Academia International College

(Affiliated to Tribhuvan University)

Lab Report

on

Interface and Delegates

Submitted By:
Sagar Timalsena
Roll No:15847(23)

Submitted To:
Chandan Bhagat Gupta

Theory:

<u>Interface:</u>

• An interface is a programming structure that allows the computer to enforce certain properties on an object.

• Interface in C# is a blueprint of a class.

• It is like abstract class because all the methods which are declared inside the interface are abstract methods.

• Its implementation must be provided by class or struct. The class or struct which which implements the interface, must provide the implementation of all the methods declared inside the interface.

<u>Delegate:</u>

• In OOP, delegation refers to evaluating a member of one object in the context of another original object.

• A delegate is a reference type variable that holds the reference to a method. The reference can be changed at run time.

• Delegates are especially used for implementing events and the call-back methods.

• All delegates are implicitly derived from the System. Delegate class.

• Delegate declaration determines the methods that can be referenced by the delegate. A delegate can refer to a method, which has the same signature as that of the delegate.

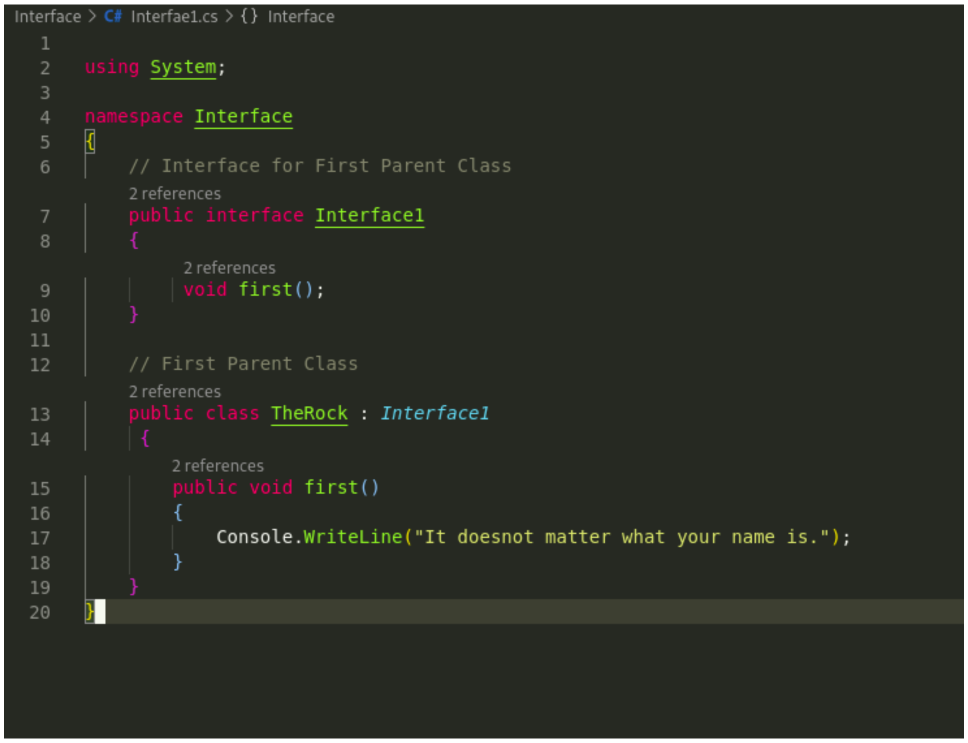
For example:

```
public delegate int Example (string s);
```

Code:

