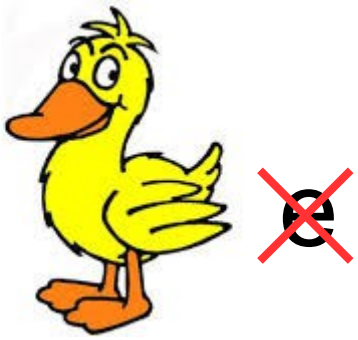
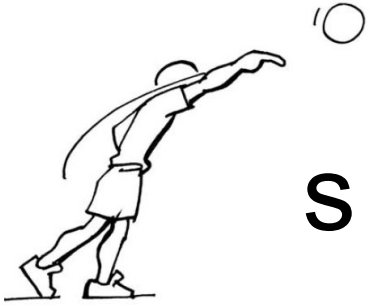
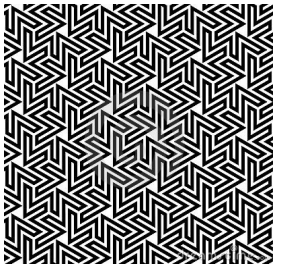


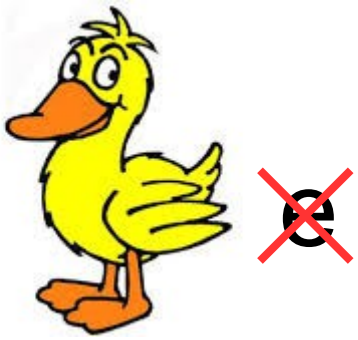
e



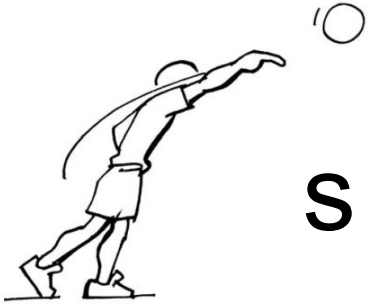
S



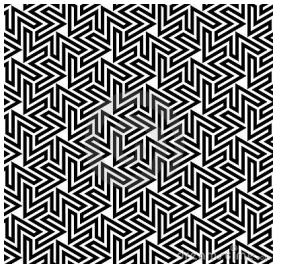
Entwurfsmuster



e



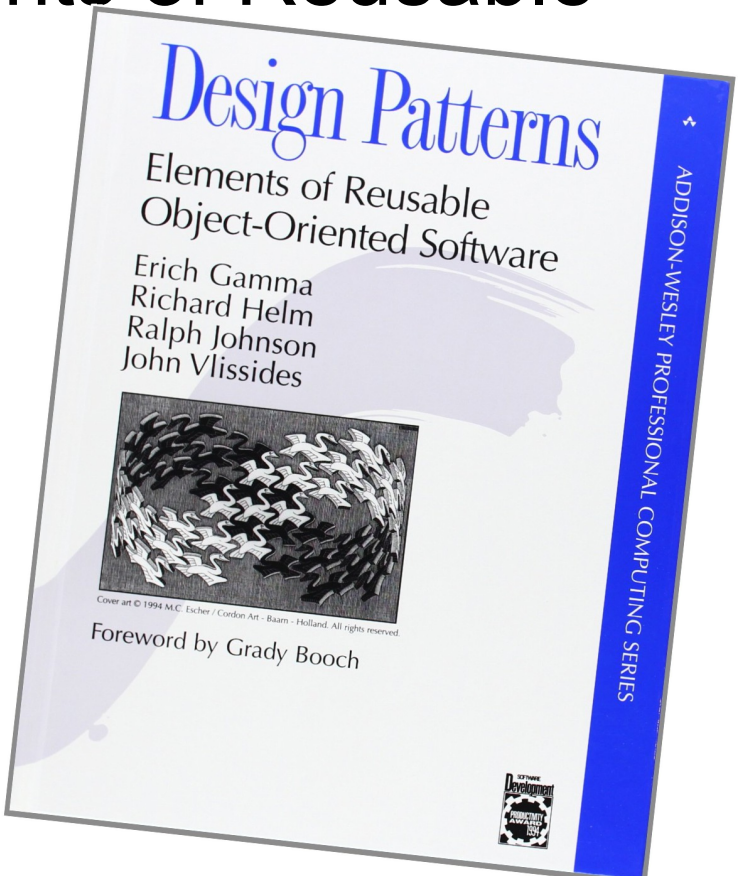
s



Aus der Geschichte...

- 1979 Christopher Alexander → Vitruv
- 1988 Erich Gamma
- 1994 „Design Patterns – Elements of Reusable Object-Oriented Software“
  - Erich Gamma
  - Richard Heim
  - Ralph Johnson
  - John Vlissides

„Gang of Four“ (GoF)



Entwurfsmuster...

