Шаблони за проектиране за поведение и архитектура

(Behavioral and Architectural Design Patterns)

Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/

Съдържание

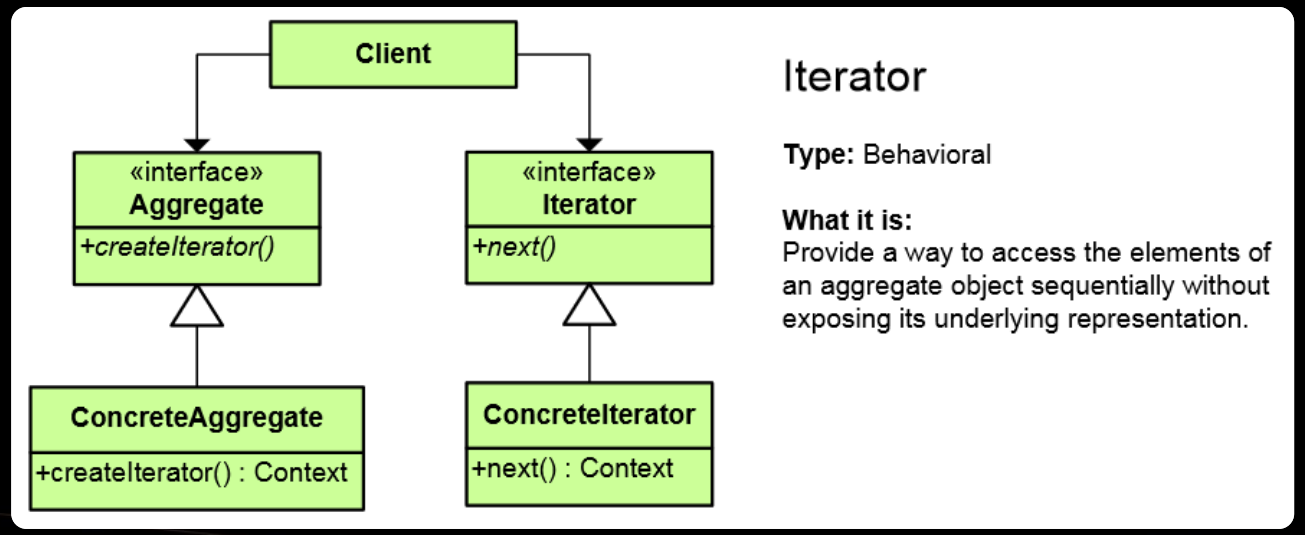
1. Шаблони в поведението
2. Шаблони в архитектурата

Шаблони в поведението

- Шаблоните в поведението се занимават с комуникацията (взаимодействието) между обектите
  - Било чрез разпределяне на отговорностите между обектите
  - Или чрез капсулиране на поведението в един обект и делегирането на заявките към него
- Увеличава гъвкавостта в комуникацията между класовете
- Класически шаблони в поведението:
  - Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Null Object, Observer, State, Strategy, Template Method, Visitor

Шаблон Iterator

- Iterator позволява достъп до елементите на съставен обект без разкриване на текущата му реализация
- Множество начини за обхождане на структура от данни
- Унифициран интерфейс за обхождане на различни структури от данни



Iterator – пример

```
public interface IEnumerator {
    bool MoveNext();
    object Current { get; }
    void Reset();
}
```

```
public interface IEnumerable {
    IEnumerator GetEnumerator();
}
```

```
private class ConcreteEnumerator : IEnumerator {
    // Implement IEnumerator interface
}
```

```
var enumerator = someObject.GetEnumerator();
enumerator.Reset();
while (enumerator.MoveNext()) {
    // Process the enumerator.Current
}
```

