DESIGN PATTERNS

NOMAN SHEIKH

OVERVIEW

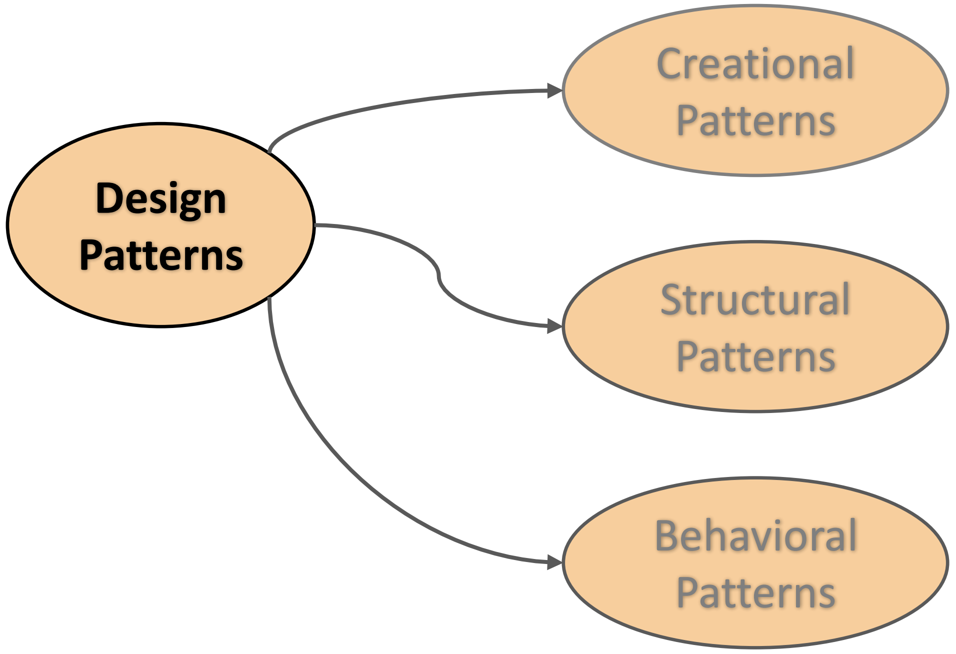
Design Patterns

❑ A general repeatable solution to a commonly occurring problem in software design

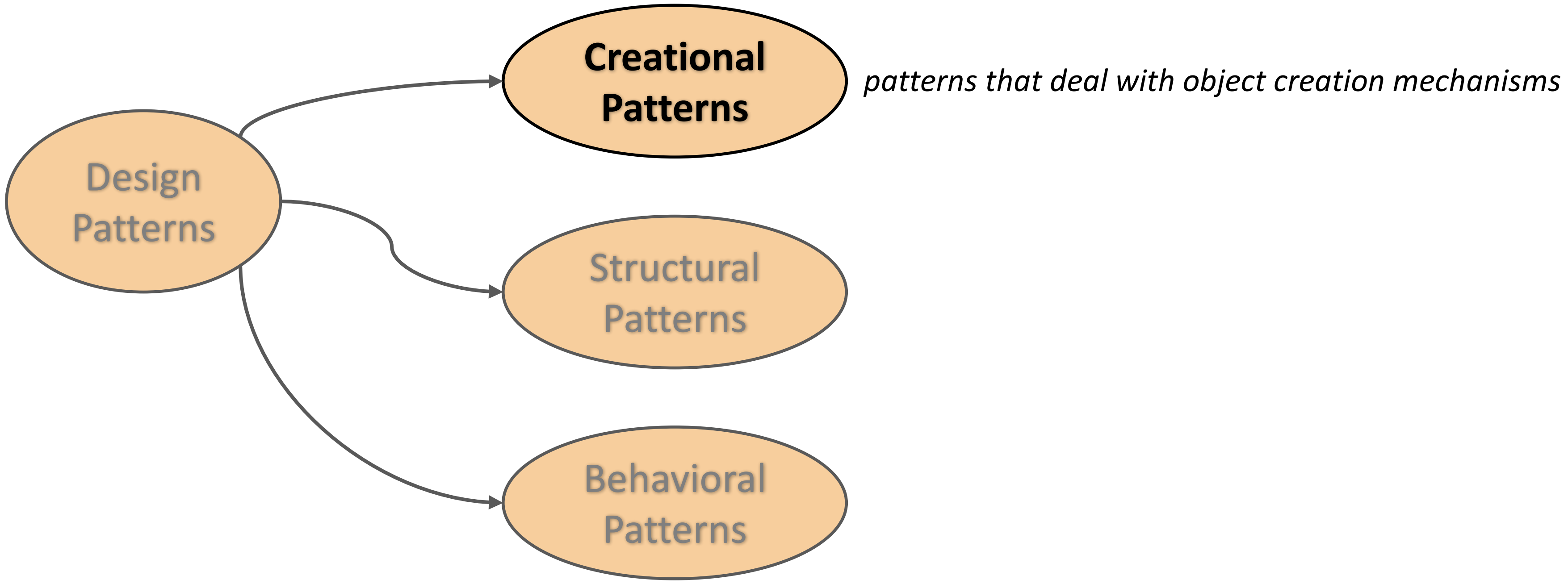
❑ Speed up the development process by providing tested, proven development paradigms

