Рад са више интерфејса

Класа може да има највише једну основну класу, али је дозвољено да примењује неограничени број интерфејса.
Та класа мора да примењује све методе које су декларисане тим интерфејсима.
Ако структура или класа примењује више интерфејса, ти интерфејси се навод екао списак, раздвојен зарезима.
Ако класа има основну класу, интерфејси се наводе после основне класе.
Нпр, дефинише се још један интерфејс под називом Iprezivari, који садржи методу ZvakanjeTrave за све сисаре који су преживари.
Класа Konj се може дефинисати:

```
class Konj: Sisari, ISuvozemci, IPrezivari
{
      ...
}
```

Изричита примена интерфеса

До сада класе су примењивале подразумевани интерфејс.
Иако класа Konj примењује интерфејс ISuvozemni, ништа у примени методе BrojKraka у класи Konj не каже да је она део део интерфејса Isuvozemni:

```
interface ISuvozemni
{
     int BrojKraka();
}
class Konj: ISuvozemni
{
      ...
      public int BrojKraka()
      {
               return 4;
      }
}
```

Проблем може да настане ако класа примењује више интерфејса.
Свако од интерфејса може да има методу са истим именом али са другачијим значењем.
Нпр, потребно је применити систем превоза који се заснива на кочијама које вуку коњи.
Дуже путовање може да буде подељено на више етапа (крака), и ако треба имати податке колико је етапа неки коњ вукао кочију, треба дефинисати следећи интерфејс:

```
interface IPutovanje
{
     int BrojKraka();
}
class Konj: ISuvozemni. IPutovanje
{
      ...
      public int BrojKraka()
      {
```

```
                return 4;
        }
}
```

Код је исправан али да ли то значи да коњ има 4 ноге или је вукао кочију током 4 етапе путовања?
Решење овог проблема се састоји у засебном примењивању различитих интерфејса.
Треба навести ком интерфејсу припада која метода:

```
class Konj: ISuvozemni, IPutovanje
{
      ...
       int ISuvozemni.BrojKraka()
       {
                return 4;
       }
       int IPutovanje.BrojKraka()
       {
              return 3;
       }
}
```

Поред тога што се пре назива методе наводи назив интерфејса, постоји још једна разлика у овој синтакси: методе се не означавају као јавне.
То води до тога да ако се у коду направи промењива konj, не може се позвати ниједна од метода BrojKraka(), пошто оне нису видљиве.
Што се тиче класе Konj, оне су приватне.

```
Konj konj = new Konj();
…
IPutovanje putovanjeKonj = konj;
int krakaNaPutovanju = putovanjeKonj.BrojKraka();
ISuvozemni suvozemniKonj = konj;
int nogeKonja = suvozemniKonj.BrojKraka();
```

Пример: експлицитна имплементација интерфејса

```
using System;
namespace ProjekatCS002
{
    interface IFajl
    {
        void ReadFile();
        void WriteFile(string tekst);
    }

    class FileInfo : IFajl
    {
        void IFajl.ReadFile()
        {
```

```
            Console.WriteLine("Citam fajl");
        }

        void IFajl.WriteFile(string text)
        {
            Console.WriteLine("Upis u fajl");
        }

        public void Search(string text)
        {
            Console.WriteLine("Pretraga po fajlu");
        }
    }
    public class Program
    {
        static void Main()
        {
            IFajl fajl1 = new FileInfo();
            FileInfo fajl2 = new FileInfo();

            fajl1.ReadFile();
            fajl1.WriteFile("sadrzaj");
            fajl2.Search("tekst koji treba da se pronadje");
        }
    }
}
```

У примеру објекат fajl1 може приступити само члановима Ifajl док fajl2 може приступити само члановима FileInfo класе.
Ово представља ограничење експлицитне имплементације.

Пример: имплементација вишеструких интерфејса

```
using System;
namespace ProjekatCS002
{
    interface IFajl
    {
        void ReadFile();
    }
    interface IBinarniFajl
    {
        void OpenBinaryFile();
        void ReadFile();
    }

    class FileInfo : IFajl, IBinarniFajl
    {
```

```
        void IFajl.ReadFile()
        {
            Console.WriteLine("Cita tekst fajl");
        }

        void IBinarniFajl.OpenBinaryFile()
        {
            Console.WriteLine("Otvara binarni fajl");
        }

        void IBinarniFajl.ReadFile()
        {
            Console.WriteLine("Cita binarni fajl");
        }

        public void Search(string text)
        {
            Console.WriteLine("Pretrazuje po fajlu");
        }
    }

    public class Program
    {
        public static void Main()
        {
            IFajl fajl1 = new FileInfo();
            IBinarniFajl fajl2 = new FileInfo();
            FileInfo fajl3 = new FileInfo();
            fajl1.ReadFile();
            fajl2.OpenBinaryFile();
            fajl2.ReadFile();
            fajl3.Search("tekst koji se trazi");
        }
    }
}
```

У примеру FileInfo имплементира два интерфејса IFajl и IbinarniFajl експлицитно.
Саветује се да се интерфејси имплементирају експлицитно када се имплементирају вишеструки интерфејси да би се избегли проблеми и лакше пришло коду.