C# CODING STANDARDS

1. Naming Conventions

a. PascalCase : Used for class names, method names, and properties.

```
Eg: class MyClass
    {
    public string MyProperty { get; set; }

    public void MyMethod() { }
    }
```

b. camelCase : Used for local variables, method parameters, and private fields.

```
Eg:class MyClass
    {
        private int myField;

        public void MyMethod(int myParameter)
        {
            int myLocalVariable = 0;
        }
    }
```

c. Uppercase: Used for constants.

```
        Eg: const int MAX_SIZE = 100;
```

2. Formatting

a. Indentation : Use 4 spaces per indentation level.

```
        Eg: if (condition)
                {
                        DoSomething();
                }
```

b. Braces: Braces should be on a new line for classes, methods, and statements.

```
class MyClass
        {
```

```
                    void MyMethod()
                    {
                            if (condition)
                            {
                            DoSomething();
                            }
                    }
            }
```

3. Comments

Single-line comments: Use `//` for single-line comments.

```
// This is a single-line comment
```

Multi-line comments**: Use `/* ... */` for multi-line comments.

```
/* This is a
   multi-line comment */
```

4. File Organization

Single Class per File**: Each file should contain one class.

```
 // File: MyClass.cs
 public class MyClass
  {
 }
```

5. Namespace: Use namespaces to organize code.

```
namespace MyNamespace
  {
          public class MyClass
           {
          }
  }
```

6. Error Handling

Exception Handling: Use try-catch blocks for exception handling.

```
 try
 {
         //Code that might throw an exception
  }

 catch (Exception ex)
 {
         // Handle exception
  }
```

7. Avoid the use of underscore while naming identifiers

```
// Correct
public DateTime fromDate;
public String firstName;


// Avoid
public DateTime from_Date ;
public String first_Name ;
```

8. Avoid the use of System data types and prefer using the Predefined data types.

```
// Correct
int employeeId;
string employeeName;
bool isActive;


// Avoid
Int32 employeeId;
String employeeName;
Boolean isActive;
```

9. Always prefix an interface with letter I.

```
// Correct
public interface IAnimal
{
}
public interface ICar
{
}
// Avoid
public interface Animal
{
}
public interface Car
{
}
```

10. Always use the using keyword when working with disposable types. It automatically disposes the object when program flow leaves the scope.

```
using(var conn = new SqlConnection(connectionString))
{
   // use the connection and the stream
   using (var dr = cmd.ExecuteReader())
   {
    //
   }
}
```

11. Always declare the variables as close as possible to their use.

```
// Correct
String firstName = "Shubham";
Console.WriteLine(firstName);
//--------------------------

// Avoid
String firstName = "Shubham";
//--------------------------
//--------------------------
//--------------------------
```

```
Console.WriteLine(firstName);
```

12. Always declare the properties as private so as to achieve Encapsulation and ensure data hiding.

```
// Correct
private int employeeId { get; set; }

// Avoid
public int employeeId { get; set; }
```

13. Always convert type ‘string’to ‘int’when using Console.readLine method for reading integer Because The Console.ReadLine() method returns a string.

```
//correct
Console.WriteLine("Enter your age:");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Your age is: " + age);

//avoid
Console.WriteLine("Enter your age:");
int age = Console.ReadLine();
Console.WriteLine("Your age is: " + age);
```

14. Avoid using Abbreviations.

```
// Correct
UserGroup userGroup;
Assignment employeeAssignment;

// Avoid
UserGroup usrGrp;
Assignment empAssignment;
```

15. Declare all member variables at the top of a class, with static variables at the very top.

```
// Correct
public class Account
{
    public static string BankName;
    public static decimal Reserves;

    public string Number {get; set;}
    public DateTime DateOpened {get; set;}
    public DateTime DateClosed {get; set;}
    public decimal Balance {get; set;}

    // Constructor
    public Account()
    {
        // ...
    }
    }
```

16. Use singular names for enums.

```
// Correct
public enum Color
{
    Red,
    Green,
    Blue,
    Yellow,
    Magenta,
    Cyan
}
```

17. Do not explicitly specify a type of an enum or values of enums (except bit fields)

```
// Don't
public enum Direction : long
{
    North = 1,
    East = 2,
    South = 3,
    West = 4
}
```

```
// Correct
public enum Direction
{
 North,
 East,
 South,
West
}
```

18. Do not suffix enum names with Enum

```
// Don't
public enum CoinEnum
{
    Penny,
    Nickel,
    Dime,
    Quarter,
    Dollar
}

// Correct
public enum Coin
{
    Penny,
    Nickel,
    Dime,
    Quarter,
    Dollar
}
```