**Singleton Design Pattern**

**1.Einleitung**

* Das Singleton-Pattern wird oft in Softwareprojekten verwendet, um sicherzustellen, dass es nur eine Instanz einer Klasse gibt.
* Das Ziel des Singleton-Patterns ist es, eine kontrollierte, globale Instanz zu haben.
* Ein Singleton wird verwendet, wenn eine Klasse nur eine einzige Instanz haben soll, z. B. bei einer Datenbankverbindung oder beim Logging.

**2.Implementierung:**

* Ein privates, statisches Feld für die Singleton-Instanz in der Klasse anlegen und eine statische Methode zum Erstellen und Abrufen dieser Instanz definieren.
* "Lazy Initialization" verwenden, um beim ersten Aufruf die Instanz zu erstellen und anschließend immer die gleiche Instanz zurückzugeben.
* Konstruktor privat machen, damit nur die statische Methode ihn aufrufen kann.

**3.Vorteile und Nachteile des Singleton-Patterns**



**4.Multithreading und Singleton**

• In einer Multithreaded Umgebung kann es passieren, dass mehrere Threads gleichzeitig auf 'getInstance()' zugreifen wollen.

• Mit dem 'synchronized' Schlüsselwort stellen wir sicher, dass immer nur ein Thread gleichzeitig die Methode aufrufen kann, und so bleibt die Klasse thread-sicher.

**5.Double-Checked Locking:** Dieses Muster stellt sicher, dass die Instanz nur dann gesperrt und erstellt wird, wenn sie wirklich noch nicht existiert, um so unnötige Wartezeiten zu vermeiden.

**6.Load Balancing (Lastverteilung):**  
Wenn viele Nutzer auf die Datenbank zugreifen, sorgt der Load Balancer dafür, dass die Anfragen gleichmäßig auf verschiedene Server verteilt werden. Dadurch wird die Last effizienter verteilt, was die Leistung verbessert und Überlastungen verhindert.