Апстрактне класе

Интерфејси Isuvozemci, Iprezivari се могу применити у више различитих класа.

Честа је појава да делови изведених класа деле заједничке примене (нпр иста метода у две различите класе).

```
class Konj: Sisar, ISuvozemni, IPrezivari
{
    ...
    void IPrezivari.PaseTravu()
    {
        Console.WriteLine("Pase travu");
        //kod metode
    }
}
class Ovca : Sisar, ISuvozemni, IPrezivari
{
    ...
    void IPrezivari.PaseTravu()
    {
        Console.WriteLine("Pase travu");
        //isti kod metode
    }
}
```

Али, понављање у коду је знак упозорења да треба прерадити код, како би се избегло непотребно понављање и тако смањити послове око одржавања програма.

Један од начина за прераду кода јесте да се заједничка примена смести у нову класу која се прави посебно за ову сврху.

Заправо, треба уметнути нову класу у хијерархију класа:

```
class SisarPrezivar : Sisar, IPrezivari
{
    ...
    void IPrezivari.PaseTravu()
    {
        Console.WriteLine("Pase travu");
        //zajednicki kod metode
    }
}
class Konj: SisarPrezivar, IPrezivari
{
    ...
}
class Ovca : SisarPrezivar, IPrezivari
{
    ...
}
```

Иако је ово добро решење, проблем је то што се на овај начин прави инстанце класа SisarPrezivar и Sisar, што нема смисла.
Класа SisarPrezivar постоји како би понудила заједничку подразумевану примену.
Њена једина сврха јесте да буде класа која треба да се наследи.
Класа SisarPrezivar је само апстракција заједничке функционалности, а не посебна независна целина.
Да би с еназначило да прављење инстанце од неке класе није дозвољено, може се декларисати да је та класа апстрактна коришћењем кључне речи abstract:

```
abstract class SisarPrezivar : Sisar, IPrezivari
{
    ...
}
```

Сада програм неће допустити компајлирање ако се покуша креирати објекат типа SisarPrezivar.

Апстрактне методе

Апстрактне класе могу да садрже апстрактне методе.
Апстрактна метода је метода која не садржи тело методе.
Изведена класа мора да надјача ту методу а апстрактна метода не сме да буде приватна.
У следећем примеру метода VarenjeTrave је апстрактна метода у класи SisarPrezivar тако да сва врста животиње која је и сисар и преживар мора да обезбеди споствену примену апстрактне методе.
Апстрактна метода је корисна када нема смисла да се подразумевана примена у апстрактној класи и када у исто време се жели да будемо сигурни да ће класа која је наслеђује понудити сопствену примену ове методе:

```
abstract class SisarPrezivvar: Sisar, IPrezivari
{
     public abstract void VarenjeTrave()
      {
           ...
      }
}
```

Пример:

```
using System;
namespace ProjekatCS002
{
    abstract class Zivotinja
    {
        public abstract void ZvukZivotinje();
        public void Spavanje()
        {
            Console.WriteLine("zzz");
        }
    }
    class Svinja : Zivotinja
```

```
    {
        public override void ZvukZivotinje()
        {
            Console.WriteLine("Svinja kaze: oink, oink");
        }
    }
    public class Program
    {
        public static void Main()
        {
            Svinja mojaSvinja = new Svinja();
            mojaSvinja.ZvukZivotinje();
            mojaSvinja.Spavanje();
        }
    }
}
```

Даје:
Svinja kaze: oink, oink
zzz

Запечаћене класе

Коришћење наслеђивања није често једноставно, ако се прави интерфејс или апстрактна класа, свесно се планира унапред примена наслеђивања, што не мора да буде једноставан посао у будућности.
Осим ако се нека класа свесно не осмисли са намером да се користи као основна класа, мало је вероватно да ће добро функционисати као основна класа.
У C# се може користити кључна реч sealed како би се спречило да се нека класа користи као основна класа, ако се претходно одлучи да она то и не буде:

```
sealed class Konj: SisarPrezivar, ISuvozemni
{
    ...
}
```

Ако нека класа покуша да искористи класу Konj као основну класу, биће генерисана грешка приликом компајлирања.

Запечаћене методе

Кључна реч sealed се користи за декларисање када је нека посебна метода у незапечаћеној класи запечаћена.
То значи да изведена класа не може да надјача ту методу.
Можете да запечатите само методу која је декларисана кључном речи override а методу декларишете као sealed override.
Кључне речи interface, virtual, sealed можете да замислите:

- Интерфејс представља назив методе
- Виртуелна метода је прва примена методе
- Надјачана метода је још једна примена методе

- Запечаћена метода је последња примена методе

Пример:

```
using System;
namespace ProjekatCS002
{
    class Class1
    {
        static void Main()
        {
            ZapecacenaKlasa klasa = new ZapecacenaKlasa();
            int suma = klasa.Sabiranje(4, 5);
            Console.WriteLine("Ukupno = " + suma.ToString());
        }
    }
    sealed class ZapecacenaKlasa
    {
        public int Sabiranje(int x, int y)
        {
            return x + y;
        }
    }
}
```