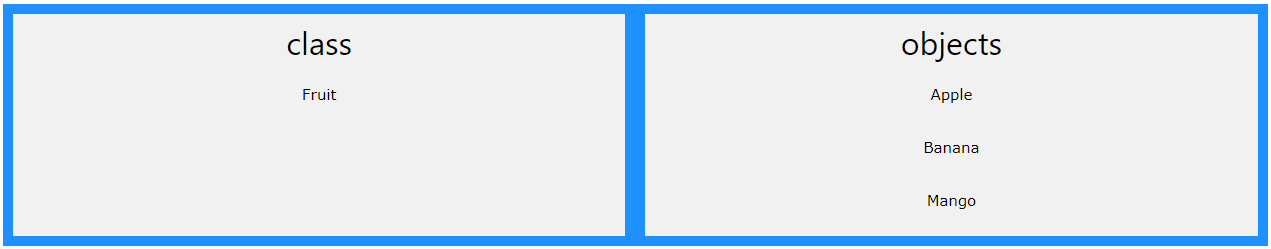
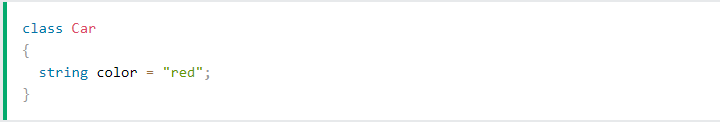
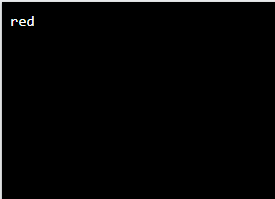
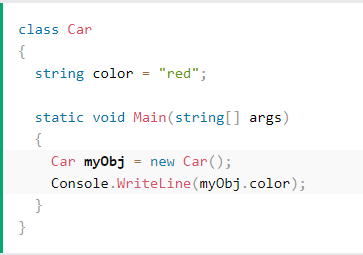
OOP in C#

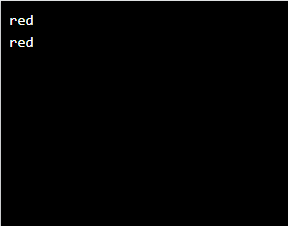
Unterschied zwischen Klassen und Objekten  


anderes Beispiel:

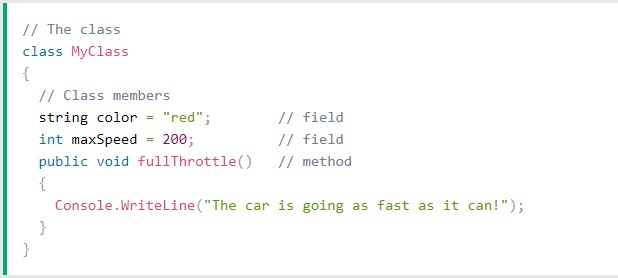


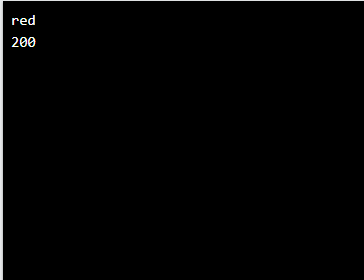
Eine Klasse mit dem Namen Car und der Variabel Color erstellen.  


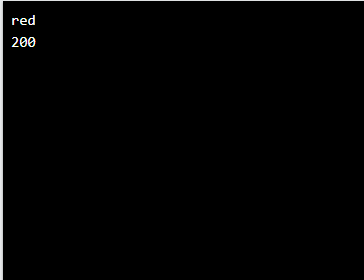
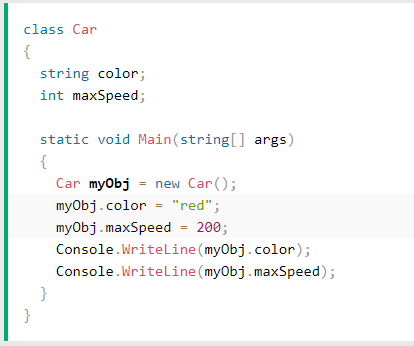
Ein Objekt wird von einer Klasse erstellt. Nun wird ein Objekt namens myObj erstellt und wird gebraucht, um den Wert color auszugeben.  


