CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

SOFTWARE DESIGN FOLIENSATZ 6

Iterator, Facade, Template

Christian Hofer

ZAM - WS 2025/26

TERMINE

0004VD2005 SOFTWARE DESIGN (29,75UE IL, WS 2025/26)

Gruppe

	Tag	Datum	von	bis	Ort	Ereignis	Termintyp	Lerneinheit	Vortragende*r	Anmerkung
	Standardgruppe									
	Do	22.01.2026	08:15	11:45	CZ106	Abhaltung	fix	**1 Organisation + Strategy Pattern**		
	Do	29.01.2026	08:15	11:45	CZ106	Abhaltung	fix	**2 Decorator Pattern + Singleton**		
	Do	05.02.2026	12:30	16:00	CZ106	Abhaltung	fix	**3 Observer**		
	Fr	06.02.2026	08:15	12:15	CZ106	Abhaltung	fix	**4 Factory**		
	Do	12.02.2026	08:15	13:00	CZ106	Abhaltung	fix	**5 Command + Adapter**		
Heute	Fr	13.02.2026	08:15	12:15	CZ106	Abhaltung	fix	**6 Facade, Template, Iterator** mit Christian Hofer!		
	Do	19.02.2026	08:15	11:45	CZ106	Abhaltung	fix	**Wiederholung**		
	Fr	20.02.2026	08:15	12:15	CZ106	Prüfungstermin	fix	**Prüfung**		

CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

ITERATOR

ITERATOR

- Iterator Pattern erlaubt es auf die Elemente einer Datenstruktur sequentiell zuzugreifen, ohne den Aufbau der Datenstruktur zu kennen.

ITERATOR IN JAVA

Iterator Klasse muss Iterator Interface umsetzen und intern die Position verwalten:

- hasNext()
- next()
- optional remove()

ITERATOR IN JAVA

- Iterable Interface ermöglich es jede Klasse (z.B. Collecion) in for each Schleifen zu verwenden.
  - iterator() muss einen passenden Iterator zurückliefern