- … lösen wiederkehrende Entwurfsprobleme in der Softwareentwicklung.
- … führen das aktuelle Problem auf ein schon gelöstes zurück.
- … helfen bei der Strukturierung des Projekts.

Arten

- Erzeugungsmuster

  Erzeugung von Objekten, z. B. Singleton

- Strukturmuster

  Zusammensetzung von Klassen/Objekten z. B. Kompositum

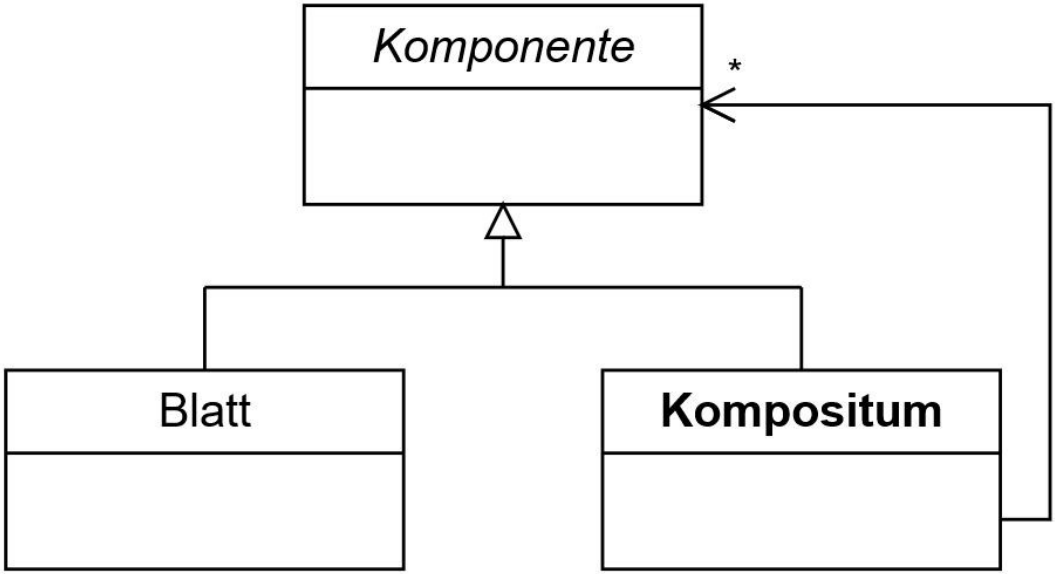
- Verhaltensmuster

  Zusammenspiel zwischen Klassen/Objekten z. B. Beobachter

- Weitere Muster

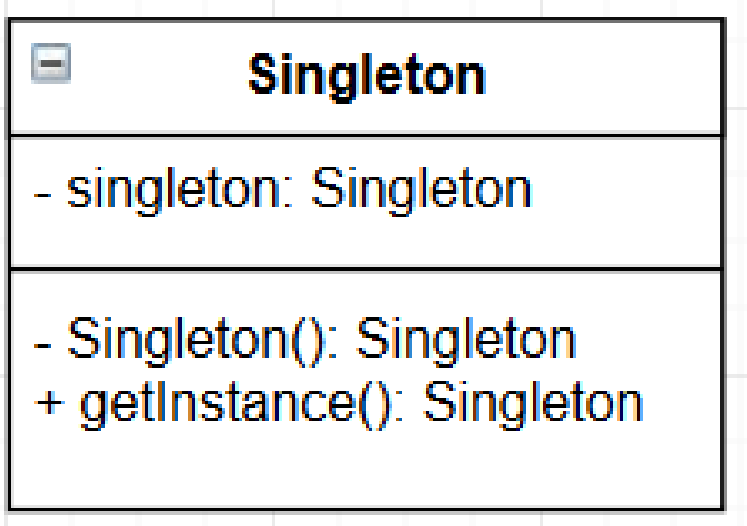
Bereits bekannt: Das Kompositum

- implementiert die „Teil-Ganzes-Hierarchie“
- Objekt einer Klasse kann Knoten oder Blatt sein
- Beispiel: Ordnerstrukturen



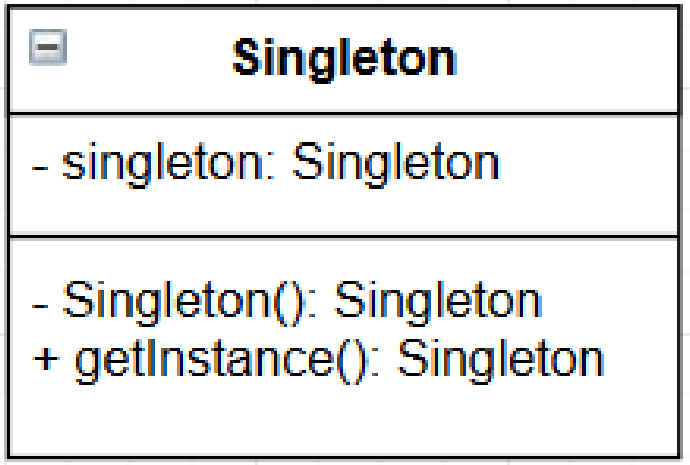
Singleton: ein Erzeugungsmuster

- erzeugt und verwaltet einziges Objekt einer Klasse
- bietet globalen Zugriff auf das Objekt über eine Instanzoperation
- die Instanzoperation ist eine Klassenmethode, d.h. statisch gebunden

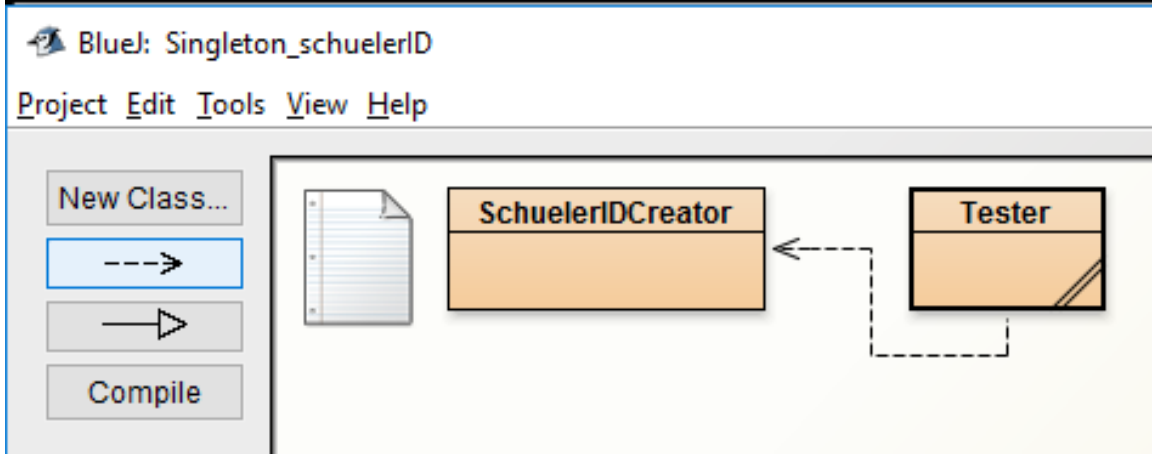


Singleton - Ein Rezept

- privater Konstruktor
- private Variable vom Typ der Klasse
- öffentliche Methode getInstance() gibt die Variable zurück
- Variable und die Methode sind statisch



Arbeitsauftrag



Situation:

- SchuelerIDCreator-Objekt, welches für neu registrierte Schüler eine fortlaufende ID erzeugt

Dein Auftrag:

- Besuche *unterrichtbei.herrkraus.de*
- Lade/Entpacke Singleton_schuelerID.zip
- Vervollständige die Klasse schuelerIDCreator
- Teste deine Klasse

Lösungsvorschlag

```java
/**Das (einzige) Objekt dieser Klasse sorgt für die fortlaufende Rückgabe von SchülerIDs.*/
public class SchuelerIDCreator
{
    /*
     REZEPT
       ~ privater Konstruktor
       ~ private Variable "schuelerIdCreator" vom Typ der Klasse
       ~ öffentliche Methode getInstance() gibt die Variable zurück
       ~ Variable und die Methode sind statisch
    */

    private static SchuelerIDCreator schuelerIdCreator = new SchuelerIDCreator();
    private int schuelerIDCounter;   //letzte erzeugte SchülerID

    /**Erzeugt die einzige Instanz dieser Klasse und initialisiert den schuelerIDCounter mit 0*/
    private SchuelerIDCreator(){
        schuelerIDCounter = 0;
    }

    /**Gibt das einzige Objekt dieser Klasse zurück*/
    public static SchuelerIDCreator getInstance(){
        return schuelerIdCreator;
    }

    /**Gibt die SchülerID für einen neu registrierten Schüler zurück*/
    public int getNewSchuelerID(){
        schuelerIDCounter++;      //schuelerID weiterzählen
        return schuelerIDCounter;
    }
}
```

Weiter gedacht...

Eager Loading:

```
public class Singleton {

    private static final Singleton instance = new Singleton();

    private Singleton() {
    }

    public static Singleton getInstance(){
        return instance;
    }

}
```

Lazy Loading:

```
public class Singleton {

    private static Singleton instance;

    private Singleton(){
    }

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

Quellen

Inhalte:

- http://www.philipphauer.de/study/se/design-pattern/singleton.php

Bilder:

- http://www.checkeins.de/sendungen/die-sendung-mit-der-maus/moderatoren/ente-100.html
- http://www.mobilesport.ch/wp-content/uploads/2013/07/L_D5_9.SJ_ABC_T3.png
- http://thumbs.dreamstime.com/x/arabeske-optisches-muster-15643529.jpg
- http://www.c-jump.com/CIS75/Week11/const_images/SingletonClass.png