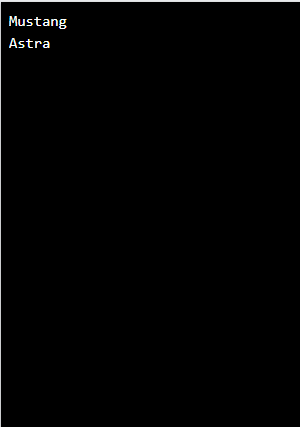
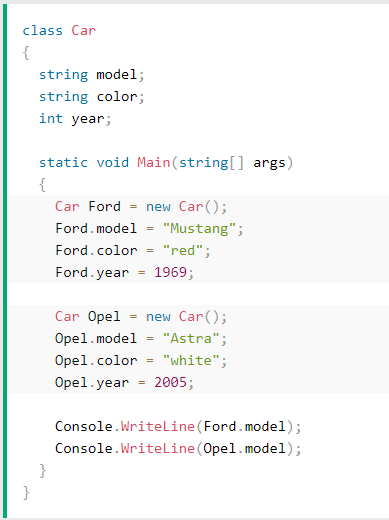
Nun werden zwei Objekte von der Klasse car erstellt.  


Man kann auch ein Objekt von einer Klasse erstellen und die Variabel von einer anderen Klasse zugreifen.  

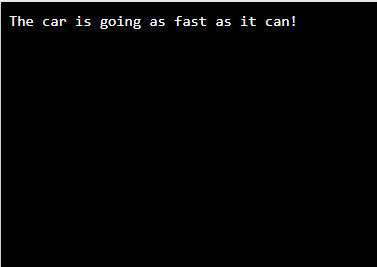
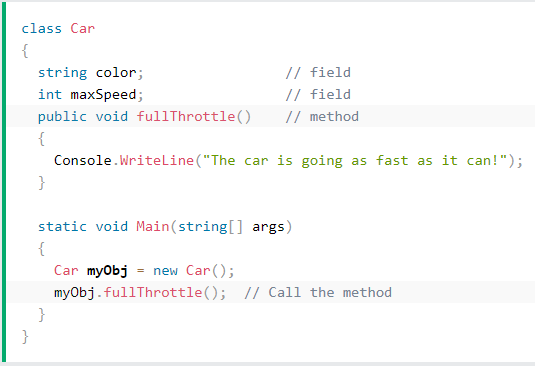

Nun wird eine Klasse mit drei Klassenmitglieder (zwei fields & eine method) erstellt.  


Hier wird ein Objekt myObj von der Klasse car erstellt und folglich die Werte von den fields color & maxSpeed ausgegeben.  


