Singleton vs Static Classes.

The similarities between Singleton and Static class.

- Both static class and Singleton class can have only once instance of the copy that it is available in memory throughout the whole application. They both are used for holding the global state of an app.

- They both can be implemented as thread-safe.

Difference between Singleton and Static Class.

- Static is a language feature and Singleton is a design pattern.

- We can not create an instance of a static Class. But we can create a single instance of a Singleton Class and them reuse that singleton instance.

- When the compiler compiles the static class it’s treat as Abstract and Sealed class. This is the reason why neither we can create a instance nor extend a Static Class,

- The Singleton constructor always is marked as private. Reason why we can not create an instance outside the singleton Class. It provide a public static property or method who job is create the singleton instance only once and return the singleton instance each and every time we called the public static property/method from outside the class.

- A singleton class can be initialized as Lazy Loading(on-Demand) or can be load automatically by the CLR when the program containing the Singleton class is loaded (Eager Loading). Whereas a Static Class is initialized when it’s loaded for the first time.

- Is not possible pass a Static Class as a method parameter, we can pass a Singleton instance as a method parameter.

- In C#, it is possible to implement interfaces, inherit from other classes and allow inheritance with Singleton class. These are not possible with a static class.

- We cannot implement the Dependency Injection design pattern using Static class because the static class is not interface driven.

Here is a list of few use cases of static classes.

* A Math class with all static methods. **Static classes are useful and provide an easy way to access its members that does not need to work differently for different objects.**
* When we know that the value of a class members will never change regardless of its objects.
* A DatabaseConfig class that may have members such as database name, server name, port number, and even a connection string. We know that these values will not change for objects.
* When we are using method that will not change at all we can use Static Class and static methods. i.e Method for calculating conversions. See Methods for Converting Temperatures in Abstract Learning project.
* App Configuration class that has all static settings about an app and the values of settings don’t change based on the objects or users.

1. **public** **static** **class** HistoryTeacher
2. {
3. // private fields
4. **private** **static** **string** name;
5. **private** **static** **string** school;
6. **private** **static** **int** rank;
7. **private** **static** **int** years;
8. // static properties
9. **public** **static** **int** Years { **get** => years; **set** => years = value; }
10. **public** **static** **int** Rank { **get** => rank; **set** => rank = value; }
11. **public** **static** **string** School { **get** => school; **set** => school = value; }
12. **public** **static** **string** Name { **get** => name; **set** => name = value; }
13. }

The following code example sets static property values.

1. HistoryTeacher.Name = "Mahesh Chand";
2. HistoryTeacher.Rank = 2;
3. HistoryTeacher.School = "Garnet Valley High School";
4. HistoryTeacher.Years = 5;

In the same way, you can access a static property by using the class name.

1. Console.WriteLine(HistoryTeacher.Name);