❑ Improves code readability for coders and architects familiar with the patterns.

TYPES OF DESIGN PATTERNS



CREATIONAL PATTERNS



Prototype

Fully initialized instance to be copied or cloned.

Type of objects to create is determined by a prototypical instance.

- Used when new() operator is harmful
- One instance of a class for use as a breeder of all future instances.

Builder

Builds a complex object using simple objects

Follows step by step approach

Separates Object construction from representation

Prevents inconsistency during object creation.

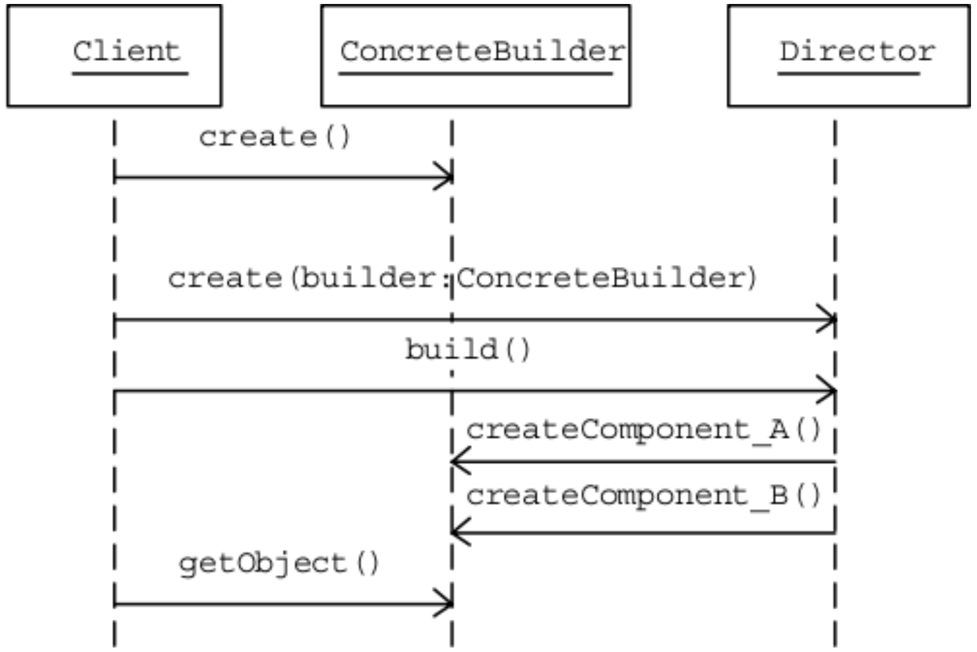
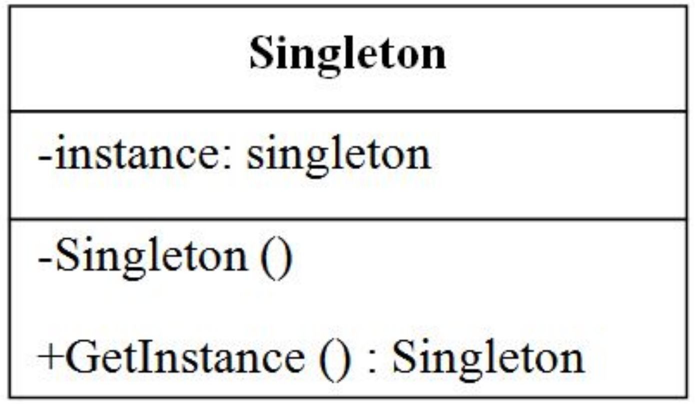