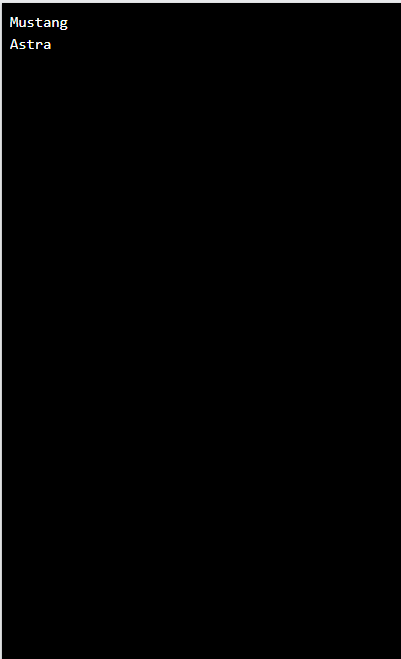
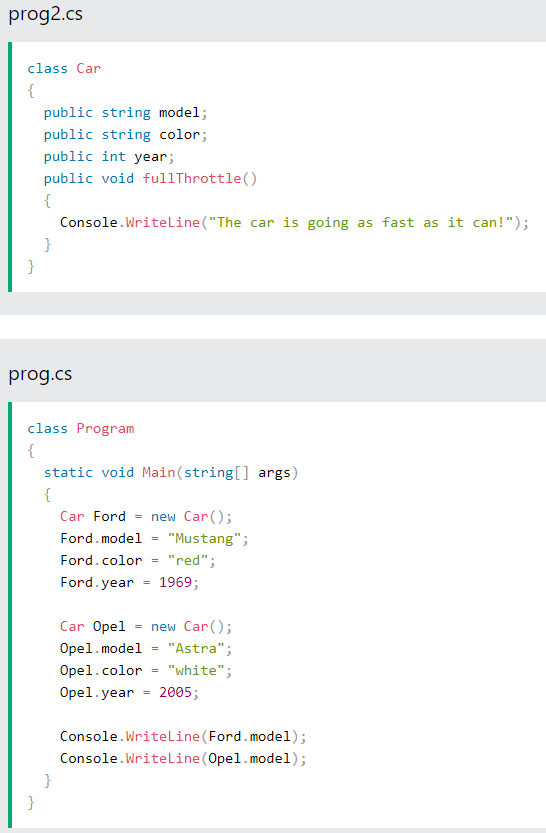
Man kann auch die fields leer lassen also nichts darin definieren und es dann beim erstellen des Objektes machen.  


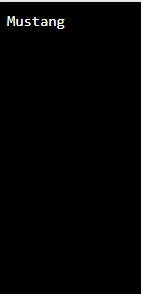
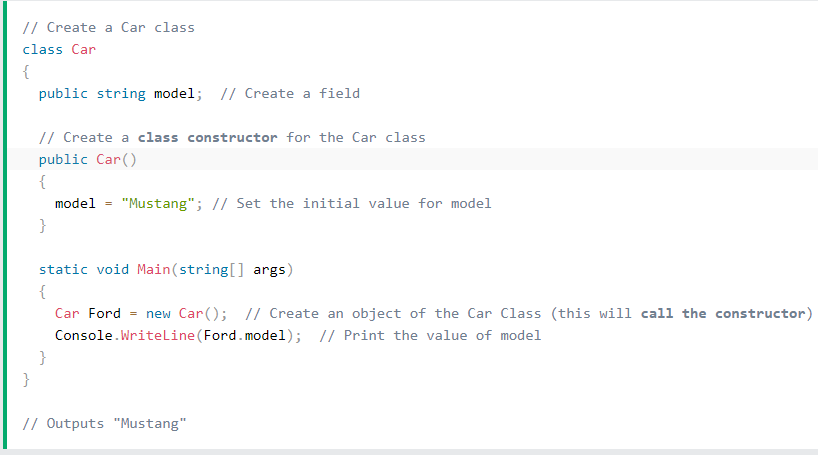
Das wird oft gebraucht, wenn man mehrere Objekte von einer Klasse erstellt.  


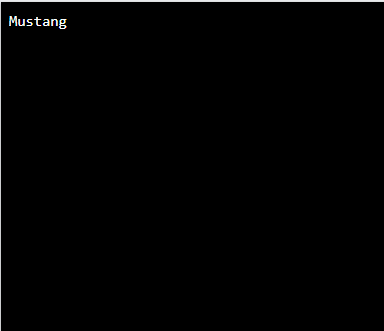
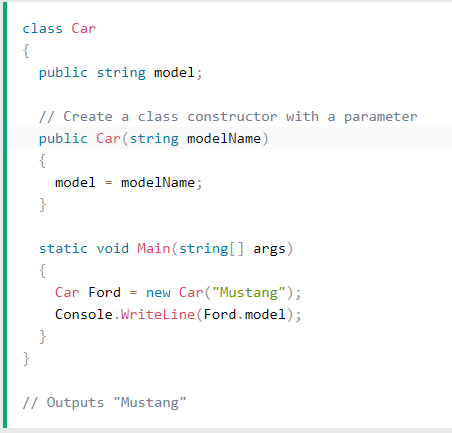
Methoden gehören normalerweise einer Klasse und sie definieren wie ein Objekt einer Klasse sich verhaltet. Wie auch fields kann man methods zugreifen. Die Methoden müssen aber public sein und zum Aufrufen muss der Name der Methode verwendet werden.

