SMART INDUSTRY CAMPUS

Singleton - Pattern

SMART INDUSTRY CAMPUS

Was ist ein Singleton?

- Ist ein simples Entwurfsmuster einer Klasse
- Bsp: Supermarkt
  - Wir haben ein einziges Lager, in welchem wir unsere Lebensmittel lagern und auch herausnehmen
  - Mehrere Lager könnten zu Inkonsistenzen und Verwirrungen führen

SMART INDUSTRY CAMPUS

Eigenschaften

- es darf nur ein einziges Objekt dieser Klasse existieren
  - Ermöglicht durch einen privaten Konstruktor
  - Der private Konstruktor wird innerhalb des Singletons aufgerufen
- die Instanz wird mittels der statischen getInstance()-Methode zurückgegeben
- die Klasse kann weitere Methoden enthalten

Beispiel Lager

```
class Warehouse {
    private static Warehouse instance;

    private Warehouse() {
    }
    public static Warehouse getInstance() {
        if (instance == null) {
            instance = new Warehouse();
        }
        return instance;
    }
}
```

Private Variable
vom Typ Warehouse

Private Konstruktor:
Von außerhalb
nicht zugreifbar

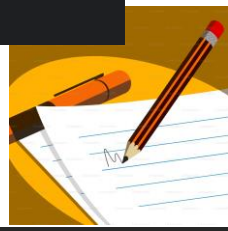
Methode, welche die
Instanz zurückgibt

Quiz

Wie kriege ich in einer
anderen Klasse die Instanz des Lagers?

```
class Warehouse {
    private static Warehouse instance;

    private Warehouse() {
    }
    public static Warehouse getInstance() {
        if (instance == null) {
            instance = new Warehouse();
        }
        return instance;
    }
}
```



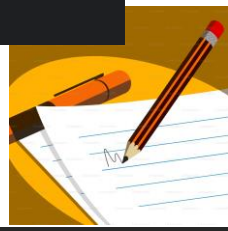
Quiz

Wie kriege ich in einer
anderen Klasse die Instanz des Lagers?

Warehouse.getInstance();

```
class Warehouse {
    private static Warehouse instance;

    private Warehouse() {
    }
    public static Warehouse getInstance() {
        if (instance == null) {
            instance = new Warehouse();
        }
        return instance;
    }
}
```



Beispiel Lager

Man kann die Methoden addBestand und removeBestand synchronized machen.

Bei mehreren Threads kann das nämlich zur Konsistenz des Zustandes verhelfen.

```
class Warehouse {
    private static Warehouse instance;
    private int noodlePackages;

    private Warehouse() {
        noodlePackages = 100;
    }

    public static Warehouse getInstance() {
        if (instance == null) {
            instance = new Warehouse();
        }
        return instance;
    }

    public void addBestand(int menge) {
        noodlePackages += menge;
        System.out.println("Bestand erhöht. Neuer Bestand: " + noodlePackages);
    }

    public void removeBestand(int menge) {
        if (menge <= noodlePackages) {
            noodlePackages -= menge;
            System.out.println("Bestand verringert. Neuer Bestand: " + noodlePackages);
        } else {
            System.out.println("Nicht genügend Bestand verfügbar!");
        }
    }

    public int getBestand() {
        return noodlePackages;
    }
}
```

SMART INDUSTRY CAMPUS

Aufgabe

Erstelle eine Klasse Mitarbeiter mit folgenden Eigenschaften:

- er hat einen Namen
- er kann eine bestimmte Menge an Waren hinzufügen
- er kann eine bestimmte Menge an Waren entnehmen

Die Methoden sollen auf das Lager zugreifen und den Bestand verändern!

Erstelle in der Main-Methode Objekte und überprüfe die Funktionalität.

SMART INDUSTRY CAMPUS

Aufgabe

Es existiert nur ein einziger Drucker.
Erstelle eine Klasse Printer:

- enthält Variable von sich selbst
- enthält eine Variable mit numberOfPages

- hat eine Methode, die die Instanz zurückgibt
- hat eine Methode, welche die Anzahl der gedruckten Seiten bisher zurückgibt

Erstelle eine Klasse Employee. Der Employee soll mit dem Drucker interagieren können.