Der Konstruktor ist eine spezielle Funktion, welche bei der Initialisierung (also dem Erstellen) einer Klasse aufgerufen wird. Der Name der Konstruktor-Funktion entspricht immer dem Klassennamen. Die Funktion besitzt keinen Rückgabewert (auch nicht void), da der Konstruktor sozusagen das neu initialisierte Objekt zurückgibt. Wird kein Konstruktor in der Klasse erstellt, gibt es immer einen sogenannten Standard-Konstruktor, welcher keine weiteren Aktionen ausführt. Der Standard-Konstruktor besitzt keine Parameter.

```
class Haus
{
    private int anzahlSchornsteine;
}
```

Jedoch kann dies bei der Implementierung eigener Konstruktoren geändert werden. Des Weiteren ist auch die Überladung des Konstruktors möglich. Wichtig zu wissen ist, dass sobald Konstruktoren in der Klasse deklariert werden, es den Standard-Konstruktor den C# automatisch erstellt, nicht mehr gibt. Der Zugriffsmodifizierer bei Konstruktoren ist normalerweise public. Wird als Zugriffsmodifzierer private verwendet, ist die Objektinitialisierung dieser Klasse nicht mehr möglich. Der private Konstruktor wird deshalb verwendet, um die Objektinitialisierung einer Klasse, welche lediglich statische Funktionen enthält, zu unterbinden.

```
class Haus
 {
     private int anzahlSchornsteine;

     public Haus()
     {
         anzahlSchornsteine = 1;
         Console.WriteLine("Neues Objekt(Typ Haus) erstell!");
     }
 }
```

Der Destruktor ist das Gegenteil des Konstruktors, da der Destruktor aufgerufen wird, wenn das Objekt zerstört / gelöscht wird. Ein Objekt wird automatisch gelöscht, sobald das .NET Framework keine Referenzierung mehr findet.

Dieses Löschen der Objekte geschieht automatisch durch den Garbage Collector (kurz GC), wenn das Programm aktuell keine Vorgänge zu bearbeiten hat oder der Speicherplatz kritisch wird. Des Weiteren ist es auch möglich, über die statische Funktion Collect() der Klasse GC das Löschen unreferenzierter Objekte manuell auszulösen.

Der Name des Destruktors entspricht ebenfalls dem Klassennamen, jedoch mit einem vorangestellten Tilde-Zeichen (~). Beim Destruktor darf des Weiteren kein Zugriffsmodifizierer und Rückgabewert angegeben werden.

```
class Haus
 {
     private int anzahlSchornsteine;

     public Haus()
     {
         anzahlSchornsteine = 1;
         Console.WriteLine("Neues Objekt(Typ Haus) erstell!");
     }
     ~Haus()
     {
         Console.WriteLine("Objekt (vom Typen Haus) zerstört!");
     }
 }
```

Überladener Konstruktor:

```
class Haus
{
    private int anzahlSchornsteine;

    public Haus()
    {
        anzahlSchornsteine = 1;
        Console.WriteLine("Neues Objekt(Typ Haus) erstell!");
    }
    public Haus(int anzahlSchornsteine)
    {
        this.anzahlSchornsteine = anzahlSchornsteine;
    }

    ~Haus()
    {
        Console.WriteLine("Objekt (vom Typen Haus) zerstört!");
    }
}
```