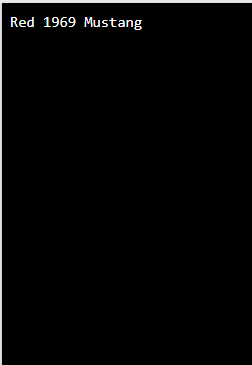
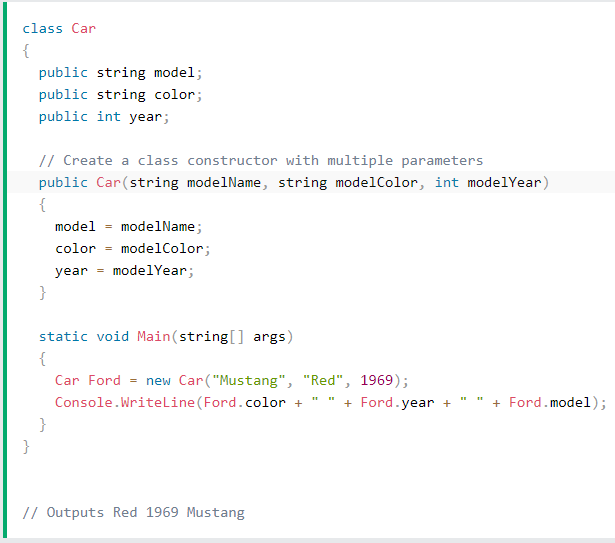


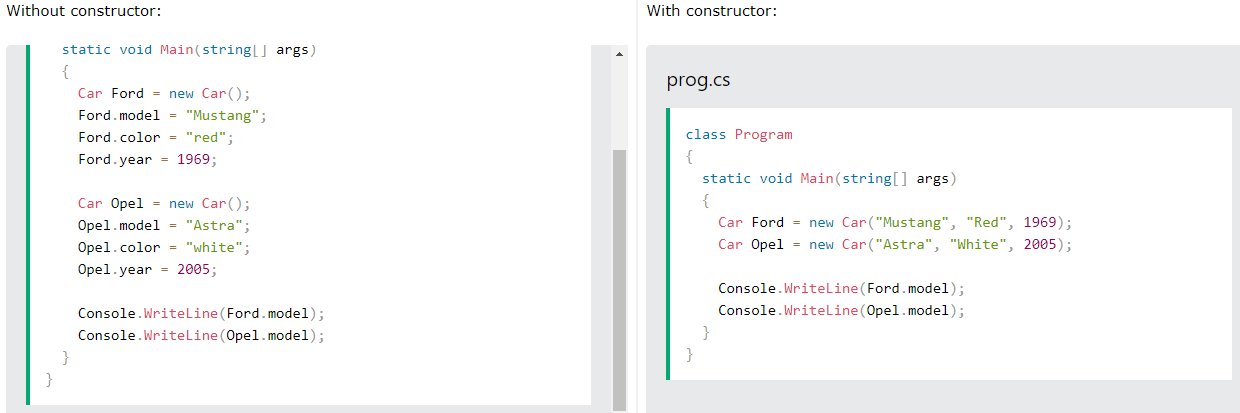
Wie schon bereits erwähnt kann man mehrere Klassen brauchen. Das macht man, damit man eine bessere Organisation hat. Eine Klasse ist für fields & methods und die andere Klasse für die ausführung (Objekt).

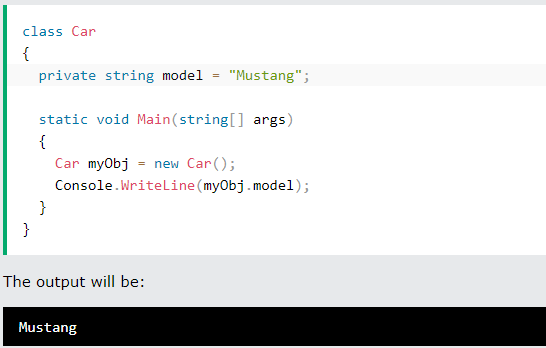


Ein Konstruktor ist eine spezielle Methode, die verwendet wird, um Objekte zu initialisieren. Der Vorteil eines Konstruktors besteht darin, dass er aufgerufen wird, wenn ein Objekt einer Klasse erstellt wird. Es kann verwendet werden, um Anfangswerte für Felder festzulegen.  
Nun wird ein constructor erstellt.  


