Разумевање интерфејса

Права снага наслеђивања потиче из наслеђивања интерфејса.
Интерфејс не садржи код или податке, њиме се само наводе методе и својства која класа која наслеђује интерфејс мора да обезбеди.
Коришћењем интерфејса можете потпуно да издвојите називе и потписе метода из неке класе од начина на који је та метода остварена.

Нека је потребно дефинисати нову класу у којој се чува колекција објеката, слично низу.
Али, за разлику од низа овде желимо да понудимо методу под називом PreuzimanjePoRedu, којом би апликацији омогућили да објекте преузима по редоследу који зависи од типа објеката које одређена колекција садржи.
Нпр, ако нека колекција садржи алфанумеричке објекте као што су стрингови, та колекција треба да омогући да апликација преузима те стрингове по редоследу који одговара редоследу рећања који се користи на рачунару.
Ако та колекција садржи нумеричке објекте као што су цели бројеви, колекција треба да омогући апликацији преузимање објеката по нумеричком редоследу.

Како да класи колекције обезбедите методу која сортира објекте чије типове не знате када пишете класу колекције?
Виртуелна метода овде није примењива јер не постоји однос наслеђивања између класе колекције и објеката које та клас садржи и зато што начин поређивања објеката у колекцији зависи од типа објеката у колекцији а не од саме колекције.
Решење је да се захтева да сви објекти понуде методу као што је метода PorediSa коју метода PreuzimanjePoRedu из колекције може да позове:

```
int PorediSa(object obj)
{
     //vrati 0 ako je ova instanca jednaka sa obj
     //vrati < 0 ako je ova instanca manja od obj
     //vrati > 0 ako je ova instanca veca od obj
     //...
}
```

Може да се дефинише интерфејс за објекте из колекције који обухвата методу PorediSa и назначите да одређена класа колекције може да садржи само класе које примењују тај интерфејс.
На тај начин, интерфејс је сличан уговору.
Ако нека класа примењује неки интерфејс, тај интерфејс гарантује да та класа садржи све методе које су наведене у том интерфејсу.
Овај механизам омогућава да се позове метода PorediSa на све објекте у колекцији и да их сортира.
Интерфејс даје само назив, тип вредности која се враћа и параметре методе.
Начин на који се метода примењује није проблем интерфејса јер интерфејс описује функционалност коју класа треба да обезбеди а не како се та функционалност остварује.

Дефинисање интерфејса

Користи се службена реч interface; унутар интерфејса се декларишу методе исто као у класи или структури, осим што никада се не наводи модификатор приступа (public, private, protected).
Методе у интерфејсу немају примену, оне су једноставне декларације а сви типови који примењују интерфејс морају да понуде сопствену примену; тело методе се замењује са ;.

```
interface IUporedivo
{
      int PorediSa(object obj);
}
```

У званичној документацији за .NET препоручује се да пре назива интерфејса се користи велико слово I.
Интерфејс не сме да садржи било какве податке и у њему нема поља.

Примена интерфејса

Да би се применио неки интерфејс, мора да се декларише класа или структура која наслеђује интерфејс и која примењује све методе које наводи интерфејс.
То заправо и није наслеђивање, мада је синтакса иста и доста подсећа на наслеђивање.

Нпр, треба дефинисати хијерархију Sisari, али је пре тога потребно назначити да копнени сисари нуде методу под називом BrojKraka која враћа целобројну вредност ногу коју неки сисар има.
Може се дефинисати интерфејс IKopneni који садржи ову методу:

```
interface IKopneni
{
      int BrojKraka();
}
```

Овај интерфејс се сада може применити у класи Konj; наслеђује се овај интерфејс и нуди се примена свих метода које дефинишу тај интерфејс:

```
class Konj: IKopneni
{
      //...
      public int BrojKraka()
      {
            return 4;
      }
}
```

Када се примењује интерфејс, мора се осигурати да се све методе потпуно подударају са њима одговарајућим методама интерфејса, према следећим правилима:

- Назив методе и тип вредности која се враћа се потпуно подударају
- Сви параметри (укључујући ref и out) се потпуно подударају
- Све методе које остварују интерфејс морају да буду јавно доступне; међутим, ако се интерфејс изричито примењује, методе не могу да имају квалификатор приступа

У VS постоји чаробњак Implement Interface којим се генерише костур за све ставке у интерфејсу које класа примењује.

