GoF (Gang of Four) Design Pattern Catalog

	Intent	Motivation	Examples
Creational			
Singleton(*)	Ensure a class only has one instance, and provide a global point of access to it.	How do we ensure that a class has only one instance and that the instance is easily accessible?	java.lang.Runtime#getRuntime() java.awt.Desktop#getDesktop()
Prototype(**)	Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.	You need to make new instances by copying template instances. When your application has to create objects without knowing their type or any details of how to create them. When creating a large/complex object that is resource intensive, cloning works better than using the constructor sometimes.	3D printing The clone() method in Java (Object or its subclasses that implement java.lang.Cloneable)
Factory Method(**)	Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.	Client may only know that it needs an object of a certain type but does not know the exact subtype to be selected	java.util.Calendar#getInstance() java.util.ResourceBundle#getBundle() java.text.NumberFormat#getInstance()
Builder(***)	Separate the construction of a complex object from its representation so that the same construction process can create different representations.	You take a step-by-step approach to construct a complex object.	Complex objects created in assembly lines in a factory. Order with multiple order-line items. Java InputStream and Outstream subclasses
Abstract Factory(***)	Provide an interface for creating families of related or dependent objects without specifying their concrete classes.	You need to create different series of products that belong to different inheritance hierarchies	javax.xml.parsers.DocumentBuilderFactory#newInstance() javax.xml.transform.TransformerFactory#newInstance() javax.xml.xpath.XPathFactory#newInstance()
Structural			
Adapter(*)	Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.	Class library cannot be used because its interface is incompatible with the interface required by an application	Electricity plug adapter, Stack implemented with an array, java.util.Arrays#asList()
Composite(**)	Compose objects into tree structures to represent part-whole hierarchies.	You want to represent part-whole or parent-child hierarchies of objects	Tree structure java.awt.Container

	Composite lets clients treat individual objects and compositions of objects uniformly.		
Façade(**)	Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.	To simplify complex interfaces exposed by classes in a set of subsystems	User interface of some home appliances (microwave oven, washer, etc) Company's front desk or telephone switchboard
Bridge(**)	Decouple an abstraction from its implementation so that the two can vary independently.	Avoid permanent binding between abstraction and implementation	JDBC-ODBC bridge, JVM spec and implementation, GC spec and implementation, etc
Flyweight(***)	Use sharing to support large numbers of fine-grained objects efficiently.	Your application deals with a potentially large number of objects of the same type, which may take a huge amount of memory space	DB connection pool, Java String Pool
Proxy(*)	Provide a surrogate or placeholder for another object to control access to it.	a client does not or cannot reference an object directly, but wants to still interact with the object	RMI or CORBA Data source for a DBMS Web Proxy Reference counting module inside Java Garbage Collector
Decorator(**)	Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.	Sometimes we want to add responsibilities to individual objects, not to an entire class	FileInputStream fis = new FileInputStream("/objects.gz"); BufferedInputStream bis = new BufferedInputStream(fis); GzipInputStream gis = new GzipInputStream(bis);
Behavioral			
Chain of Responsi-bility(***)	Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.	From the handlers' perspective, multiple resources with different skills, different roles, or different responsibilities organized in a chain structure	Java Exception Handling, Template hierarchy in WordPress core, Customer Services Model(rep, senior rep, supervisor, etc)
Command(***)	Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.	Sometimes it's necessary to issue requests without knowing anything about the operation being requested or the receiver of the request.	Database operations modelled as commands that are part of a transaction (which also support roll-back) Implement actionPerformed(ActionEvent e) with Command Pattern. Replace public void actionPerformed(ActionEvent e){ Object o = e.getSource();

			if (o == fileNewMenuItem)
			doFileNewAction();
			else if (o == fileOpenMenuItem)
			doFileOpenAction();
			else if (o == fileOpenRecentMenuItem)
			
			doFileOpenRecentAction();
			else if (o == fileSaveMenuItem)
			doFileSaveAction();
			// and more ...
			}
			With
			public void actionPerformed(ActionEvent e){
			Command command = (Command)e.getSource();
			command.execute();
			}

Strategy(**)	Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.	The client and a strategy it uses to perform a function should not be hard-wired together -to keep the client lean; -or change of strategy easier; -or add new strategies without affecting client code.	Different shipping methods (UPS, FedEx, USPS, etc) Different file compression methods (zip, rar, gzip, etc)
Iterator(**)	Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.	An aggregate object such as a list or hash map should allow a way to traverse its elements without exposing its internal structure	All implementations of java.util.Iterator All implementations of java.util.Enumeration
Template Method(*)	Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.	Sometimes you want to specify the order of operations that a method uses, but allow subclasses to provide their own implementations of some of these operations	java.io.InputStream java.io.OutputStream java.io.Reader java.io.Writer. java.util.AbstractList java.util.AbstractSet java.util.AbstractMap. javax.servlet.http.HttpServlet
Mediator(**)	Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.	Need to centralize complex communications and control between related objects	Airport control tower; A Chatroom software application for multiple users; Auction;
Observer(**)	Define a one-to-many dependency between objects so that when	The multiple subscribers are dependent on the state of the publisher; therefore	Publisher-subscriber java.util.Observer java.util.Observable

	one object changes state, all its dependents are notified and updated automatically.	they need to be notified of any change of state with the publisher.	All implementations of java.util.EventListener
Memento(**)	Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.	You must save state information somewhere so that you can restore objects to their previous states	"Undo" button in Microsoft Word. "Restore Data" button on a GUI. Having snapshots when repairing/dismantling a machine (with an assistant taking the snapshots for you)
State(**)	Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.	Object behavior varies for different states the object is in.	State machine, vending machine All-in-one remote control for Fan, Light, TV, etc
Interpreter (***)	Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.	Sometimes we need to create a rule-based language to represent expressions or statements or sentences, we want to describe how to define the grammar (the rules) and how to interpret the language.	Evaluating prefix-operator expressions like "- + 10 5 + - 8 2 9" which is equivalent to (10 + 5) – ((8 – 2) + 9) Roman Numerals Convertor XIV = 10 + (5 – 1)
Visitor(***)	Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.	Visitor allows us to be even more flexible to perform different functions on a collection of objects (that do not need to be part of the same hierarchy) with "double dispatch".	-People do tax return by themselves with a method – doTaxReturn(); -Or they delegate it to a professional tax preparer by passing all information over with taxProparer.doTaxReturn(incomeStatements); - Or they accept a visit by a professional preparer to do tax return, not having to pass any information(you come and visit me). This is like a "double dispatch" in Software Engineering, which sits at the core of the Visitor pattern.