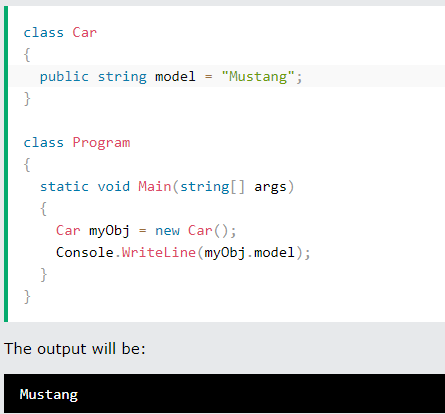
Constructor können auch Parameters haben.   


Mehrere Parameters in einem Constructor ist auch möglich  


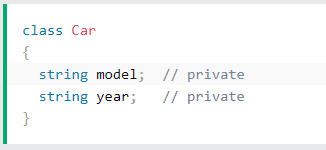
Jetzt zum vergleich wie viel Code man mit und wie viel ohne constructor hat.  


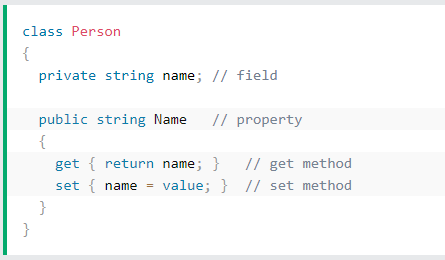
Wenn man einen field private definiert, kann es nur in der gleichen Klasse gebraucht werden.  


Wenn man in einer anderen Klasse darauf zugreifen möchte wird ein error kommen.  


Wenn man einen field public deklariert kann es in jeder Klasse gebraucht werden.  


Standartmässig sind alle members einer Klasse private, wenn man nicht davor deklariert.



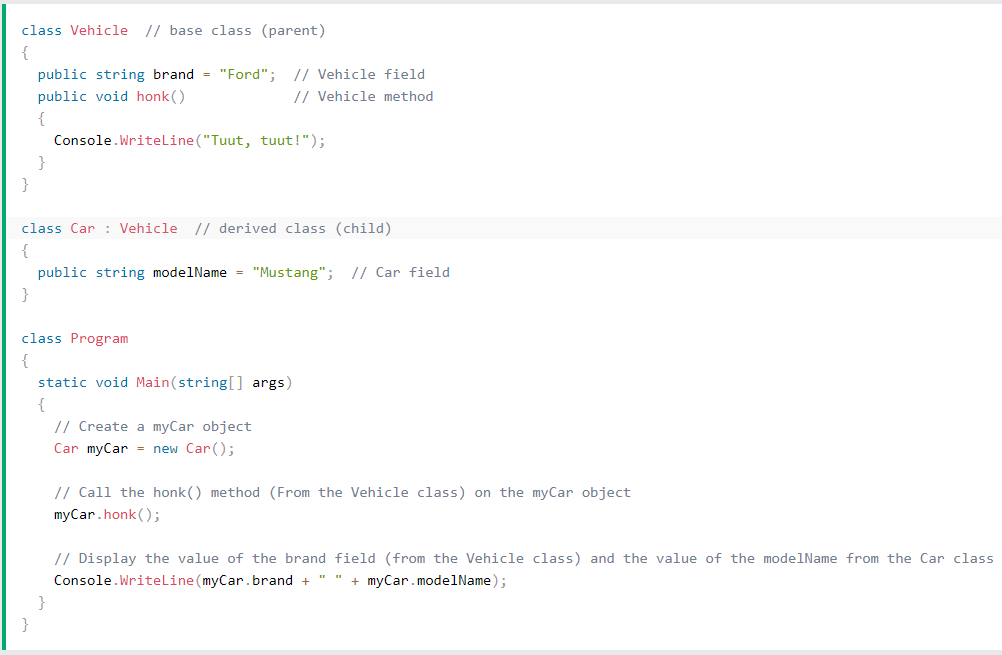
Vorher haben wir gelernt, dass private Variablen nur in der gleichen Klasse zugegriffen werden kann. Manchmal müssen wir jedoch Zugriff auf diese haben und diese werden mit properties gemacht.  
Eine property ist eine Kombination von Variablen und Methoden, und es hat zwei Methoden: ein get und ein set method.  