Iterator – примери за реална употреба

- `IEnumerable<T>` / `foreach` в C#

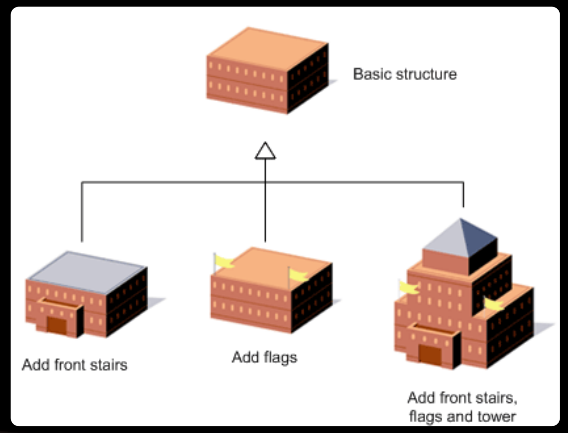
```
IEnumerator<T> GetEnumerator()
{
    foreach(var element in this.array)
    {
        yield return element;
    }
}
```

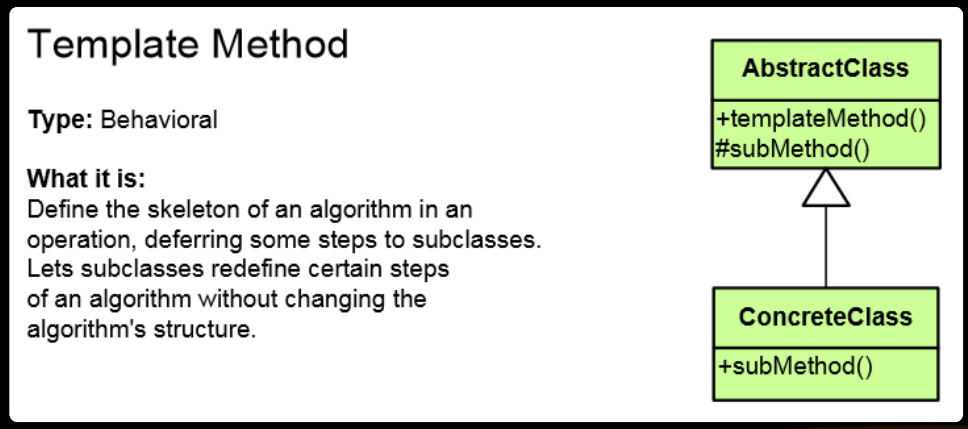
- `Iterable<T>` в Java

```
public boolean hasNext() {
    if (count < str.length()) { return true; }
    else return false;
}
public Character next() {
    return str.charAt(count++);
}
```

Template Method шаблон

- Template Method дефинира основната част от алгоритъм в метод и оставя част от реализацията на подкласовете си
  - Това позволява подкласовете да предефинират реализацията на части от алгоритъма
  - Но не им дава възможност да променят структурата на алгоритъма





Template Method – пример

```
public abstract class HotDrink {
    public void PrepareRecipe()
    {
        BoilWater(); Brew(); PourInCup(); AddSpices();
    }
    protected abstract void Brew();
    protected abstract void AddSpices();
    private void BoilWater() { ... }
    private void PourInCup() { ... }
}
public class Coffee : HotDrink {
    protected override void Brew() { ... }
    protected override void AddSpices() { ... }
}
public class Tea : HotDrink {
    protected override void Brew() { ... }
    protected override void AddSpices() { ... }
}
```

