**OOP-Grundlagen (1)**

**Objektorientierte Datenmodellierung**

**1:N-Beziehungen (One-to-Many):**

Eine **Company** kann mehrere **Products** anbieten.

Eine **Company** kann mehrere **Customers** haben.

Ein **Customer** kann mehrere **Orders** tätigen.

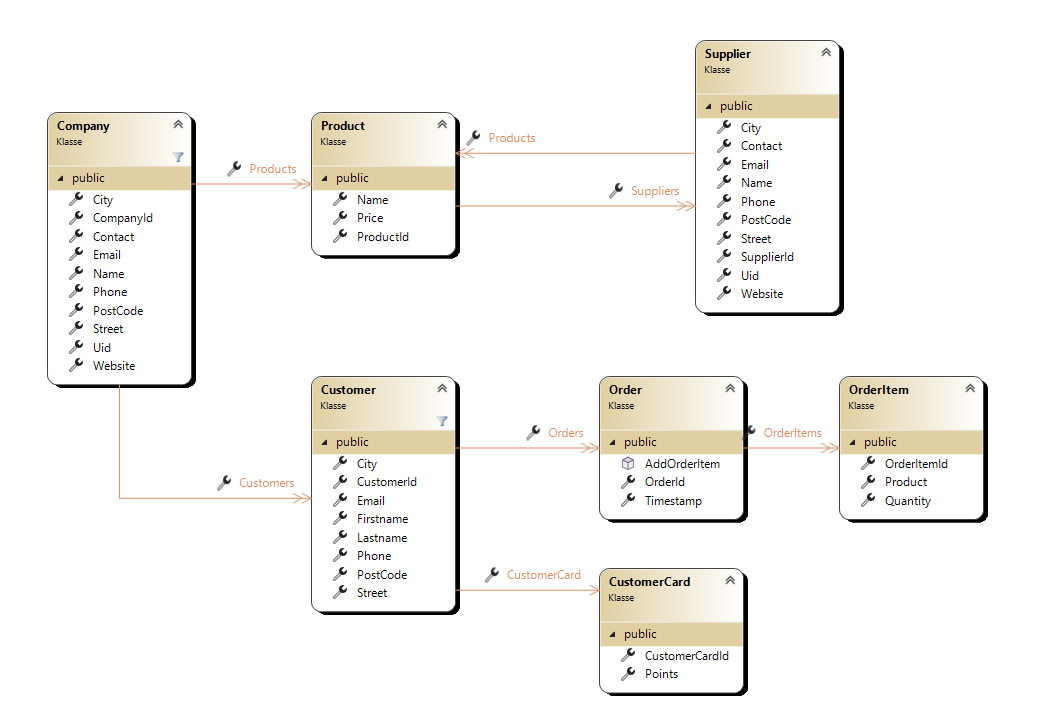
Ein **Order** kann mehrere **OrderItems** haben.

**1:1-Beziehungen (One-to-One):**

Ein **Customer** kann nur eine **CustomerCard** haben.

**N:M-Beziehungen (Many-to-Many)**

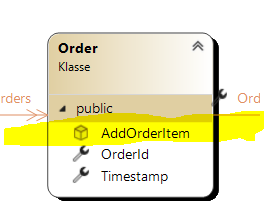
Mehrere **Suppliers** können mehrere **Products** liefern und mehrere **Products** können mehrere **Suppliers** als Lieferant haben.



**Methoden eines Objektes**

Methoden beschreiben Verhalten: Sie geben an, was ein Objekt **tun kann**, und werden aufgerufen, um eine bestimmte Aktion durchzuführen oder mittels **return** einen Wert abzurufen.

**Beispiel**

Methode **AddOrderItem** fügt dem Order-Objekt ein **OrderItem** hinzu.  


**Vererbung, IS-A- Prinzip und HAS-A-Prinzip**

Die Vererbung ermöglicht einer Klasse die **Eigenschaften** und **Methoden** einer anderen Klasse zu übernehmen (zu erben).

Die abgeleitete Klasse, die das Erbe bekommt nennt man **Unterklasse** oder **Kindklasse**. Die Erbe gebende Klasse bezeichnet man **Basisklasse** oder **Elternklasse.**

**Wozu?**

Vererbungfördert **logische Kapselung** und **Wiederverwendbarkeit** von Code und ermöglicht es, neue Klassen zu erstellen, die auf existierenden Klassen basieren und deren Funktionalität erweitern (Eine Kind Klasse hat die Eigenschaften und Methoden ihrer Elternklasse **plus** ihre eigene Eigenschaften und/oder Methoden).

**Beispiel zum gemeinsamen Programmieren:**

Basisklasse Baum

Baum **HAS A** Kronenform

Baum **HAS A** Hoehe

public class **Baum**

{

public string Kronenform { get; set; }

public int Hoehe { get; set; }

}

Klasse Ahorn bekommt von Baum implizit *Kronenform* und *Hoehe* als „Erbe“ und hat zusätzlich *Blattfarbe*.

Ahorn **IS A** Baum

Ahorn **HAS** Elemente seiner Basisklasse

Ahorn **HAS A** Blattfarbe

public class **Ahorn**: **Baum**

{

public string Blattfarbe { get; set; }

}

Klasse Tanne erbt von Baum *Kronenform* und *Hoehe* und hat zusätzlich *AnzahlZapfen*.

Tanne **IS A** Baum

Tanne **HAS** Elemente ihrer Basisklasse

Tanne **HAS A** AnzahlZapfen

public class **Tanne**: **Baum**

{

public int AnzahlZapfen { get; set; }

}

**Merke:**

Das **IS-A-Prinzip** (oder **"ist-ein"-Prinzip**) ist ein Grundsatz in der objektorientierten Programmierung, der beschreibt, wie Vererbung verwendet wird, um eine "ist-ein"-Beziehung zwischen einer Basisklasse und ihren abgeleiteten Klassen darzustellen.

Das **HAS-A-Prinzip** (oder **"hat-ein"-Prinzip**) beschreibt eine Beziehung zwischen Objekten, bei der ein Objekt ein anderes **besitzt** oder von ihm abhängt.

C# unterstützt – wie die meisten anderen Programmiersprachen nur **Einfachvererbung**. Das heißt ein Objekt kann nur von einem Objekt erben. Mehrfaches erben kann man durch Implementierung einer oder mehrerer **Interfaces** (Schnittstellen) „simulieren“.

**Merke:**

Ein Interface kann weder instanziiert noch geerbt werden. „Implementieren“ wäre hier das richtige Zeitwort.