Класа може да наследи другу класу и примени интерфејс истовремено.
У том случају се у C# не прави разлика између основне класе и интерфејса коришћењем одређених службених речи већ се користи позициона нотација.
Назив основне класе се увек прво наводи, после чега следи зарез, а затим назив интерфејса.
У примеру, класа Konj је класа која је Sisari, али која додатно примењује интерфејс Ikopneni:

```
interface IKopneni
{
     //...
}
class Sisari
{
     //...
}
class Konj: Sisari, IKopneni
{
     //...
}
```

Неки InterfejsA може да наследи други интерфејс interfejsB и то се назива проширење интерфејса.
У овом случају све класе или структуре које примењују InterfejsA морају да обезбеде примену свих метода у интерфејсима InterfejsB и InterfejsA.

Пример: Креирати класу која наслеђује интерфејс имплицитно за рад са текстуалним фајловима

```
using System;
interface IFajl
{
    void ReadFile();
    void WriteFile(string text);
}

class FajlInfo : IFajl
{
    public void ReadFile()
    {
        Console.WriteLine("Citanje iz fajla");
    }
    public void WriteFile(string text)
    {
        Console.WriteLine("Upis u fajl");
    }
}
public class Program
{
    public static void Main()
    {
        IFajl fajl1 = new FajlInfo();
```

```
        FajlInfo fajl2 = new FajlInfo();

        fajl1.ReadFile();
        fajl1.WriteFile("Sadrzaj1");

        fajl2.ReadFile();
        fajl2.WriteFile("Sadrzaj2");
    }
}
```

У примеру, класа FajlInfo имплементира интерфејс Ifajl.
Класа дефинише све чланове интерфејса помоћу јавног модификатора приступа.
Класа такође може садржати чланове који нису чланови интерфејса.
Унутар главне функције, креиран је објекат класе (fajl1) и њему је додељена промењива типа интерфејса, као и промењива типа саме класе (fajl2).
Када се интерфејс имплементира имплицитно, може се приступити члановима интерфејса помоћу промењивих типа интерфејса као и помоћу промењивих типа класе која имплицитно имплементира интерфејс.

Указивање на класу преко интерфејса

На исти начин на који се може укази на неки објекат коришћењем промењиве дефинисане као класа која је виша у хијерархији, може се указати на објекат коришћењем промењиве дефинисане као интерфејс, који објекат те класе примењује.
Може се указати на објекат Konj коришћењем промењиве Isuvozemni:

```
Konj mojKonj = new Konj(...);
ISuvozemni iMojKonj = mojKonj;
```

Пример је могућ пошто су сви коњи сувоземни сисари, мада обрнуто није тачно – не може се објекат Isuvozemni доделити промењивој Konj без претходне промене типа како би се проверило да ли је то стварно референца објекта Konj, а не нека друга класа која такође случајно примењује интерфејс Isuvozemni.
Техника указивања на објекат преко интерфејса је корисна, пошто с еможе користити за дефинисање метода које могу да узимају различите типове као параметре све док ти типови примењују тачно одређени интерфејс.
Нпр, метода BrzinaKretanja може да узима било који аргумент који примењује интерфејс Isuvozemni:

```
int BrzinaKretanja(ISuvozemni SuvozemniSisari)
{
      ...
}
```

Да ли је неки објакт инстанца класе која примењује одређени интерфејс, може се проверити коришћењем оператора is .

Оператор is се користи за одређивање да ли неки објекат има одређени тип, а то може да с еради са интерфејсима, класама и структурама.
Нпр, следећи блок проверава да ли промењива mojKonj стварно примењује интерфејс iSuvozemni пре него покуша да га додели промењивој Isuvozemni:

```
if (mojKonj is ISuvozemni)
{
     ISuvozemni iSuvozemniZivotinja = mojKonj;
}
```