1. WAP to show the implementation of the multiple inheritance with the use of the interface.

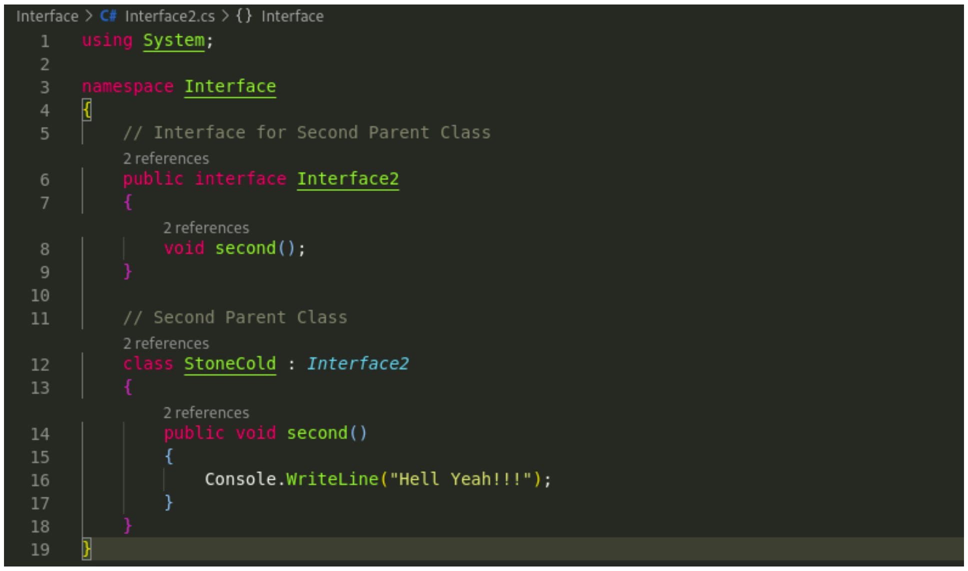
In Multiple inheritance, one class can have more than one superclass and inherit features from all its parent classes. But C# does not support multiple class inheritance. To overcome this problem we use interfaces to achieve multiple class inheritance.



Interface > C# Interfae1.cs > {} Interface

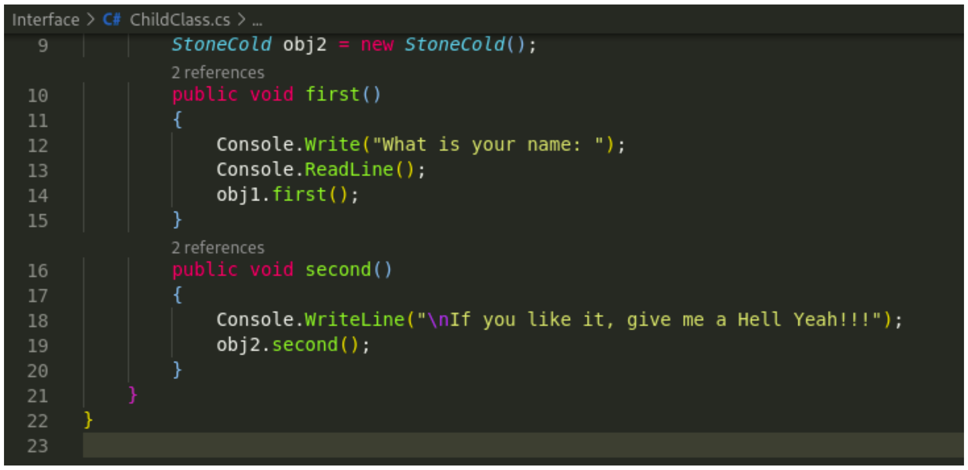
```

using System;

namespace Interface
{
    // Interface for First Parent Class
    2 references
    public interface Interface1
    {
        2 references
        void first();
    }

    // First Parent Class
    2 references
    public class TheRock : Interface1
     {
        2 references
        public void first()
        {
            Console.WriteLine("It doesnot matter what your name is.");
        }
    }
}
```

