

# C# - Interface

Ett gränssnitt(interface) är en klass med abstrakta metoder, instansvariabler, events och kan ses som ett kontrakt som en eller fler klasser vill använda sig av. Ett gränssnitt är väldigt lik abstrakta klasser i det att en klass som ärver av dessa måste implementera de abstrakta klassmedlemmarna(metoder, fält osv), dock använder vi abstrakta klasser till de basklasser som vi inte fullt kan definiera medans ett gränssnitt ses som en utökning av en klass och en klass kan använda sig av flera gränssnitt.

Gränssnitt skapar vi med nyckelordet "interface" därefter ett namn som börjar med prefixen "I" och ett beskrivande adjektiv, exempelvis "IMovable". Innanför våra klammerparenteser deklarerar vi det som behövs. Precis som med abstrakta metoder så avslutar vi metoden efter vi definierat returtyp och namn (exempel: `void Move();`). Klasser som ärver från gränssnittet måste uppfylla "kontraktet", det vill säga implementera gränssnittets medlemmar.

## En boll vi kan sparka iväg

Som exempel ska vi skapa en boll som vi ska kunna sparka iväg och detta ska vi göra med hjälp av ett gränssnitt. Vi börjar med att högerklicka på vårt projektnamn och trycker "Add" och letar på "Interface" och ger denna namnet "IKickable". Därefter deklarerar vi "Force" som är av typen float för kraft samt en metod för att lägga till kraften, "AddForce".

```
// Vårt interface för objekt vi kan sparka på
interface IKickable
{
    // "Kontraktet" kräver en kraft bollen sparkas i
    // och därefter en metod som adderar kraften till bollen
    float Force { get; set; }
    void AddForce();
}
```

Nu kan vi skapa klassen "Ball" och definiera den. Vi deklarerar en sträng för bollens färg och ärver från "IKickable" och implementerar dessa. I konstruktorn tar vi in en sträng för färg och en float för kraft.

```
// Klassen Ball implementerar "kontraktet" IKickable
class Ball : IKickable
{
    public float Force { get; set; }    // Kontraktets variabel
    string Color { get; set; }         // Klassens variabel

    // Konstruktör
    public Ball(string color, float force)
    {
        Color = color;
        Force = force;
    }

    // Kontraktets metod
    public void AddForce()
    {
        Console.WriteLine($"You kick the {Color} ball with a force of {Force}N.");
    }
}
```

Nu när vi uppfyllt "kontraktet" så kan vi sparka iväg bollen. Vi skapar ett objekt av "Ball" och ger den färgen röd och en kraft på 6.521N. Sedan skapar vi en if-sats för att kolla om bollen kan sparkas, och isåfall vill vi lägga till kraften till bollen och den sparkas iväg.

```
static void Main(string[] args)
{
    // Skapa ett boll objekt med färgen röd och
    // en kraft på 6.521N
    Ball ball = new Ball("Red", 6.521F);

    // Här kollar vi om vårt objekt har
    // IKickable-interface. Om den har det
    // så kan vi använda IKickables metod AddForce
    if (ball is IKickable) ball.AddForce();

    Console.ReadLine();
}
```

Resultat:

```
E:\C# Projects\GitHub\Interfaces\Interfaces\bin\Debug\Interfaces.exe
You kick the Red ball with a force of 6,521N.
```

Med hjälp av gränssnitt kan vi skapa mer flexibla och säkrare kod. För att ge er en bättre bild av hur och när ni kan använda er av gränssnitt för att underlätta arbetet kan vi gå igenom ytterligare ett exempel på hur de kan användas.

## Spelprogrammering: kanin vs fiende

Vi ska nu kunna interagera med en kanin och en fiende så vi börjar med att implementera ett gränssnitt vi kallar "ITargetable" för dessa objekt.

```
interface ITargetable
{
    void Target();
}
```

Nu skapar vi två klasser, "Bunny" och "Skeleton", ärver från "ITargetable" och implementerar dessa.

Bunny:

```
class Bunny : ITargetable
{
    string Name { get; set; }

    public Bunny (string name) { Name = name; }

    public void Target() // Gränssnittets metod
    {
        Console.WriteLine($"You target the friendly bunny named {Name}.");
    }
}
```

Skeleton:

```
class Skeleton : ITargetable
{
    string Name { get; set; }

    public Skeleton (string name) { Name = name; }

    public void Target() // Gränssnittets metod
    {
        Console.WriteLine($"You target the enemy skeleton named {Name}.");
    }
}
```

Vi måste nu åtskilja dessa två då "Bunny" är en fredlig kanin medans "Skeleton" är en fiende vi måste döda. Detta kan vi göra genom att skapa ett nytt gränssnitt och döpa den till "IEnemy" och deklarerera metoden "Kill()".

```
interface IEnemy
{
    void Kill();
}
```

För varje fiende vi skapar i framtiden kan vi nu ge "kontraktet" "IEnemy" för att åtskilja fiender från fredliga varelser.

```
class Skeleton : ITargetable, IEnemy
{
    string Name { get; set; }

    public Skeleton (string name) { Name = name; }

    public void Target() // Gränssnittets metod
    {
        Console.WriteLine($"You target the enemy skeleton named {Name}.");
    }

    public void Kill()
    {
        Console.WriteLine($"You killed the skeleton named {Name}.");
    }
}
```

Som ni ser så har "Skeleton" två gränssnitt, "ITargetable" och "IEnemy".

Nu kan vi testa våra objekt. Vi placerar dessa i en lista och börjar med "Target":

```
// Skapa lista för creatures som har "Target"
List<ITargetable> creatureList = new List<ITargetable>();

creatureList.Add(new Bunny("Bernard")); // Lägg till Bunny
creatureList.Add(new Skeleton("Skelly Ton")); // Lägg till Skeleton

// För varje targetable creature i listan
foreach (ITargetable creature in creatureList)
{
    creature.Target(); // Anropa target
}
```

Resultat:


```
E:\C# Projects\GitHub\Interfaces\Interfaces\bin\Debug\Interfaces.exe
You target the friendly bunny named Bernard.
You target the enemy skeleton named Skelly Ton.
```

Nu testar vi att döda dom:

```
// För varje targetable creature i listan
foreach (ITargetable creature in creatureList)
{
    // Om creature också har "IEnemy"
    if (creature is IEnemy)
    {
        // Casta creature till en "IEnemy"
        IEnemy c = (IEnemy)creature;
        c.Kill(); // Anropa metoden "Kill()"
    }
    else
    {
        Console.WriteLine("You cant kill this creature");
    }
}
```

Här måste vi kolla om varelsen i listan också har gränssnittet "IEnemy" och isåfall castar vi objektet till en "IEnemy" och anropar "Kill()" metoden.

## Resultat:

 Välj E:\C# Projects\GitHub\Interfaces\Interfaces\bin\Debug\Interfaces.exe

```
You target the friendly bunny named Bernard.  
You target the enemy skeleton named Skelly Ton.  
You cant kill this creature  
You killed the skeleton named Skelly Ton.
```

Med hjälp av gränssnitt har vi nu sparat tid och arbete när det gäller att skilja på vänliga och fientliga varelser.