Image Source: http://rezagh.wikidot.com/local--files/builder-pattern/builder_seq.png

Singleton

Only one instance of an object is allowed on JVM



Singleton
-instance: singleton
-Singleton () +GetInstance () : Singleton

Examples

- System class in Java is singleton
- Spring has beans that are singleton

Features

- Private Constructors
- Lazy Implementation of Singleton

Image Source: http://csharpcorner.mindcrackerinc.netdna-cdn.com/UploadFile/bd5be5/design-patterns-in-net/Images/Design-Patterns-8.jpg

Factory

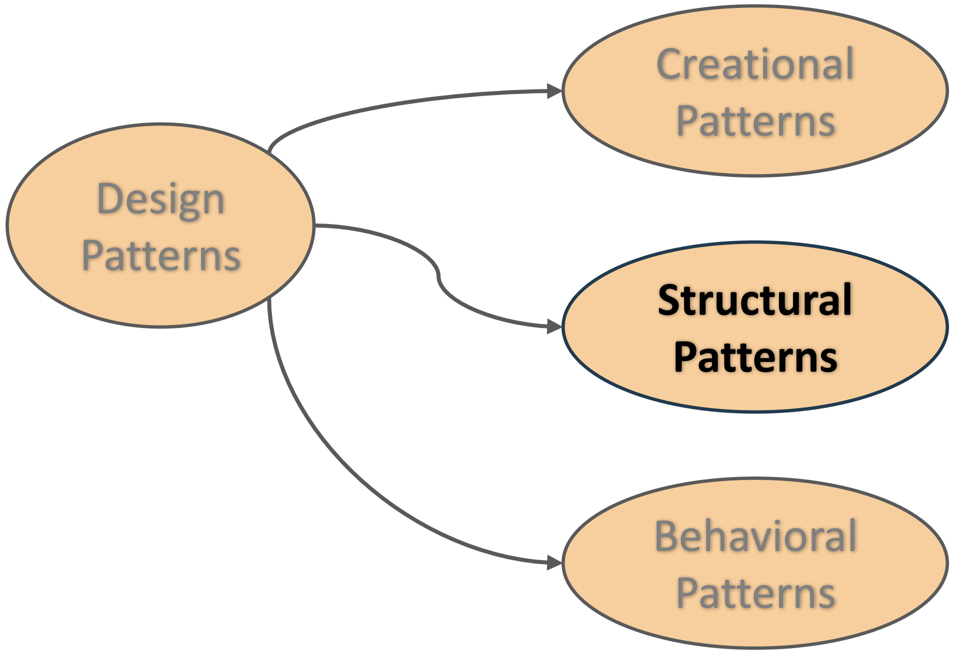
Create a family of classes at once

Handles construction of Family members

Takes out the responsibility of instantiation of a class from client program to the factory class.

```
public class ComputerFactory {
    public static Computer getComputer( String type, String ram, String hdd, String cpu) {
        if ("PC".equalsIgnoreCase(type))
            return new PC (ram, hdd, cpu);
        else if("Server".equalsIgnoreCase(type))
            return new Server (ram, hdd, cpu);
        return null;
    }
}
```

