POLA-POLA PERANCANGAN (PPP)

Pengantar
Tujuan perkuliahan

- Memahami pengertian dan karakteristik pola perancangan
- Memahami klasifikasi pola perancangan
- Memahami keuntungan dan kelemahan penggunaan pola
Motivation

- OOD methods emphasize design notations
  - Fine for specification, documentation
- But OOD is more than just drawing diagrams
  - Good draftsmen $\rightarrow$ good designers
- Good OO designers rely on lots of experience
  - At least as important as syntax
- Most powerful reuse is *design* reuse
  - Match problem to design experience
  - Not to ‘reinventing the wheel’
  - Apply the same design solution to the same problems in different context
Recurring design structures

- OO systems exhibit recurring structures that promote
  - abstraction
  - flexibility
  - modularity
  - information hiding
- Therein lies valuable design knowledge
- Problem → capturing, communicating & applying this knowledge to be reused in many different contexts
Design pattern

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice

(Christopher Alexander – a building architect)

Then...

Popularized by Gamma, Helm, Johnson and Vlissides (The gang of four, Go4, GOF)
Why?

- Designing OO software is hard
- Designing *reusable* OO software – harder
- Experienced OO designers make good design
- New designers tend to fall back on non-OO techniques used before
- Experienced designers know something – what is it?
- Expert designers know *not* to solve every problem from first principles
- They *reuse* solutions
- These patterns make OO designs more flexible, elegant, and ultimately reusable
A design pattern

- Abstracts a recurring design structure
  - comprises class and/or object
  - dependencies
  - structures
  - interactions
  - conventions
- Names & specifies the design structure explicitly
- Distills design experience
- Is good, if it:
  - be as general as possible
  - contains a solution that has been proven to effectively solve the problem in the indicated context
Design pattern template

Name
- A meaningful name that reflects the knowledge embodied by the pattern

Problem
- Describes the problem that the pattern addresses

Context
- The general situation in which the pattern applies, including the application domain

Forces
- The issues or concerns to consider when solving the problem, including limitations and constraints

Solution
- The recommended way to solve the problem in the given context → should resolve all the forces
Goals

- Codify good design
  - distill & generalize experience
  - aid to novices & experts alike
- Give design structures explicit names
  - common vocabulary
  - reduced complexity
  - greater expressiveness
- Capture & preserve design information
  - articulate design decisions succinctly
  - improve documentation
- Facilitate restructuring/refactoring
  - patterns are interrelated
  - additional flexibility
GOF’s categories

Based on scope (domain over which a pattern applies):

- **Class** → concerns with class and the relationship to its sub-classes at the compile-time (static)
- **Object** → concerns with objects and their relationships at the run-time (dynamic)
Based on purposes (reflects what a pattern does):

- **Creational** → concerns with the construction of object instances
  - Class  → defer its object creation to subclasses
  - Object → defer part of its object creation to another object

- **Structural** → how objects are composed into larger groups
  - Class  → structure via inheritance
  - Object → structure via composition

- **Behavioral** → how responsibilities are distributed
  - Class  → algorithms/control via inheritance
  - Object → algorithms/control via object groups/composition
# Types

<table>
<thead>
<tr>
<th>Scope</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td></td>
</tr>
<tr>
<td>Factory Method</td>
<td>Adapter (class)</td>
</tr>
<tr>
<td>Abstract Factory</td>
<td>Adapter (object)</td>
</tr>
<tr>
<td>Builder</td>
<td>Bridge</td>
</tr>
<tr>
<td>Prototype</td>
<td>Composite</td>
</tr>
<tr>
<td>Singleton</td>
<td>Decorator</td>
</tr>
<tr>
<td></td>
<td>Flyweight</td>
</tr>
<tr>
<td></td>
<td>Facade</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td>Object</td>
<td></td>
</tr>
</tbody>
</table>

Bahan Kuliah PPP - Pengantar | Tri A. Kurniawan, S.T, M.T, Ph.D
Example: Observer (behavioral)

- **Context:**
  - When an association is created between two classes, the code for the classes becomes inseparable.
  - If you want to reuse one class, then you also have to reuse the other.

- **Problem:**
  - How do you reduce the interconnection between classes, especially between classes that belong to different modules or subsystems?

- **Forces:**
  - You want to maximize the flexibility of the system to the greatest extent possible.
Example: Observer (behavioral)

Solution:

```java
Subject
attach(Observer)
detach(Observer)
notify()

for all o in observers {
  o.update()
}

ConcreteSubject
getState()
subjectState

return subjectState

Observer
update()

ConcreteObserver
update()
observerState

observerState = subject.getState()
```
Example: Observer (behavioral)

Example:
Benefits and dangers of using patterns

+ Reuse of generic solutions
+ They provide a vocabulary for discussing the problem domain at a higher level of abstraction
+ Enhance understanding, restructuring, & team communication

– May limit creativity
– The use of patterns may lead to over-design

Organizational impact
– The use of patterns requires care and planning
– Patterns must be used with intelligence
Patterns have been identified in many different application domains and are applicable at many different stages of the software development process.

Patterns are not a panacea:

- Whenever you see an indication that a pattern should be applied, you might be tempted to blindly apply the pattern.
- This can lead to unwise design decisions.
- Always understand in depth the forces that need to be balanced, and when other patterns better balance the forces.
- Make sure you justify each design decision carefully.

Summary