

# Singleton-Pattern

---

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

# Motivation

```
public enum Fach { IFM, ELM, ARC }
```

```
Logger l = Logger.getLogger(MyClass.class.getName());
```

# Umsetzung: “Eager” Singleton Pattern

```
public class SingletonEager {  
    private static final SingletonEager inst = new SingletonEager();  
  
    // Privater Constructor: Niemand kann Objekte außerhalb der Klasse anlegen  
    private SingletonEager() {}  
  
    public static SingletonEager getInst() {  
        return inst;  
    }  
}
```

# Umsetzung: “Lazy” Singleton Pattern

```
public class SingletonLazy {  
    private static SingletonLazy inst = null;  
  
    // Privater Constructor: Niemand kann Objekte außerhalb der Klasse anlegen  
    private SingletonLazy() {}  
  
    public static SingletonLazy getInst() {  
        // Thread-safe. Kann weggelassen werden bei Single-Threaded-Gebrauch  
        synchronized (SingletonLazy.class) {  
            if (inst == null) {  
                inst = new SingletonLazy();  
            }  
        }  
        return inst;  
    }  
}
```

Sie schaffen damit eine globale Variable!

Singleton-Pattern: Klasse, von der nur genau ein Objekt instantiiert werden kann

1. Konstruktor “verstecken” (Sichtbarkeit auf `private` setzen)
2. Methode zum Zugriff auf die eine Instanz
3. Anlegen der Instanz beispielsweise beim Laden der Klasse (“Eager”) oder beim Aufruf der Zugriffsmethode (“Lazy”)



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.