined using the keyword static, virtual, abstract or sealed.
6. **What is CI/CD (Continuous Integration and Continuous Deployment).**

Continuous Integration is the practice of testing each change done to your codebase automatically and as early as possible. Continuous Deployment follows the testing that happens during Continuous Integration and pushes changes to a staging or production system

Continuous delivery is a DevOps software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

1. **What “Sealed” keyword used for?**

Sealed classes are used to restrict the inheritance feature of object oriented programming. Once a class is defined as a sealed class, the class cannot be inherited. In C#, the sealed modifier is used to define a class as sealed. In Visual Basic .NET the Not Inheritable keyword serves the purpose of sealed.

1. **Customization of “CRM Dynamics”**

Microsoft Dynamics CRM is a customer relationship management software package developed by Microsoft. ... Dynamics CRM is a server-client application, which, like Microsoft SharePoint, is primarily an IIS-based web application which also supports extensive web services interfaces.

**Customize MS Dynamics CRM for Businesses**

Microsoft Dynamics CRM is a product that allows a great deal of customization for your business. This includes creating new functionality, changing your user interface and navigation methods for better managing your customer relationships. Most of the customization is pretty straight forward and do not require a lot of investment. All you need to have is the System Administrator or System Customizer security role, or have equivalent privileges. In case of complex business requirements, Dynamics CRM provides Software Development Kit (SDK) for developers through which they can customize business logic or integrate Dynamics CRM with applications like SharePoint, MS Outlook etc

**7 Customization levels of Dynamics CRM for your Business:**

1. **Database schema level:** Microsoft Dynamics CRM database can be customized to add new tables and entities using administrative access directly from the CRM interface without making any database level changes.
2. **Application level:** Forms, views and templates can be customized as per your enterprise’s needs.
3. **Code level:** MS Dynamics CRM has a lot of visual tools available straight out of the box. In addition to this, it also lets you change the code to provide limitless possibilities.
4. **Feature level:** Microsoft Dynamics CRM offers several features for customization that helps the sales and marketing team capture and process more customer information.

I. Application Customization to Capture Additional Information

i. Customizing schema features

ii. Modifying forms feature for edit and update

iii. Publishing changes to forms

iv. Using the view editor

v. Altering previews

vi. Tailoring e-mail templates

vii. Replicating with the import/export feature

viii. Editing reports features

II. Business Logic Customization

Using the Workflow Editor to create customized workflow processes that best suits your business

1. **Business process level:** Each organization has distinct processes to manage customer relationships. This CRM management tool from Microsoft provides process workflow automation to automate business processes as needed.
2. **User Interface (UI) level:** Microsoft Dynamics can be customized for its menu items, tool tips, field level description and label text. It also supports custom UI themes and profiles.
3. **Deployment level:** Microsoft Dynamics CRM supports custom solution development and publishing service by importing and exporting features. This makes deployment of custom features very easy.

The CRM solution from Microsoft Dynamics is a great tool to deploy, customize, integrate and extend. It provides a wealth of opportunities for enterprises to create a highly customized and integrated application for their business helping them achieve their business goals more effectively.

1. **How to implement thread safe**

Sharing resources between threads is a frequent necessity in multithreaded programming. Multiple threads may need to access a shared database, for instance, or make updates to a set of system variables. When more than one thread simultaneously competes for access to shared resources, the possibility of a race condition occurs. A race condition exists when a thread modifies a resource to an invalid state, and then another thread attempts to access that resource and use it in the invalid state. Consider the following example:

Thread safety is a concept applicable in the context of multi-threaded programs. Multiple thread can access to the same address space at the same time. So, they can write to the exact same memory location at the same time. It is a defining property of threads. So, this property of thread is not good for the functionality.

So, Thread safety is a technique which manipulates shared data structure in a manner that guarantees the safe execution of a piece of code by the multiple threads at the same time. A code is called thread safe if it is being called from multiple threads concurrently without the breaking of functionalities.

Thread safety removes the following conditions in the code:

Race Condition

Deadlocks

1. **Why to use iDisposal interface.**

Use the Dispose method of this interface to explicitly release unmanaged resources in conjunction with the garbage collector. The consumer of an object can call this method when the object is no longer needed. Warning. It is a breaking change to add the IDisposable interface to an existing class.

1. **What is a design pattern?**

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.

1. **Name some design patterns in C#**

Layers (n-Tier architecture)

Facade (Service Layer)

Iterator (LINQ really)

Singleton.

Factory.

Proxy.

MVC (ASP.NET MVC)

MVVM (Silverlight)

**All about MVC:**

1. **What are different versions of MVC available, what is the use of it.**
2. **What is Razor (CSHTML), what is the use of it.**
3. **How do you maintain connection string to DataBase?**
4. **How we divide the functionality according to the screens (aspx forms) or views in MVC**