MEMENTO

Design Pattern - Behavioral (Verhaltensmuster)

Ziel: Internen Zustand eines Objektes speichern & wiederherstellen ohne die Kapselung zu verletzen

Klassen: Originator, Memento, Caretaker

Synonyme: Snapshot, Undo-Pattern

Warum existiert das Pattern?

Um Objekten zu ermöglichen, ihren Zustand zu sichern & bei Bedarf zurückzusetzen,
ohne dass andere Objekte Einblick in Ihre internen Daten erhalten
→ schützt Kapselung
→ ermöglicht Undo-/Redo-Funktionalität

Originator

- Objekt, dessen Zustand gespeichert/restauriert wird

```
class TextEditor {
  text: string = "";

  setText(newText: string): void {
    this.text = newText;
    console.log("TextEditor: aktueller Text:", this.text);
  }

  saveToMemento(): Memento {
    console.log("TextEditor: Zustand wird gespeichert...");
    return new Memento(this.text);
  }

  restoreFromMemento(memento: Memento): void {
    this.text = memento.textState;
    console.log("TextEditor: Text wiederhergestellt:", this.text);
  }
}
```

Einsatzgebiete

- Texteditoren *(Undo/Redo)*
- Spiele *(Spielstände speichern)*
- Konfigurationsverwaltung
- Datenbank-Transaktionen
- Formular-Wiederherstellung

Memento

- Speicherobjekt, das den gespeicherten Zustand kapselt
- keine eigene Logik, nur „getter” - Funktionen

```
class Memento {
  textState: string;

  constructor(text: string) {
    this.textState = text;
  }
```

Caretaker

- verwaltet Mementos, ohne deren Inhalt zu verändern
- übergibt Memento an Originator
- bspw. mit Stack:

```
class Caretaker {
  private mementos: Memento[] = [];
  private textEditor: TextEditor;

  constructor(textEditor: TextEditor) {
    this.textEditor = textEditor;
  }

  save(): void {
    console.log("Caretaker: Zustand speichern...");
    this.mementos.push(this.textEditor.saveToMemento());
  }

  undo(): void {
    if (this.mementos.length === 0) {
      console.log("Caretaker: Kein Zustand zum Rückgängig machen.");
      return;
    }

    const lastMemento = this.mementos.pop();
    console.log("Caretaker: Letzte Änderung rückgängig...");
    if (lastMemento) {
      this.textEditor.restoreFromMemento(lastMemento);
    }
  }
}
```

Vor & Nachteile

+ Wahrung der Kapselung	− Hoher Speicherverbrauch bei vielen Zuständen
+ Einfache Undo-/Redo-Mechanismen	− Komplexere Verwaltung großer Objekte
+ Mehrere Zustände können gespeichert werden	− Manuelle Speicherfreigabe nötig