Java Design Patterns Reference and Examples

 GoF Creational Patterns...

**GoF Creational Patterns**

**Abstract Factory**

Sets of methods to make various objects.  
Abstract Factory is recognizable by the creational methods returning the factory itself which in turn can be used to create another abstract or interface type  
Eamples:  
javax.xml.parsers.DocumentBuilderFactory#newInstance()  
javax.xml.transform.TransformerFactory#newInstance()  
javax.xml.xpath.XPathFactory#newInstance()

**Builder**

Make and return one object various ways.  
Builder is recognizeable by creational methods returning the instance itself  
Examples:  
java.lang.StringBuilder#append() (unsynchronized)  
java.lang.StringBuffer#append() (synchronized)  
java.nio.ByteBuffer#put() (also on CharBuffer, ShortBuffer, IntBuffer, LongBuffer, FloatBuffer and DoubleBuffer)  
javax.swing.GroupLayout.Group#addComponent()  
All implementations of java.lang.Appendable

**Factory Method**

Methods to make and return components of one object various ways.  
Factory Method is recognizable by creational methods returning an implementation of an abstract/interface type  
Examples:  
java.util.Calendar#getInstance()  
java.util.ResourceBundle#getBundle()  
java.text.NumberFormat#getInstance()  
java.nio.charset.Charset#forName()  
[java.net](http://java.net/).URLStreamHandlerFactory#createURLStreamHandler(String) (Returns singleton object per protocol)  
java.util.EnumSet#of()  
javax.xml.bind.JAXBContext#createMarshaller() and other similar methods

**Prototype**

Make new objects by cloning the objects which you set as prototypes.  
Prototype is recognizeable by creational methods returning a different instance of itself with the same properties  
Example:  
java.lang.Object#clone() (the class has to implement java.lang.Cloneable)

**Singleton**

A class distributes the only instance of itself.  
Singleton is recognizeable by creational methods returning the same instance (usually of itself) everytime)  
Examples:  
java.lang.Runtime#getRuntime()  
java.awt.Desktop#getDesktop()  
java.lang.System#getSecurityManager()

 GoF Structural Patterns...

**GoF Structural Patterns**

**Adapter**

A class extends another class, takes in an object, and makes the taken object behave like the extended class.  
Adapter is recognizeable by creational methods taking an instance of different abstract/interface type and returning an implementation of own/another abstract/interface type which decorates/overrides the given instance)  
Examples:  
java.util.Arrays#asList()  
java.util.Collections#list()  
java.util.Collections#enumeration()  
[java.io](http://java.io/).InputStreamReader(InputStream) (returns a Reader)  
[java.io](http://java.io/).OutputStreamWriter(OutputStream) (returns a Writer)  
javax.xml.bind.annotation.adapters.XmlAdapter#marshal() and #unmarshal()

**Bridge**

An abstraction and implementation are in different class hierarchies.  
Bridge is recognizeable by creational methods taking an instance of different abstract/interface type and returning an implementation of own abstract/interface type which delegates/uses the given instance  
Examples:  
None comes to mind yet. A fictive example would be new LinkedHashMap(LinkedHashSet<K>, List<V>) which returns an unmodifiable linked map which doesn't clone the items, but uses them. The java.util.Collections#newSetFromMap() and singletonXXX() methods however comes close.

**Composite**

Assemble groups of objects with the same signature.  
Composite is recognizeable by behavioral methods taking an instance of same abstract/interface type into a tree structure  
Examples:  
java.awt.Container#add(Component) (practically all over Swing thus)  
javax.faces.component.UIComponent#getChildren() (practically all over JSF UI thus)

**Decorator**

One class takes in another class, both of which extend the same abstract class, and adds functionality.  
Decorator is recognizeable by creational methods taking an instance of same abstract/interface type which adds additional behaviour  
Examples:  
All subclasses of [java.io](http://java.io/).InputStream, OutputStream, Reader and Writer have a constructor taking an instance of same type.  
java.util.Collections, the checkedXXX(), synchronizedXXX() and unmodifiableXXX() methods.  
javax.servlet.http.HttpServletRequestWrapper and HttpServletResponseWrapper

**Facade**

One class has a method that performs a complex process calling several other classes.  
Facade is recognizeable by behavioral methods which internally uses instances of different independent abstract/interface types  
Examples:  
javax.faces.context.FacesContext, it internally uses among others the abstract/interface types LifeCycle, ViewHandler, NavigationHandler and many more without that the enduser has to worry about it (which are however overrideable by injection).  
javax.faces.context.ExternalContext, which internally uses ServletContext, HttpSession, HttpServletRequest, HttpServletResponse, etc.

**Flyweight**

The reusable and variable parts of a class are broken into two classes to save resources.  
Flyweight is recognizeable by creational methods returning a cached instance, a bit the "multiton" idea  
Example:  
java.lang.Integer#valueOf(int) (also on Boolean, Byte, Character, Short, Long and BigDecimal)

**Proxy**

One class controls the creation of and access to objects in another class.  
Proxy is recognizeable by creational methods which returns an implementation of given abstract/interface type which in turn delegates/uses a different implementation of given abstract/interface type  
Examples:  
java.lang.reflect.Proxy  
java.rmi.\*  
javax.ejb.EJB (explanation here)  
javax.inject.Inject (explanation here)  
javax.persistence.PersistenceContext

 GoF Behavorial Patterns...

