**Vad är ett interface?**

Ett interface är en typ som deklarerar en eller flera metoder och/eller egenskaper, men implementerar dem inte. Andra typer, som klasser, kan implementera ett eller flera interface. Det går inte att instansiera ett objekt från ett interface.

Ett interface representerar hur ett objekt ska kunna användas, inte hur det implementeras i detalj av en klass. I ett interfaces kropp deklareras metoder (och egenskaper) på samma sätt som i en vanlig klass, med den skillnaden att någon modifierare, som t.ex. private eller public, aldrig anges och metodkroppen ersätts med ett semikolon.

I dotnetramverket finns interfacet IComparable deklarerad, enligt koden som följer.

public interface IComparable

{

int CompareTo(object obj);

}

Interfacet innehåller en deklaration av metoden CompareTo. Metoden har en returtyp (int), ett namn (CompareTo) och en parameterlista (object obj) men saknar alltså en metodkropp som har ersatts med ett semikolon.

Av onlinedokumentationen framgår att metoden CompareTo används till att jämföra två objekt av samma typ med varandra och vad metoden ska returnera.

* Ett negativt heltal om det anropade objektet är mindre än objektet som parametern refererar till.
* Ett positivt heltal om det anropade objektet är större än objektet som parametern refererar till.
* Noll, om det två objekten anses vara lika.

Ett exempel får visa hur interfacet IComparable används i praktiken.

Följande kod skapar en array och initierar den med referenser till strängar, som sorteras och skrivs ut.

string[] names = new string[3] { "Bertil", "Ceasar", "Adam" };

Array.Sort(names);

foreach (string name in names)

{

Console.WriteLine(name);

}

Denna kod skriver ut följande:



Figur 1.

Metoden Sort fungerar med många av typerna som finns i dotnetramverket, som t.ex. string och int. Men vad händer om du har en egen typ, en egen klass du själv skapat?

class PersonDetails

{

private int \_age;

private string \_name;

public PersonDetails(int age, string name)

{

\_age = age;

\_name = name;

}

public override string ToString()

{

return String.Format("{0}, {1}", \_name, \_age);

}

}

static void Main(string[] args)

{

PersonDetails[] persons = new PersonDetails[3];

persons[0] = new PersonDetails(8, "Johanna");

persons[1] = new PersonDetails(1, "Filippa");

persons[2] = new PersonDetails(6, "Emma");

Array.Sort(persons); // kastar ett undantag!

...

}

Då koden körs kastar metoden Sort ett undantag. Anledningen till detta är att Sort inte vet hur objekt instansierade av klassen PersonDetails ska jämföras med varandra.

Sort fungerar då objekt av typen String ska sorteras eftersom klassen String implementerar interfacet IComparable. Då Sort exekveras anropas metoden CompareTo och det anropande objektet jämförs med det objekt parametern refererar till.

För att objekt av typen PersonDetails ska kunna jämföras och sorteras måste typen implementera interfacet IComparable. Detta innebär att klassens kropp måste kompletteras med metoden CompareTo. Följande kod visar hur klassen PersonDetails implementerar interfacet IComparable.

class PersonDetails : **IComparable**

{

...

public int CompareTo(object obj)

{

if (obj == null)

{

return 1;

}

PersonDetails other = obj as PersonDetails;

if (other == null)

{

throw new ArgumentException("obj is not a PersonDetails");

}

return \_name.CompareTo(other.\_name);

}

}

Först undersöks om parametern obj refererar till ett objekt eller null. Refererar obj till null ska det anropande objektet sorteras efter null varför ett värde större än noll returneras.

Satsen PersonDetails other = obj as PersonDetails; typomvandlar referensen obj från object till PersonDetalis. Går det inte att typomvandla obj från en referens av typen object till en referens av typen PersonDetails tilldelas other värdet null. Refererar inte obj till ett PersonDetails-objekt kastas ett undantag av typen ArgumentException eftersom det bara är objekt av typen PersonDetails som kan jämföras.

Metoden avslutas med att värdena fälten \_name har jämförs med hjälp av metoden CompareTo som klassen String implementerar.

Koden

static void Main(string[] args)

{

PersonDetails[] persons = new PersonDetails[3];

persons[0] = new PersonDetails(8, "Johanna");

persons[1] = new PersonDetails(1, "Filippa");

persons[2] = new PersonDetails(6, "Emma");

Array.Sort(persons); // kastar INTE ett undantag

foreach (PersonDetails pd in persons)

{

Console.WriteLine(pd);

}

}

kastar nu inget undantag utan följande skrivs ut i konsolfönstret:



Figur 2.

De tre PersonDetails-objekten har sorteras med avseende på fältet \_name.