

Software Engineering – Structural design patterns

Adam Krechowicz

1 Structural design patterns

Structural design pattern allows to solve problems that occurs during creating class or object structure:

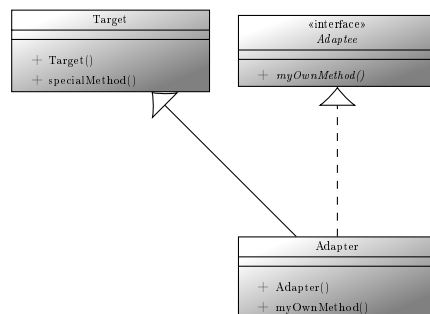
This group contains the following patterns:

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

1.1 Adapter

Is used when we want to:

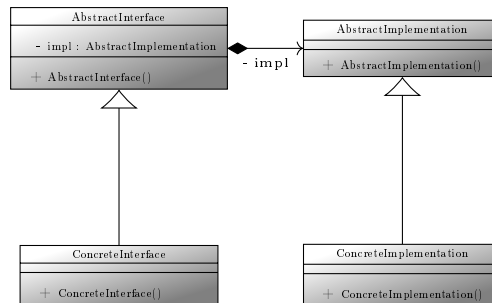
- Use two elements
- with incompatible interfaces
- Pattern allows to change the interface of class



1.2 Bridge

Is used when we want to:

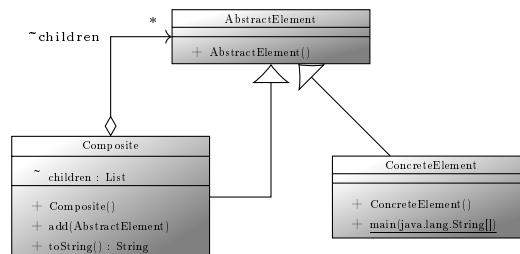
- Separate interface from implementation
- so that we can freely extend both



1.3 Composite

Is used when we want to:

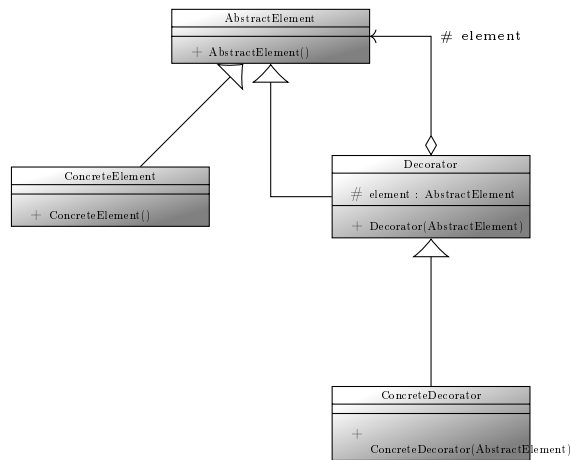
- Create universal, hierarchical object structure
- to freely compose objects
- and treat them as one



1.4 Decorator

Is used when we want to:

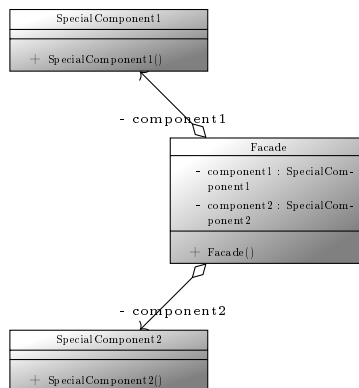
- Dynamically add new responsibility to object
- Prevents class explosion
- Allows to create many class combinations



1.5 Facade

Is used when we want to:

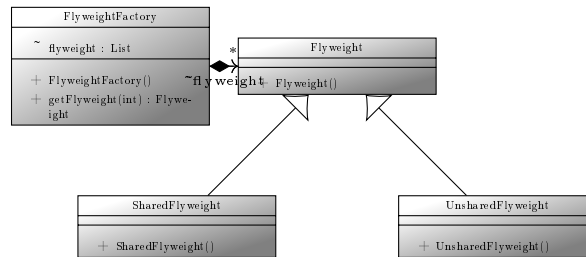
- Create uniform interface for a set of subsystem
- Creates higher level interface



1.6 Flyweight

Is used when we want to:

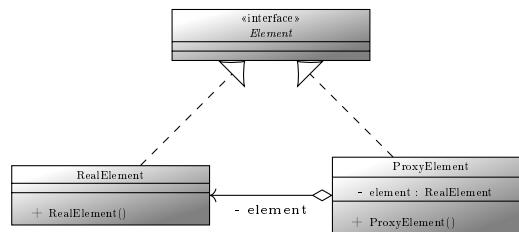
- Manage a very big set of objects
- by sharing resources
- Allows to save memory



1.7 Proxy

Is used when we want to:

- Control access to the object
- in invisible way



2 Tasks to complete

1. Identify the place for structural design patten in the system
2. Describe the problem that justifies the need of design pattern
3. Describe the pattern its theory and its place in the system
4. Create class diagram for pattern
5. Create source code that implements pattern
6. Create test source code for pattern

Each member of the team should pick other pattern.

Results should be placed in appropriate article each patter for section (`<section class="pattern">`). The structure of the section is as follows:

- `h5` – pattern name
- `<p class="author">` – author of pattern
- `<div class="pattern-problem">` – pattern problem

- `<div class="pattern-description">` – pattern description
- `<p class="uml pattern-diagram">` – pattern class diagram
- `<pre class="pattern-code"><code class="lang-java">` – source code
- `<pre class="pattern-test-code"><code class="lang-java">` – test source code