**GoF Behavorial Patterns**

**Chain of Responsibility**

A method called in one class can move up a hierarchy to find an object that can properly execute the method.  
Chain Of Responsibility is recognizeable by behavioral methods which (indirectly) invokes the same method in another implementation of same abstract/interface type in a queue  
Examples:  
java.util.logging.Logger#log()  
javax.servlet.Filter#doFilter()

**Command**

An object encapsulates everything needed to execute a method in another object.  
Command is recognizeable by behavioral methods in an abstract/interface type which invokes a method in an implementation of a different abstract/interface type which has been encapsulated by the command implementation during its creation  
Examples:  
All implementations of java.lang.Runnable  
All implementations of javax.swing.Action

**Interpreter**

Define a macro language and syntax, parsing input into objects which perform the correct opertaions.  
Interpreter is recognizeable by behavioral methods returning a structurally different instance/type of the given instance/type; note that parsing/formatting is not part of the pattern, determining the pattern and how to apply it is  
Examples:  
java.util.Pattern  
java.text.Normalizer  
All subclasses of java.text.Format  
All subclasses of javax.el.ELResolver

**Iterator**

One object can traverse the elements of another object.  
Iterator is recognizeable by behavioral methods sequentially returning instances of a different type from a queue  
Examples:  
All implementations of java.util.Iterator (thus among others also java.util.Scanner!).  
All implementations of java.util.Enumeration

**Mediator**

An object distributes communication between two or more objects.  
Mediator is recognizeable by behavioral methods taking an instance of different abstract/interface type (usually using the command pattern) which delegates/uses the given instance  
Examples:  
java.util.Timer (all scheduleXXX() methods)  
java.util.concurrent.Executor#execute()  
java.util.concurrent.ExecutorService (the invokeXXX() and submit() methods)  
java.util.concurrent.ScheduledExecutorService (all scheduleXXX() methods)  
java.lang.reflect.Method#invoke()

**Memento**

One object stores another objects state.  
Memento is recognizeable by behavioral methods which internally changes the state of the whole instance  
Examples:  
java.util.Date (the setter methods do that, Date is internally represented by a long value)  
All implementations of [java.io](http://java.io/).Serializable  
All implementations of javax.faces.component.StateHolder

**Observer**

An object notifies other object(s) if it changes.  
Observer (or Publish/Subscribe) is recognizeable by behavioral methods which invokes a method on an instance of another abstract/interface type, depending on own state  
Examples:  
java.util.Observer/java.util.Observable (rarely used in real world though)  
All implementations of java.util.EventListener (practically all over Swing thus)  
javax.servlet.http.HttpSessionBindingListener  
javax.servlet.http.HttpSessionAttributeListener  
javax.faces.event.PhaseListener

**State**

An object appears to change its` class when the class it passes calls through to switches itself for a related class.  
State is recognizeable by behavioral methods which changes its behaviour depending on the instance's state which can be controlled externally  
Example:  
javax.faces.lifecycle.LifeCycle#execute() (controlled by FacesServlet, the behaviour is dependent on current phase (state) of JSF lifecycle)

**Strategy**

An object controls which of a family of methods is called. Each method is in its` own class that extends a common base class.  
Strategy is recognizeable by behavioral methods in an abstract/interface type which invokes a method in an implementation of a different abstract/interface type which has been passed-in as method argument into the strategy implementation  
Examples:  
java.util.Comparator#compare(), executed by among others Collections#sort().  
javax.servlet.http.HttpServlet, the service() and all doXXX() methods take HttpServletRequest and HttpServletResponse and the implementor has to process them (and not to get hold of them as instance variables!).  
javax.servlet.Filter#doFilter()

**Template**

An abstract class defines various methods, and has one non-overridden method which calls the various methods.  
Template is recognizeable by behavioral methods which already have a "default" behaviour definied by an abstract type  
Examples:  
All non-abstract methods of [java.io](http://java.io/).InputStream, [java.io](http://java.io/).OutputStream, [java.io](http://java.io/).Reader and [java.io](http://java.io/).Writer.  
All non-abstract methods of java.util.AbstractList, java.util.AbstractSet and java.util.AbstractMap.  
javax.servlet.http.HttpServlet, all the doXXX() methods by default sends a HTTP 405 "Method Not Allowed" error to the response. You're free to implement none or any of them.

**Visitor**

One or more related classes have the same method, which calls a method specific for themselves in another class.  
Visitor is recognizeable by two different abstract/interface types which has methods definied which takes each the other abstract/interface type; the one actually calls the method of the other and the other executes the desired strategy on it  
Examples:  
javax.lang.model.element.AnnotationValue and AnnotationValueVisitor  
javax.lang.model.element.Element and ElementVisitor  
javax.lang.model.type.TypeMirror and TypeVisitor  
java.nio.file.FileVisitor and SimpleFileVisitor  
javax.faces.component.visit.VisitContext and VisitCallback