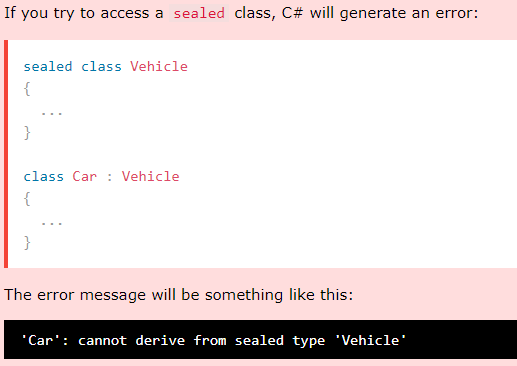
Jetzt können wir den Name property brauchen um auf den private field von der Klasse Person zugreifen und updaten zu können.

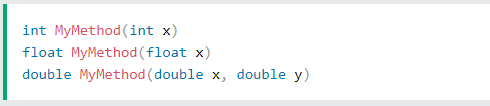


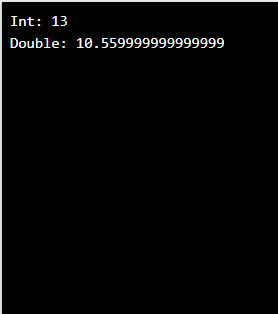
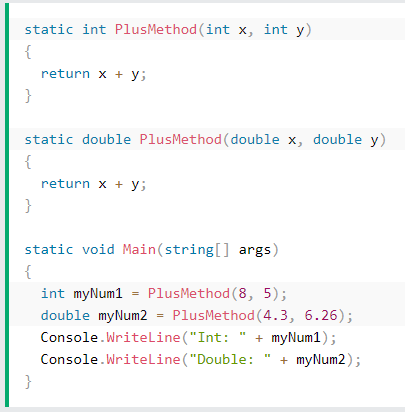
Es gibt auch short-handy / automatic properties. Das ist genau das gleiche nur müssen wir hier den field für den protery nicht definieren und können einfach get; und set; schreiben.  
Folglich das gleiche Ergebnis mit dem Unterschied, dass jetzt weniger code ist.

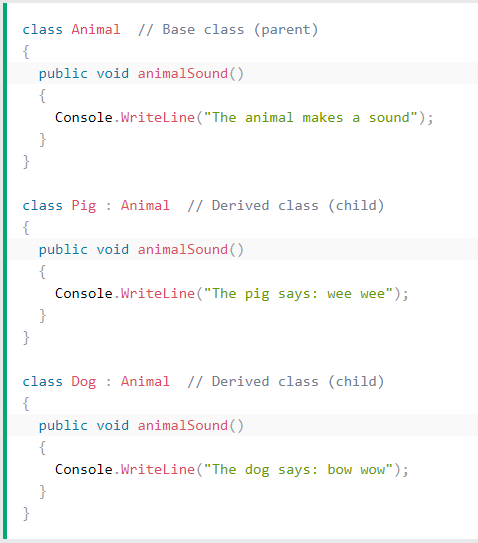


Es ist möglich fields und methods von einer Klasse zu erben. Dies ist nützlich für die Wiederverwendbarkeit von Code: Verwenden Sie Felder und Methoden einer vorhandenen Klasse wieder, wenn Sie eine neue Klasse erstellen.  
Hier erbt die Klasse Car (child) die fields und methods von der Klasse Vehicle (parent)  


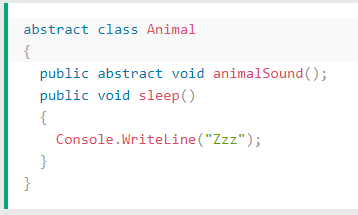
Wenn man nicht will das andere Klassen von einer Klasse erben kann.  