Sagar Timalsena



Interface > C# Interface2.cs > {} Interface

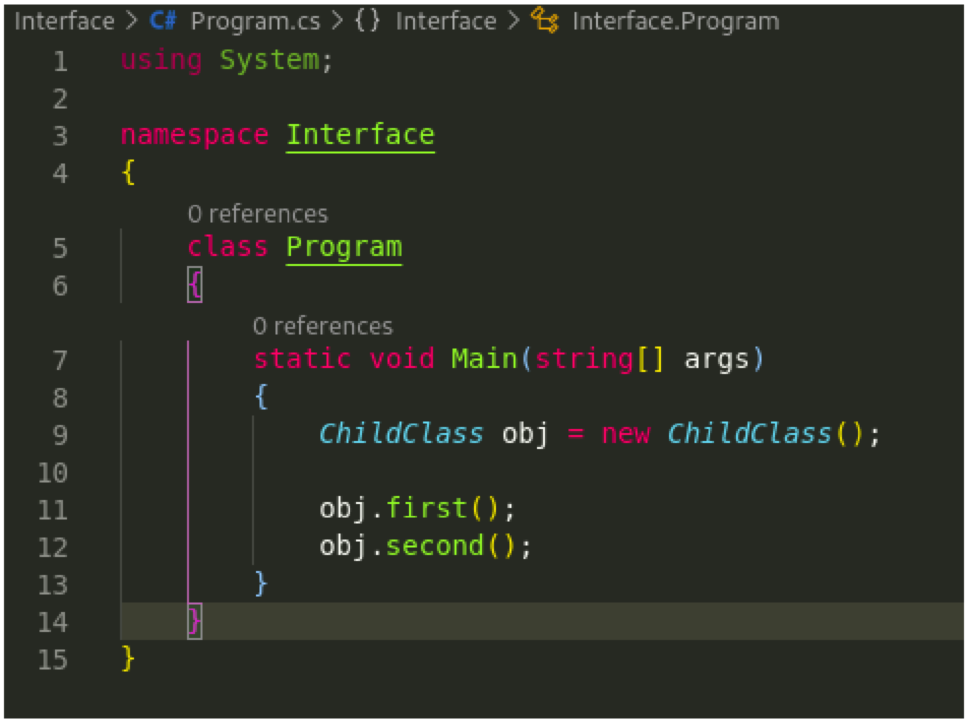
```
using System;

namespace Interface
{
    // Interface for Second Parent Class
    2 references
    public interface Interface2
    {
        2 references
        void second();
    }

    // Second Parent Class
    2 references
    class StoneCold : Interface2
    {
        2 references
        public void second()
        {
            Console.WriteLine("Hell Yeah!!!");
        }
    }
}
```



Interface > C# ChildClass.cs > ...

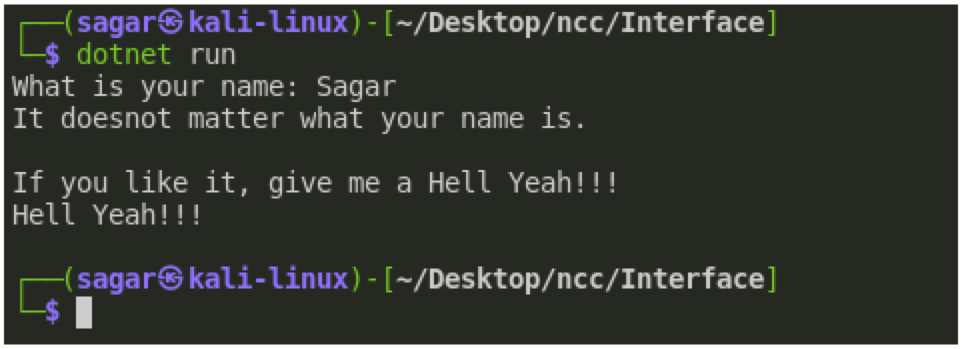
```
        StoneCold obj2 = new StoneCold();
        2 references
        public void first()
        {
            Console.Write("What is your name: ");
            Console.ReadLine();
            obj1.first();
        }
        2 references
        public void second()
        {
            Console.WriteLine("\nIf you like it, give me a Hell Yeah!!!");
            obj2.second();
        }
    }
}

