

Organizing the catalog

Designed by Rita Ganatra

The different sections of Design Pattern

The different sections of Design Pattern

Basically, design patterns are categorized into two parts:

- Core Java (or JSE) Design Patterns.
- JEE Design Patterns.



Core Java Design Patterns

In core java, there are mainly three types of design patterns, which are further divided into their sub-parts:

1. Creational Design Pattern
2. Structural Design Pattern
3. Behavioral Design Pattern



J2EE Design Patterns

J2EE design patterns are built for developing the Enterprise Web-based Applications.

In J2EE , there are mainly three types of design patterns, which are further divided into their sub-parts:

1. Presentation Layer Design Pattern

- Intercepting Filter Pattern
- Front Controller Pattern
- View Helper Pattern
- Composite View Pattern

2. Business Layer Design Pattern

- Business Delegate Pattern
- Service Locator Pattern
- Session Facade Pattern
- Transfer Object Pattern

3. Integration Layer Design Pattern

- Data Access Object Pattern
- Web Service Broker Pattern

Organizing the catalog

Design patterns vary in their granularity(size or scope of the smallest piece of data) and level of abstraction which means they can address problems at different scales and provide solutions that range from very concrete to highly abstract.

As there are many design patterns, we can organize or classify patterns to learn them faster.

Here we classify the patterns along two dimensions:

- one is purpose which reflects what a pattern does
- and the second one is scope which specifies whether the pattern applies to classes or objects.

Classification of Design Pattern

Along the purpose dimension, patterns are classified into three types:

- 1) Creational patterns
- 2) Structural patterns and
- 3) Behavioral patterns



Purpose / Scope		Purpose		
		Creational (5)	Structural (7)	Behavioral (11)
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Patterns Purpose

- The creational patterns focus on the process of object creation.
- The structural patterns concern is, the structure of classes and objects
- The behavioral patterns focus on how the classes and objects collaborate with each other to carry out their responsibilities.



Patterns Scope

Along the scope dimension, patterns are classified into two types:

1) Class patterns -

Class patterns deal with the relationships between classes which is through inheritance.

So, these class relationships are static at compile time.

2) Object patterns -


Object patterns deal with object relationships which can be changed at run-time and these are dynamic.

Almost all patterns use inheritance to some degree.



Creational Patterns

- Creational patterns deals with object creation process.
- Creational design patterns are further classified into:
 - class-creational patterns
 - object-creational patterns.
- **Class-creational patterns deals with instantiation of a subclass based on the context.**
- **Object-creational patterns deals with delegation of object creation process to another object.**
- The creational design patterns allow configuring of a software system as a system with —product ll objects that vary widely in **structure and functionality**. Such configuration can be static, i.e., specified at compile-time, or dynamic, i.e., specified at run-time.

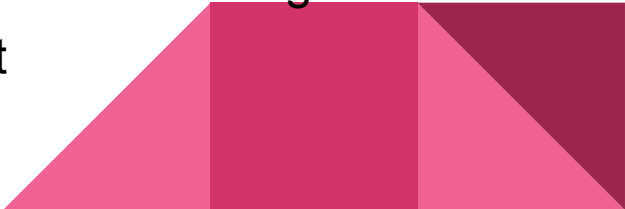
- In software engineering, creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. The basic form of object creation could result in design problems or added complexity to the design. Creational design patterns solve this problem by somehow controlling this object creation.
 - There are five creational patterns:
 1. Abstract Factory -Creates an instance of several families of classes
 2. Builder - Separates object construction from its representation
 3. Factory Method - Creates an instance of several derived classes
 4. Prototype - A fully initialized instance to be copied or cloned
 5. Singleton - A class of which only a single instance can exist
- 

Structural Patterns

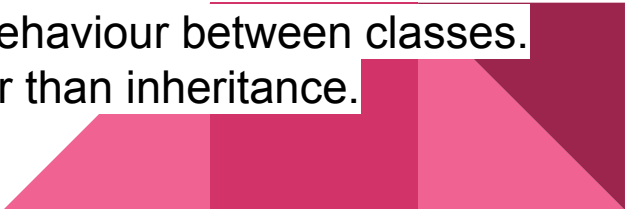
- Structural patterns are concerned with how classes and objects can be arranged to form larger structures.
- Structural class patterns use inheritance to compose interfaces or different implementations.
- For example, multiple inheritance can be used to combine features from two or more classes into a single class. This allows two or more independently developed class libraries to work together.
- Rather than composing interfaces or implementations, Structural object patterns specify a way to create new objects to realize new functionality. The flexibility of object composition allows us to change the composition at run-time, which is impossible with static class composition.

Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

There are seven structural GOF patterns. They are:

- Adapter pattern - Match interfaces of different classes
 - Bridge pattern - Separates an object's interface from its implementation
 - Composite pattern - A tree structure of simple and composite objects
 - Decorator pattern - Add responsibilities to objects dynamically
 - Façade pattern - A single class that represents an entire subsystem
 - Flyweight pattern - A fine-grained instance used for efficient sharing
 - Proxy pattern - An object representing another object
- 

Behavioral Design Patterns

- Behavioral design patterns is all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects
 - Behavioral patterns are concerned with algorithms and the assignment of responsibility between objects.
 - Behavioural patterns describe not just patterns of objects or classes but also the patterns of communication between them.
 - These patterns characterize complex control flow that is difficult to follow at run-time.
 - They shift your focus away from flow of control to let you concentrate just on the way objects are interconnected.
 - Behavioral class patterns use inheritance to distribute behaviour between classes.
 - Behavioral object patterns use object composition rather than inheritance.
- 

- For example, a behavioral object pattern can describe how a group of object might cooperate to perform a task that no single object can carry out by itself. A typical example is the Observer pattern from the Smalltalk (Model/View/Controller paradigm). Views are used to show the state of data (contained in Model) and they are observers of this data. Whenever a model changes its state all views are notified and they can update the representation of the data in views.



- Chain of responsibility - A way of passing a request between a chain of objects
- Command - Encapsulate a command request as an object
- Interpreter - A way to include language elements in a program
- Iterator - Sequentially access the elements of a collection
- Mediator - Defines simplified communication between classes
- Memento - Capture and restore an object's internal state Null Object - Designed to act as a default value of an object
- Observer - A way of notifying change to a number of classes
- State - Alter an object's behavior when its state changes
- Strategy - Encapsulates an algorithm inside a class
- Template method - Defer the exact steps of an algorithm to a subclass
- Visitor - Defines a new operation to a class without change.

Thank You