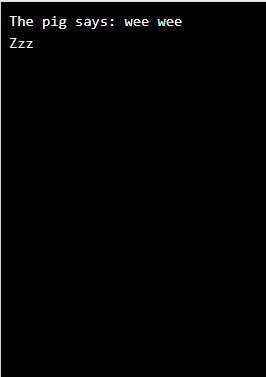
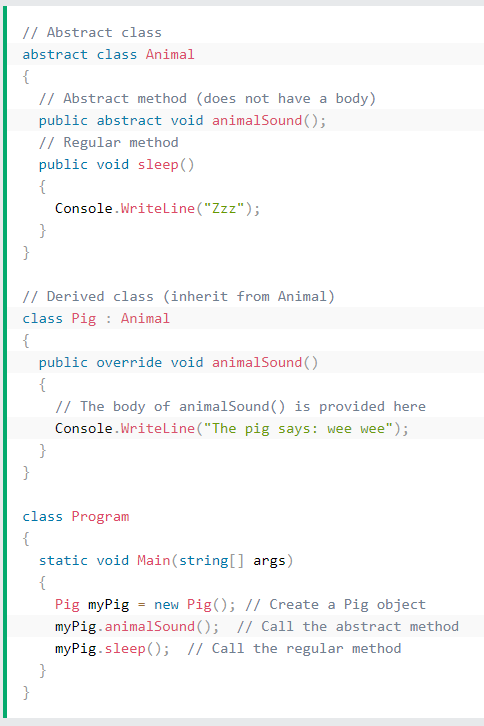
Mit der Methode overloading können mehrere Methoden denselben Namen mit verschiedenen Parametern haben.  


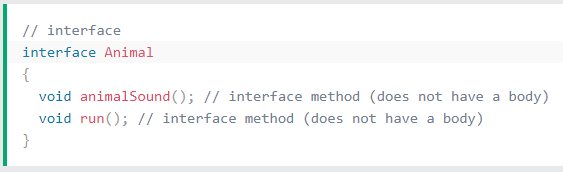
Nun wurde die Methode PlusMethod overloaded  


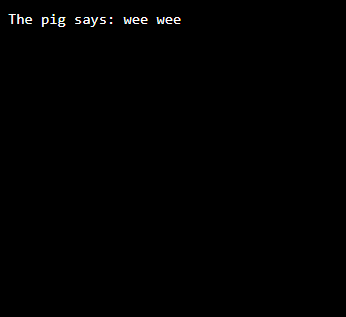
Polymorphismus bedeutet "viele Formen" und tritt auf, wenn wir viele Klassen haben, die durch Vererbung miteinander verbunden sind.  
Nun eine Basisklasse namens Animal mit einer Methode animalSound() und abgeleitete Tierklassen Pig und Dog mit ihrer eigenen Tierlaut.  


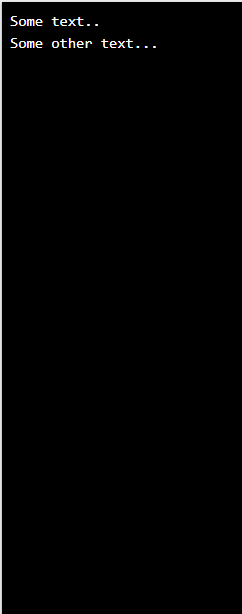
Jetzt je ein Objekt für Pig und Dog und rufen die Methode animalsound() bei beidem auf.  

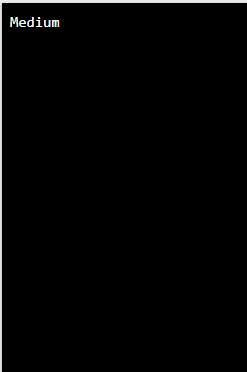
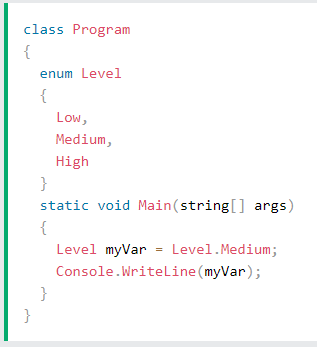

Data abstraction ist der Prozess, bei dem bestimmte Details verborgen und dem Benutzer nur bedeutsame Infromationen angezeigt werden. Abstraction kann entweder mit abstrakten Klassen oder Interfaces erreicht werden.  
Eine abstrakte Klasse kann beide Methoden haben also abstract method und regular method.  


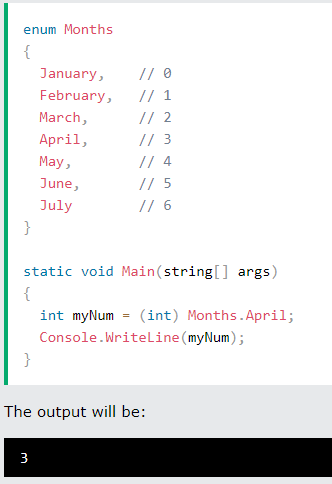
Noch ein beispiel  


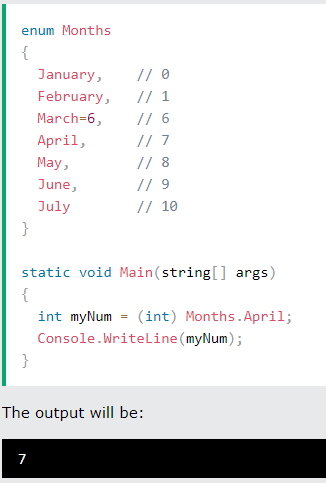
Ein interface ist eine abstrakte Klasse, welche nur abstrakte Methoden und properties mit leeren bodys enthalten darf.  


Zum auf die unterface methods zugreifen zu können muss es bei einer anderen Klasse implementiert werden. Zum implemtieren ist es das gleiche wie bei inheritance. Nur muss man diesmal kein override benutzen.  


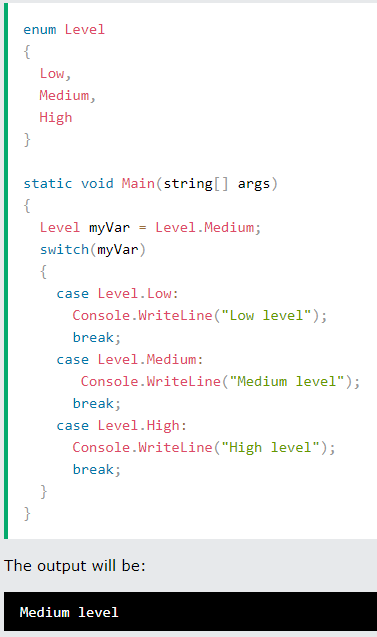
Um mehrere interfaces zu implementieren muss man diese mit einem Komma trennen  


Ein enum ist eine spezielle Klasse welche eine Gruppe von constants repräsentiert. Enums braucht man, wenn man Werte hat bei dem wir wissen die werden nicht verändert wie z.B Monate, Tage, Farbe oder Rechenoperatoren.  
Ein enum macht man mit dem keyword enum und separiert dann die enum items mit einem Komma. Enum kann auch in einer Klasse sein.  


Standartmässig fängt der erste item von einem enum mit 0 an und nummeriert reihennach weiter. Um einen Integer Wert von einem Item zu erhalten muss man es zuerst Konvertieren.  


Man kann zum Beispiel dem dritten enum den Wert 6 geben und dann wird’s ab 6 weiter nummeriert.  


Enums werden oft in einem switch statement gebraucht, um nach entsprechenden Werten zu suchen.



Die Throw-Anweisung wird zusammen mit einer Ausnahmeklasse verwendet (ArithmeticException, FileNotFoundException, IndexOutOfRangeException, TimeOutException)  