```

Sagar Timalsena


Interface > C# Program.cs > {} Interface > Interface.Program

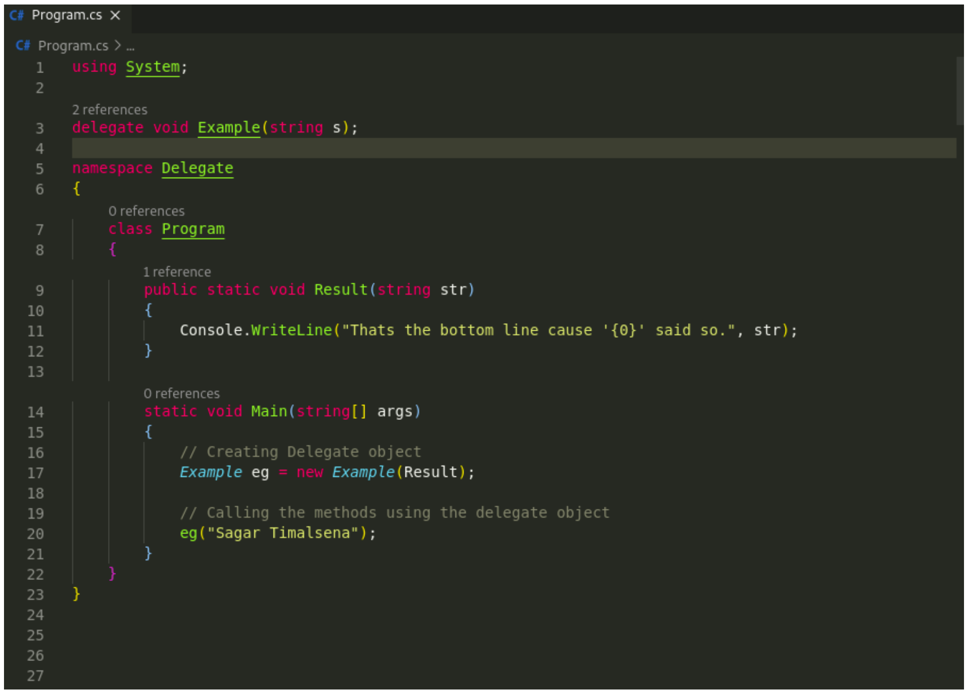
```csharp
using System;

namespace Interface
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            ChildClass obj = new ChildClass();

            obj.first();
            obj.second();
        }
    }
}
```

Output:



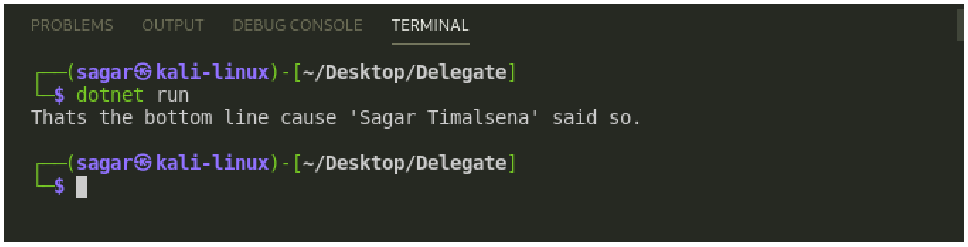
```
┌──(sagar㉿kali-linux)-[~/Desktop/ncc/Interface]
└─$ dotnet run
What is your name: Sagar
It doesnot matter what your name is.

If you like it, give me a Hell Yeah!!!
Hell Yeah!!!

┌──(sagar㉿kali-linux)-[~/Desktop/ncc/Interface]
└─$ 
```

2. WAP that reflects the Delegate and Events.



```
using System;

2 references
delegate void Example(string s);

namespace Delegate
{
    0 references
    class Program
    {
        1 reference
        public static void Result(string str)
        {
            Console.WriteLine("Thats the bottom line cause '{0}' said so.", str);
        }

        0 references
        static void Main(string[] args)
        {
            // Creating Delegate object
            Example eg = new Example(Result);

            // Calling the methods using the delegate object
            eg("Sagar Timalsena");
        }
    }
}
```

Output:



```
┌──(sagar㉿kali-linux)-[~/Desktop/Delegate]
└─$ dotnet run
Thats the bottom line cause 'Sagar Timalsena' said so.

┌──(sagar㉿kali-linux)-[~/Desktop/Delegate]
└─$ 
```

Conclusion:

Hence, we implemented the basic concept of OOP using C# including the four fundamentals principles of OOP: Encapsulation, Abstraction, Inheritance and Polymorphism.

GitHub Repository:

All the above codes used in this report are uploaded in GitHub Repository:

Github Project Link