Реализирано в подкласове

Template Method – примери за реална употреба

- `Thread.run()` в Java

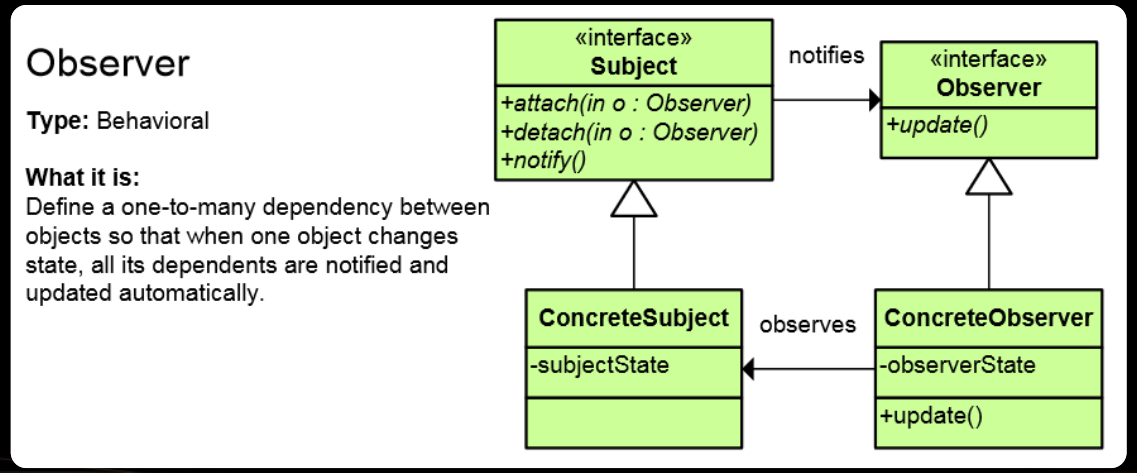
```
Thread thread = new Thread(){
    public void run() {
        System.out.println("Thread is running.");
    }
};

thread.start();
```

- `Thread.Start()` в .NET

```
Thread thread = new Thread(
    () => Console.WriteLine("Thread is running."));
thread.Start();
```

Шаблон Observer

- Observer представя интерфейс, позволяващ на обектите да комуникират помежду си без конкретно знание един за друг
- Известен е и като Publish-Subscribe шаблон
- Обект информира друг обект за своето състояние, без да се знае кои и какви са тези обекти

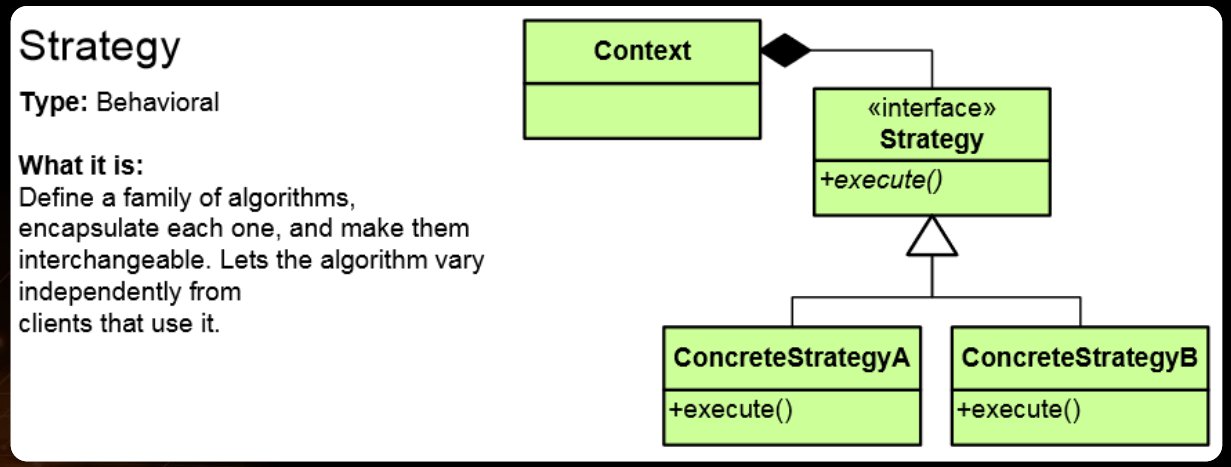


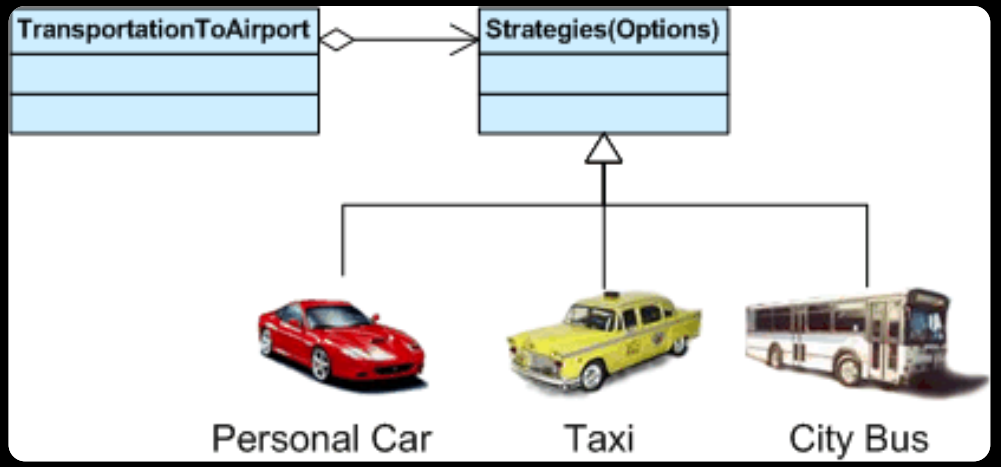
Observer – примери за реална употреба

- Събития и обработчици на събития в .NET
  - Източниците на събития (компонентите) публикуват събития (например **`Button`**)
  - Събитията в .NET имат механизъм за абониране (например **`Click`**)
- **`java.util.Observable`** / **`java.util.Observer`**
  - Класически observer шаблон
- **`ActionListener`** в Java
  - **`java.awt.event.ActionListener`** има **`actionPerformed()`**
  - **`java.awt.Button`** има **`addActionListener()`**

Шаблон Strategy

- Strategy капсулира алгоритъм в клас
  - Прави алгоритмите взаимозаменяеми
    - Всеки алгоритъм може да работи без промяна със същите данни
    - Клиентът може да ползва без промяна всеки един алгоритъм





Strategy – пример

```
abstract class SortStrategy {
   public abstract void Sort(IList<object> list);
}
```

```
class QuickSort : SortStrategy {
   public override void Sort(IList<object> list) { … }
}
```

```
class MergeSort : SortStrategy {
   public override void Sort(IList<object> list) { … }
}
```

```
class SortedList {
   private IList<object> list = new List<object>();
   public void Sort(SortStrategy strategy) {
      // sortStrategy can be passed in constructor
      sortStrategy.Sort(list);
   }
}
```

Strategy – примери за реална употреба

- `IComparer<T>`, `Cloneable<T>` в .NET
  - Сортирането ползва `IComparer<T>` за стратегия за сравняване
  - `Cloneable<T>` е стратегия за клониране на обекти
- `Comparer<T>` в Java
  - Сортирането ползва `Comparer<T>` за стратегия за сравняване
  - `TreeMap<K, V>` ползва `Comparer<T>` за стратегия за подреждането на върховете в дървото

Шаблони в архитектурата

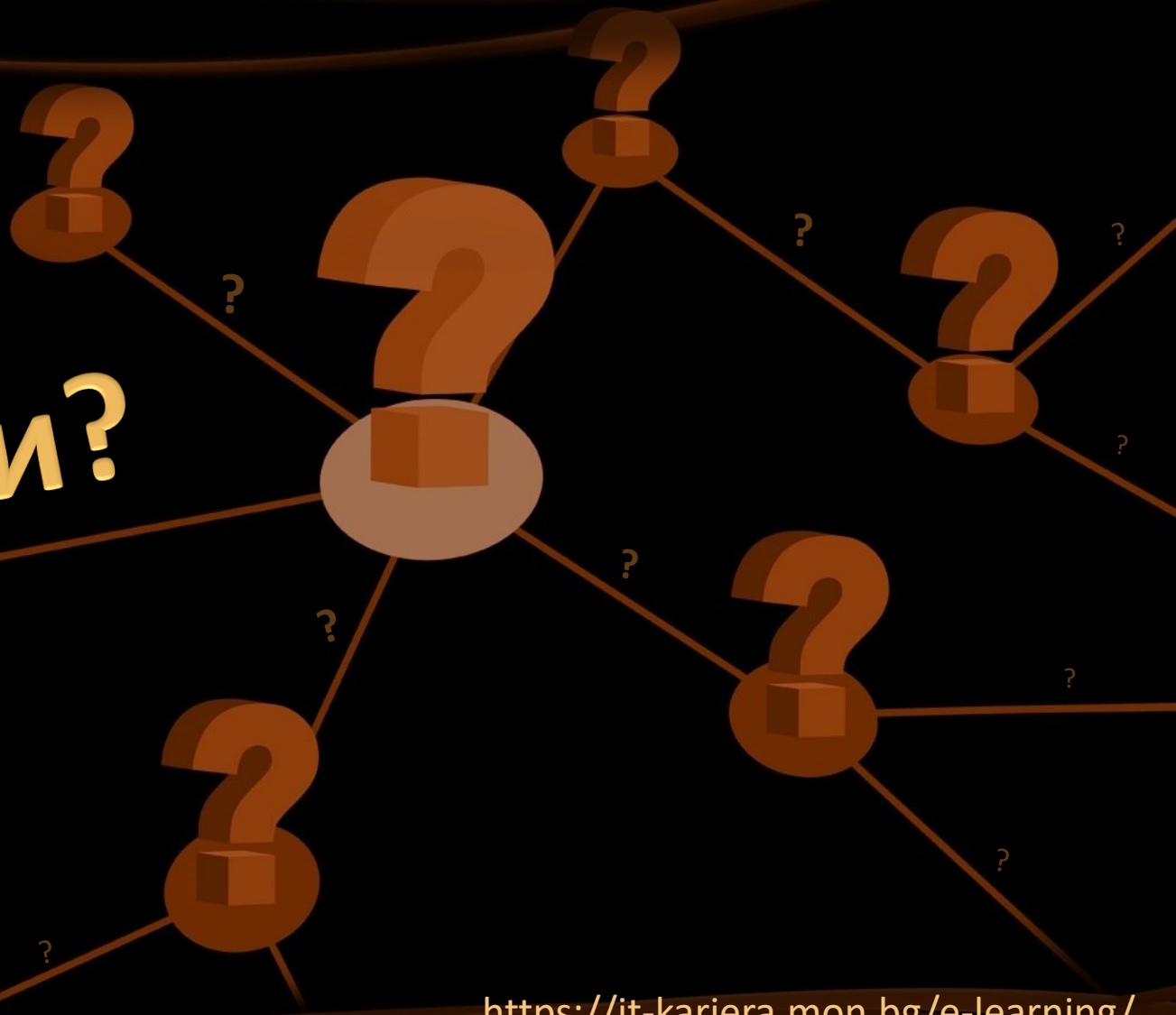
- Client-Server Model – client ↔ server
- 3-tier Architecture – front-end ↔ logic tier ↔ back-end
- Multi-tier Architecture – front-end ↔ tier ↔ tier ↔ back-end
- Model-View-Controller (MVC) – за създаване на UI
- Model-View-Presenter (MVP) – за създаване на UI
- Model-View-ViewModel (MVVM) – за създаване на UI
- Front Controller – за изпращане на заявки в Web приложения
- Active Record – обвива таблици с класове + CRUD операции

Обобщение

- Шаблони в поведението
  - Iterator, Observer, Template Method, Strategy
- Шаблони в архитектурата
  - Client-Server Model
  - 3-tier and Multi-tier Architecture
  - Model-View-Controller (MVC), Model-View-Presenter (MVP) and Model-View-ViewModel (MVVM)
  - Front Controller, Active Record

Шаблони за проектиране за поведение и архитектура

Въпроси?



https://it-kariera.mon.bg/e-learning/

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист"

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**