Sommersemester 2021

Wolfgang Berger

Software Paradigmen

Creational Patterns

Erzeugungsmuster

Erzeugungsmuster

- Singleton
- Prototype
- Abstract Factory
- Factory
- Builder

Singleton

- Zweck
  - Objektbasiertes Erzeugungsmuster
  - Klasse, von der es genau ein Exemplar gibt
  - Zentraler Zugriffspunkt auf das Exemplar

Singleton

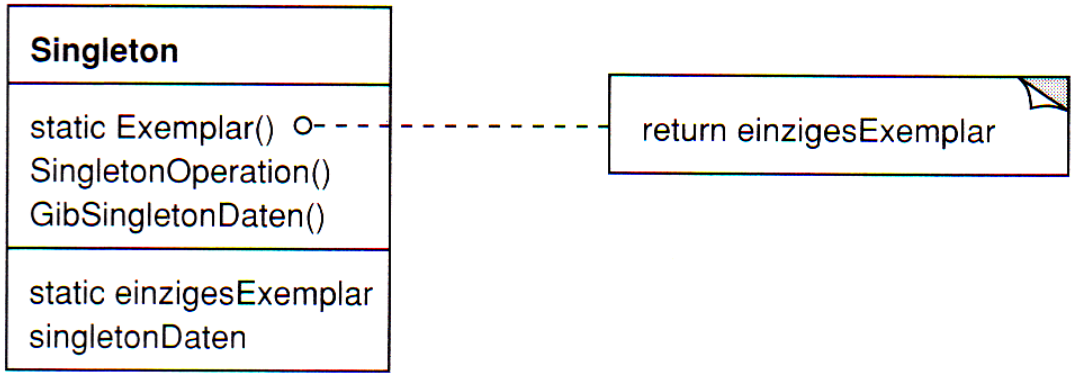
- Motivation
  - Entwicklungsmuster, das sehr oft zum Einsatz kommt
  - Immer wenn ein globales Exemplar benötigt wird, das nur einmal vorkommen darf
  - Beispiel: 1 Datenbankverbindung in einer Anwendung
  - Globale Variable stellt zwar eine zentralen Zugriff sicher, verhindert aber nicht, dass mehrere Exemplare von der Klasse erzeugt werden.
  - Es muss verhindert werden, dass der Konstruktor der Singleton-Klasse öfter als 1 mal aufgerufen wird.

Singleton

- Anwendbarkeit
  - Verwendet das Singleton Design Pattern wenn…
    - … es genau ein Exemplar einer Klasse geben muss
    - … es für den Klienten einen wohldefinierten Zugriffspunkt geben muss
    - … das Exemplar durch Bildung von Unterklassen erweiterbar sein soll
    - … Klienten das erweiterte Exemplar ohne Modifikation seines Codes verwenden können

Singleton

- Struktur



Singleton

- Teilnehmer
  - definiert eine Exemplaroperation, die es Klienten ermöglicht auf sein einziges Exemplar zuzugreifen
  - ist selbst für die Erzeugung es einzigen Exemplars verantwortlich

Singleton

- Interaktionen
  - Klienten greifen auf das Singletonexemplar ausschließlich durch die Exemplar-Operation der Singletonklasse zu

Singleton

- Konsequenzen
  - Singletonmuster besitzt mehrere Vorteile:
    - Zugriffskontrolle auf das Exemplar, weil die Singletonklasse das einzige Exemplar kapselt
    - Eingeschränkter Namensraum
    - Verfeinerung von Operationen und Repräsentation weil die Singletonklasse abgeleitet und spezialisiert werden kann
    - Variable Anzahl an Exemplaren
    - Flexibler als Klassenoperationen – Klassenoperationen sind statisch und deren Unterklassen nicht polymorph

Singleton

- Implementierung
  - Garantie eines einzigen Exemplars
    - protected Konstruktor (nur innerhalb der Klasse kann ein Exemplar erzeugt werden)
    - statische Referenz
    - Erzeugung:
      - statisch - beim Laden der Klasse – NICHT anwendbar wenn das Singleton einen dynamischen Parameter zur Initalisierung benötigt.
      - dynamisch – beim ersten Aufruf der getInstance() Methode (Lazy Initialization)
      - Vorteil der dynamischen Methode ist, dass wenn ein Singleton über die gesamte Anwendungslaufzeit nicht benötigt wird gibt es auch kein Exemplar davon. Java lädt allerdings Klassen erst wenn sie das erste Mal benötigt werden.

Singleton

- Implementierung
  - Globaler Zugriff
    - statische Zugriffsmethode getInstance(), die das einzige Exemplar liefert
  - Ableiten der Singletonklasse
    - Flexibler Ansatz der Registratur einer Singletonklasse und einer speziellen getInstance(String name) Methode

Singleton

- Übung 1!