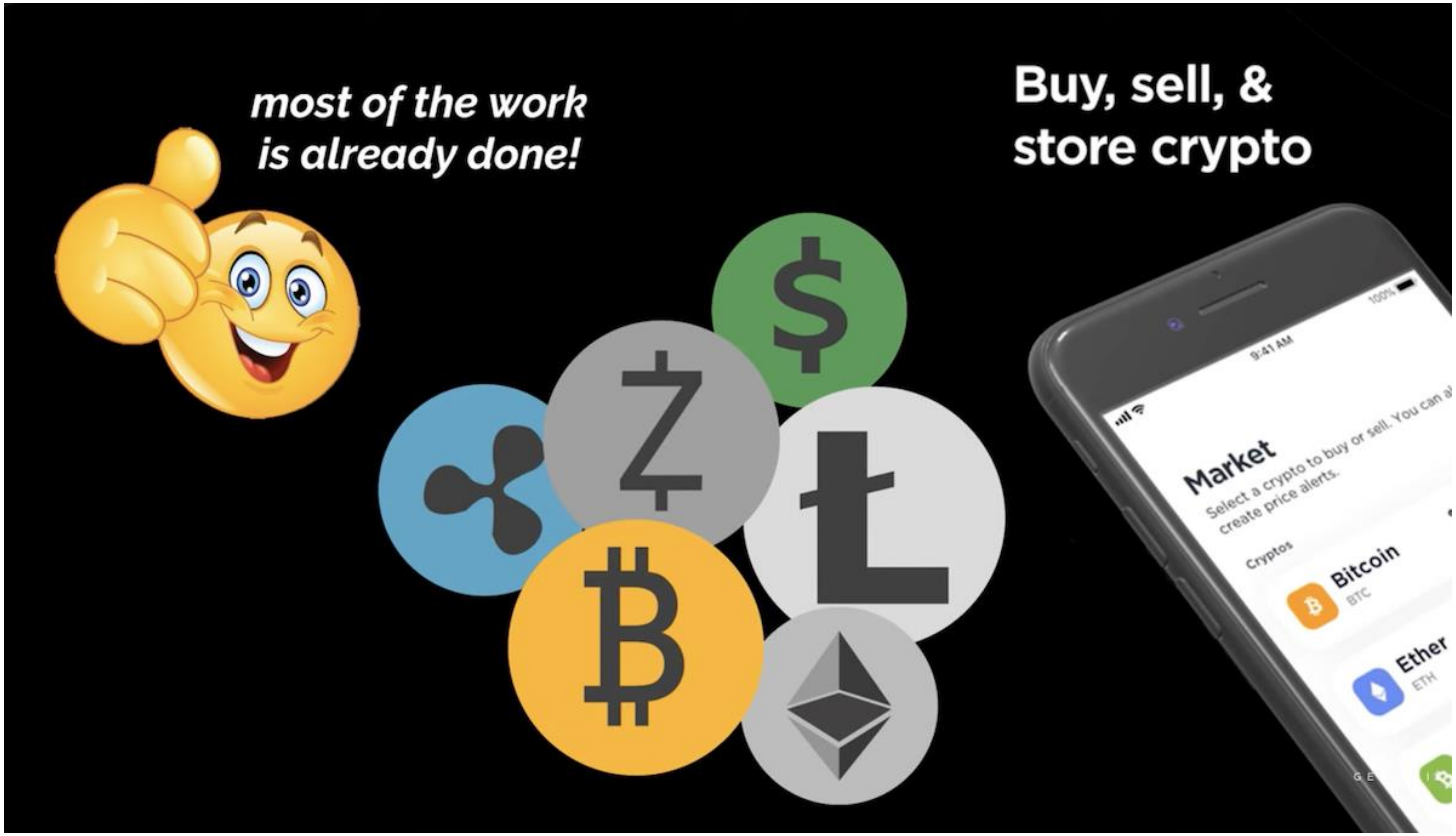
ITERATOR

- Gemeinsames Beispiel

CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

FACADE

FACADE PATTERN MIT BEISPIEL



https://www.youtube.com/watch?v=xWk6jvqyhAQ (6min, Englisch)

FACADE - FILMABEND WAS IST NÖTIG?

- 1. Popcornmaschine einschalten
- 2. Popcorn hineingeben
- 3. Licht dimmen
- 4. Leinwand herunter rollen
- 5. Projektor einschalten
- 6. Projektoreingang wählen
- 7. Verstärker einschalten
- 8. Häschen Polster aufschütteln
- 9. …

```
PopcornPopper popper = new
PopcornPopper();
popper.on();
popper.pop();
Light l1 = new Light();
l1.dim();
Light l2 = new Light();
l2.dim();
Screen screen = new Screen();
screen.down();
Projector pr = new Projector();
pr.on();
pr.switchInput();
Amplifier amp = new Amplifier();
amp.on();
BunnyCushion cushion = new
BunnyCushion();
cushion.fluff();
```

FACADE - EINFACH-ERES INTERFACE

```
// Wrappt alle notwendigen Funktionsaufrufen hinter
// dieser simpleren Fassade
public class HomeTheaterFacade {
    // alle zuvor verwendeten Klassen
    PopcornPopper popper;
    Screen screen;
    // ...
    // Rufen hier alle notwendigen Aktionen auf
    public void startMovie(String movie){
        popper.on();
        popper.pop();
        screen.on();
        // ...
    }
    public void endMovie() {
        // ...
    }
}
```

FACADE PATTERN

Bietet ein einheitliches Interface für eine Menge an Interfaces in einem Subsystem. Facade definiert ein hochleveliges Interface um die Subsysteme einfacher benutzen zu können.

CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

TEMPLATE

KAFFEE UND TEE REZEPT SEHR ÄHNLICH

- Kaffee:
  1. Wasser kochen
  2. Kaffee in Wasser brühen
  3. Kaffee in Häferl gießen
  4. Zucker und Milch hinzugeben
- Tee:
  1. Wasser kochen
  2. Tee in Wasser ziehen lassen
  3. Tee in Häferl gießen
  4. Zitronensaft hinzugeben

DUPLIZIERTEN CODE VERMEIDEN

```
public abstract class CaffeineBeverage {
    // Wir möchten nicht,
    // dass es von Subklassen überschrieben wird
    final void prepareRecipe() {
        boilWater();
        brew();//unterschiedlich-> Algo Subklasse entscheiden
        pourInCup();
        addCondiments(); // auch unterschiedlich -> Analog
    }
}
```

TEMPLATE METHODE

Template Methode Pattern definiert ein Skelett für einen Algorithmus in einer Methode, wobei Schritte an die Subklassen delegiert werden. Subklassen redefinieren gewissen Schritte ohne die Struktur zu verändern.

TEMPLATE METHODE

- Let's code

BEISPIEL EINKAUFSTASCHE

- Erstellen Sie eine Klasse Artikel mit Attributen um den Namen und die Anzahl (double) zu speichern.
- Erstellen Sie eine Klasse Einkaufstasche in der Sie mehrere Artikel speichern können. Wählen Sie selbst in welcher Datenstruktur Sie die Artikel speichern möchten. Fügen Sie Methoden hinzu
  - um einen Artikel hinzuzufügen
  - um einen Artikel abzurufen
- Erstellen Sie einen Iterator um ein Einkaufstaschen Objekt in einer for each Schleife verwenden zu können.
  - Erweiterung: next() soll Exception werfen
    - https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html#next--