8. Design Patterns

Design Patterns Need

- Very hard to design reusable OO Software
- Design must be specific to problem on hand
- Design must be general enough to address future problems and requirements
- Almost impossible to do it right the first time
- Experienced designers don’t reinvent the wheel
- They use good solutions again and again
- Find & use recurring patterns of classes & communicating objects
Design Patterns Benefits

- Design Pattern systematically
  - names
  - explains
  - evaluates an important recurring design

- Easier to reuse successful designs & architectures

- More accessible to developers of new systems

But, What is a 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 even doing it the same way twice”
Design Pattern?

• Description of communicating objects & classes that are customized to solve a general design problem in a particular context

• Pattern has four essential elements:
  – Pattern Name
  – Problem
  – Solution
  – Consequences

Classification

• Several Design Patterns are identified & more will be identified
• Classification helps to group these
• Helps you to understand the related patterns
Classification…

- **Purpose:**
  - Creational - concerns process of object creation
  - Structural - concerns composition of classes or objects
  - Behavioral - concerns way in which classes or objects interact & distribute responsibilities

- **Scope:**
  - Class - deals with relationship between classes and their subclasses - static
  - Object - deals with relationship between objects - dynamic

---

### Classification of Catalog of Patterns

<table>
<thead>
<tr>
<th>Scope</th>
<th>Class</th>
<th>Factory Method</th>
<th>Adapter</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Abstract Factory</td>
<td>Adapter</td>
<td>Chain_of Responsibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Builder</td>
<td>Bridge</td>
<td>Command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Prototype</td>
<td>Composite</td>
<td>Iterator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Singleton</td>
<td>Decorator</td>
<td>Mediator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Façade</td>
<td>Memento</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Flyweight</td>
<td>Observer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Proxy</td>
<td>State</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Strategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Visitor</td>
</tr>
</tbody>
</table>

Venkat Subramaniam

DP-7

Venkat Subramaniam

DP-8