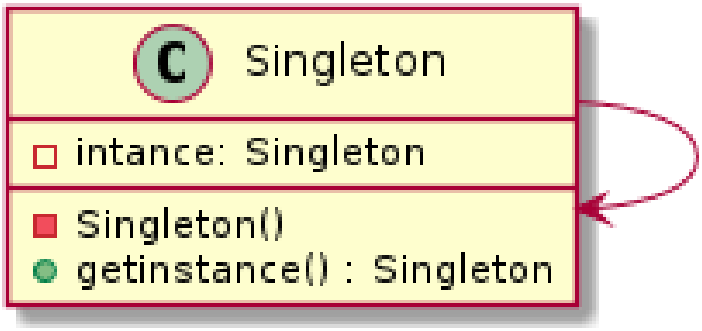
# Merkblatt Design Pattern – Singleton Pattern

## Zweck

Das Singleton Pattern ist dazu da, dass es von einer Klasse nur genau ein Objekt gibt, auf welches das gesamte Programm Zugriff hat, es hat also einen globalen Zugriff. So haben alle Klassen im System dasselbe Objekt auch wenn dieses verändert wird.

## UML




## Problem

Sie möchten ein Objekt erstellen welches für alle anderen Klassen ein gemeinsamer Nenner ist, d.h. alle haben darauf Zugriff und wenn es verändert wird passiert dies nicht nur in einer Klasse, sondern alle Klassen haben danach noch das selbe Objekt oder anders ausgedrückt dieselben Informationen welche im Objekt gespeichert sind.

## Lösung

In der Klasse, von welcher das einzigartige Objekt erstellt werden soll, muss der Konstruktor nicht auf public sondern auf private gesetzt werden. Ausserdem erstellt man in der Klasse selber ein privates statisches Objekt seiner selbst und fügt noch eine Methode getInstance() hinzu welche ebenfalls statisch jedoch aber nicht privat sondern public ist, damit alle anderen Klassen über diese Methode sich das nur einmal bestehende Objekt holen können. Mit dem privaten Konstruktor wird sichergestellt, dass es nur ein einziges Objekt dieser Klasse geben kann, da dieser nur aus der eigenen Klasse aufgerufen werden kann. Da das Objekt in der Klasse statisch ist, gibt es dieses nur einmal und es kann aufgrund der Klassifizierung privat nur von der eigenen Klasse verwendet werden daher wird die getInstance() Methode benötigt. Lazy Creation kann verwendet werden, um Speicherbedarf zu sparen, dazu erstellt man das Objekt erst, wenn die getInstance() Methode aufgerufen wird.

## Erweiterbarkeit

Sollte man das Objekt aus anderen Klassen anpassen können müssen kann dies sehr einfach realisiert werden. Mann erstellt Variablen welche auf privat sind und erstellt dazu die passenden Getter & Setter, welche ihrerseits wiederum public sind.

# Beispiel in Java

```
public class Singleton {

	//Privater Konstruktor
	private Singleton() {
	}

	//Privates und statisches Objekt
	private static Singleton singleton;

	//Methode um das Objekt zu erhalten
	public static Singleton getInstance() {
		// Lazy Creation; so wird das Objekt erst bei Bedarf erstellt
		if (Singleton.singleton == null) {
			singleton = new Singleton();
		}
		return singleton;
	}
}
```

```
public class Main {

	public static void main(String[] args) {

		Singleton singleton = Singleton.getInstance();

		System.out.println(singleton);

	}

}
```