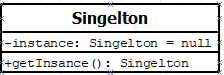
Patterns

# Singelton

Das Singelton Pattern ermöglicht es die Instanziierung einer Klasse so zu beschränken, dass es nur einmal erzeugt werden kann. Hierfür implementiert die Singelton Klasse eine Instanz von sich selbst.  
Will man diese Klasse verwenden möchte, dann ruft man diese Instanz auf.  
Bsp.: Datenbankverbindung

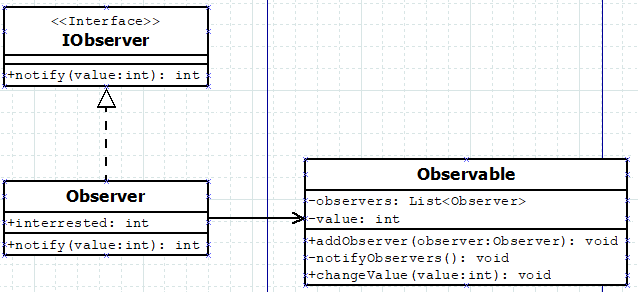
### UML Diagramm:



# Observer Pattern

Beim Observer Pattern wird ein Objekt implementiert, das beobachtet wird und Objekte, die dieses beobachten. Die Beobachter werden dann benachrichtigt, wenn sich bei dem beobachteten Objekt etwas ändert das den Beobachter interessiert.  
Bsp.: Push Nachrichten (Digitale Zeitung)

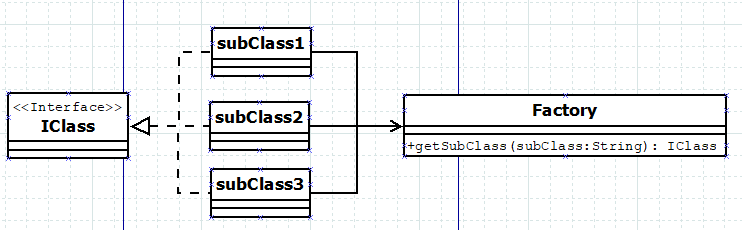
### UML Diagramm:



# Factory

Die Idee hinter dem Factory Pattern ist, dass der Nutzer nicht die Implementierung von Objekten übernimmt, also dass dies nicht in der Main Methode passiert, sondern von einem Factory Objekt übernommen wird. Als Rückgabewert definieren wir eine Abstrakte Klasse bzw. ein Interface von dem Alle Klassen, die mit dieser Methode der Factory erzeugt werden soll, erben bzw. implementieren.  
Bsp.: Pizza Factory

### UML Diagramm:



# Strategy

Das Ziel ist es Überschneidungen von Funktionen zu verhindern

Hierbei werden Algorithmen, die dasselbe Ziel haben aber dies auf verschiedenen Wegen erreichen zu einer Gruppe zusammengefasst.  
Das Ziel ist es Überschneidungen von Code, vor allem von Funktionen zu verhindern und es zu ermöglichen, dass diese von einer Zentralen Klasse aufgerufen werden.  
Besonders eignet sich dies, wenn es sich hierbei um Eigenschaften von bestimmten Klassen handelt. Diese können dann in einem Interface zusammengefasst werden.

### UML Diagramm:

