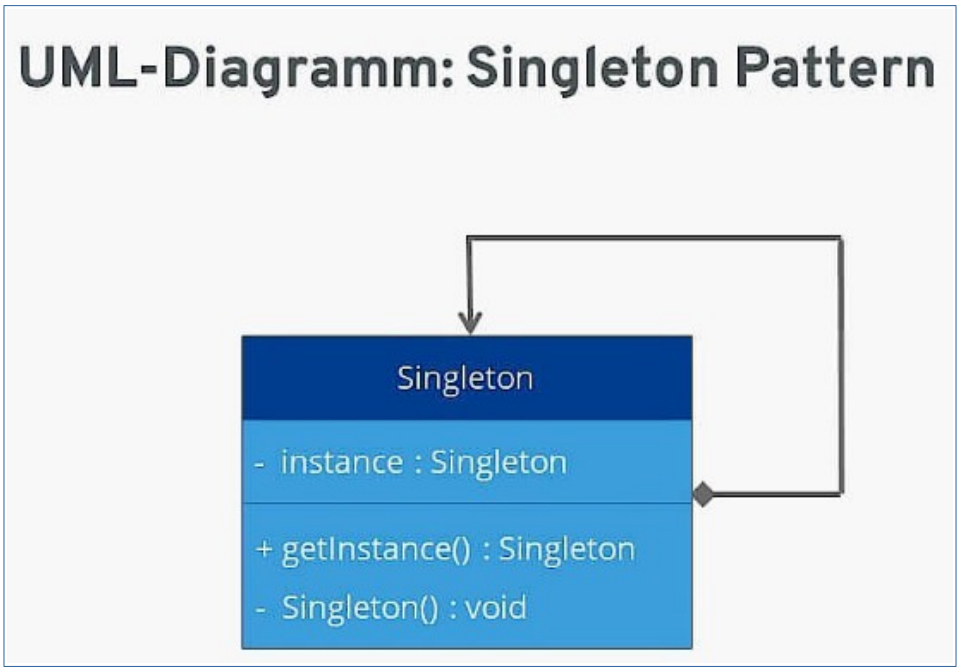
Seine Aufgabe besteht darin, **zu verhindern, dass von einer Klasse mehr als ein Objekt erstellt werden kann**. Das wird dadurch erreicht, dass das gewünschte **Objekt in einer Klasse selbst erzeugt** dann **als statische Instanz abgerufen wird.** Das Singleton zählt zu den einfachsten, aber dafür mächtigsten Patterns in der Software-Entwicklung.



Wie implementiert man Singalton richtig?

Lazy Loading: Die Singleton-Instanz wird nur dann erstellt, wenn `getObjekt()` zum ersten Mal aufgerufen wird, also wenn sie tatsächlich benötigt wird.

Synchronized (Lazy Loading):
Lazy Loading bedeutet, dass ein Objekt oder eine Ressource erst dann erstellt oder geladen wird, wenn es wirklich gebraucht wird.

„Also macht Thread-Sicherheit“

```java
public class Singleton {

    // privater Konstruktor
    private Singleton() {}

    // statische Variable für das Singleton-Objekt
    private static Singleton erzeuge_Objekt = null;

    // synchronisierte Methode für den Zugriff auf das Singleton
    public static synchronized Singleton getObjekt() {
        if (erzeuge_Objekt == null) {
            erzeuge_Objekt = new Singleton();
        }
        return erzeuge_Objekt;
    }

    public void print() {
        System.out.println("Hallo Welt!");
    }

    public static void main(String[] args) {
        Singleton instance = Singleton.getObjekt();
        instance.print();
    }
}
```

Beispiel:

```
public class Drucker {
    private static Drucker drucker;
    private int AnzahlSeiten;
    private Drucker() {
    }
    public static Drucker getInstance() {
        return drucker == Null ?
                drucker = new Drucker() :
                drucker;
    }
    public void print(String text){
        System.out.println(text +
                "\n" + "Anzahl der heute gedruckten Seiten" + ++ AnzahlSeiten +
                "\n" + "---------");
    }
}
```

```
public class Employee {
    private final String name;
    private final String position;
    private final String taetigkeit;
    public Mitarbeiter(String name, String position, String taetigkeit) {
        this.name = name;
        this.position = position;
        this.taetigkeit = taetigkeit;
    }
    public void printCurrent taetigkeit (){
        Drucker drucker = Drucker.getInstance();
        drucker.print("Mitarbeiter: " + name + "\n" +
            "Position: " + position + "\n" +
            "Taetigkeit: " + taetigkeit + "\n");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Mitarbeiter andreas = new Mitarbeiter ("Andreas",
                "Chef",
                "Leitet die Niederlassung");
        Mitarbeiter julia = new Mitarbeiter ("Julia",
                "Beraterin",
                "Berät Kunden bei Reklamationen");
        Mitarbeiter tom = new Mitarbeiter ("Tom",
                "Verkäufer",
                "Verkauft die Produkte");
        Mitarbeiter stefanie = new Mitarbeiter ("Stefanie",
                "Entwicklerin",
                "Wartung der IT in der Niederlassung.");
        Mitarbeiter matthias = new Mitarbeiter ("Matthias",
                "Buchhalter",
                "Finanzbuchhaltung der Niederlassung.");
        andreas.printCurrentTaetigkeit();
        julia.printCurrentTaetigkeit ();
        tom.printCurrentTaetigkeit ();
        stefanie.printCurrentTaetigkeit ();
        matthias.printCurrentTaetigkeit ();
    }
}
```