STRUCTURAL PATTERNS

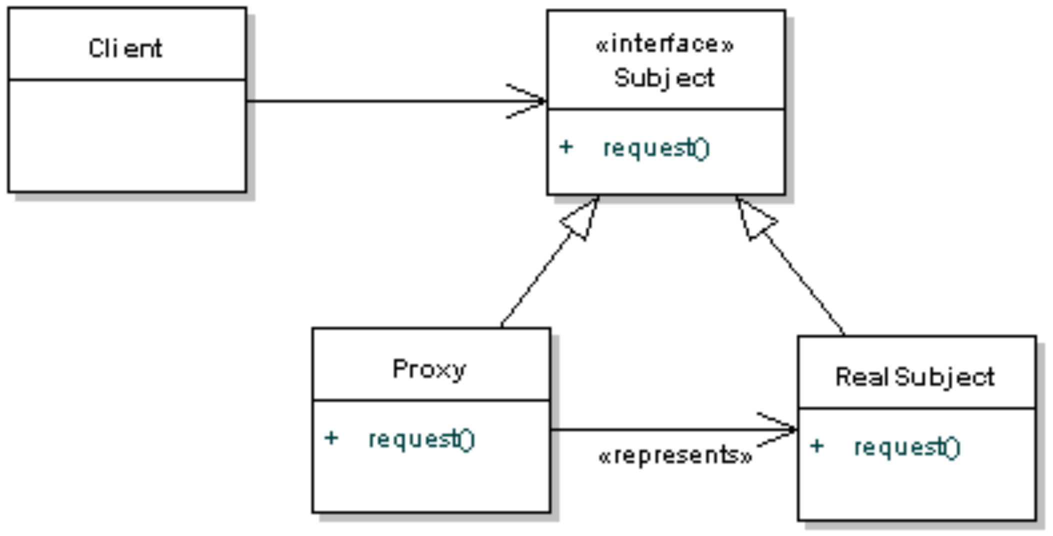


simple ways to realize relationships between entities

Proxy

An object representing another object.

Used to form large object structures across many disparate objects



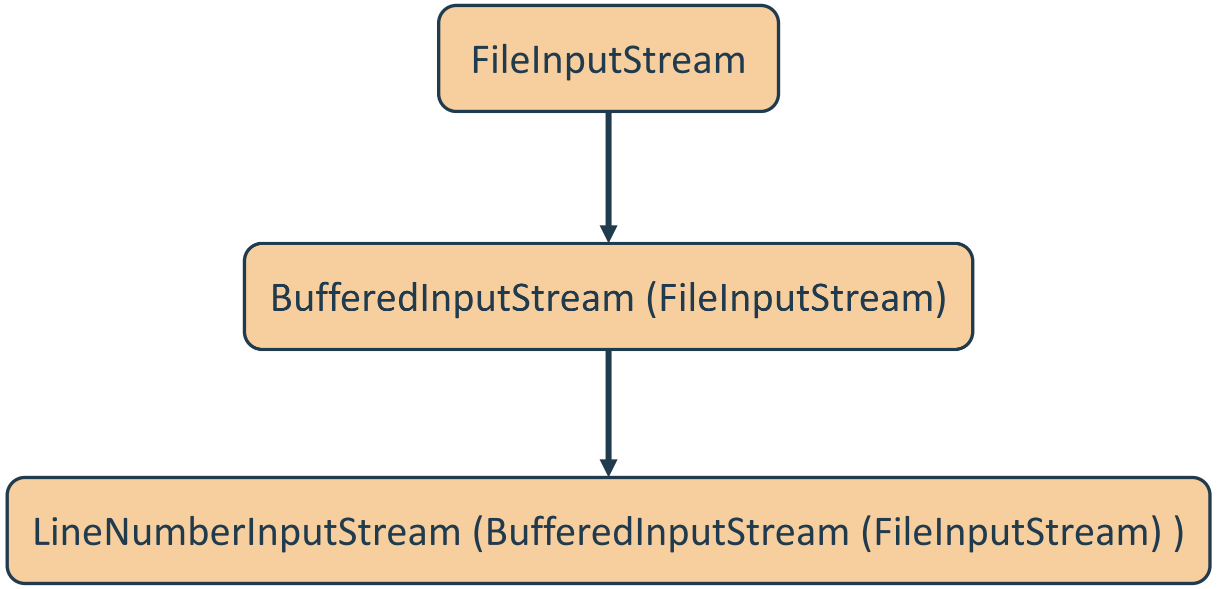
- Credit card as a proxy for bank account

Image Source: https://dzone.com/articles/design-patterns-proxy

Decorator

Also know as wrapper

Allows behavior to be added to an individual object, either statically or dynamically, without affecting the behavior of other objects from the same class.


FileInputStream
BufferedInputStream (FileInputStream)
LineNumberInputStream (BufferedInputStream (FileInputStream))

Facade

Provides a simplified interface to a larger body of code, such as a class library.

Single Class that represents the entire subsystem

Benefits

- Less Coupling
- Subsystem can be changed independently
- Reduced Network Calls
- Helps in establishing transaction boundary

Adapter

Used so that two unrelated interfaces can work together

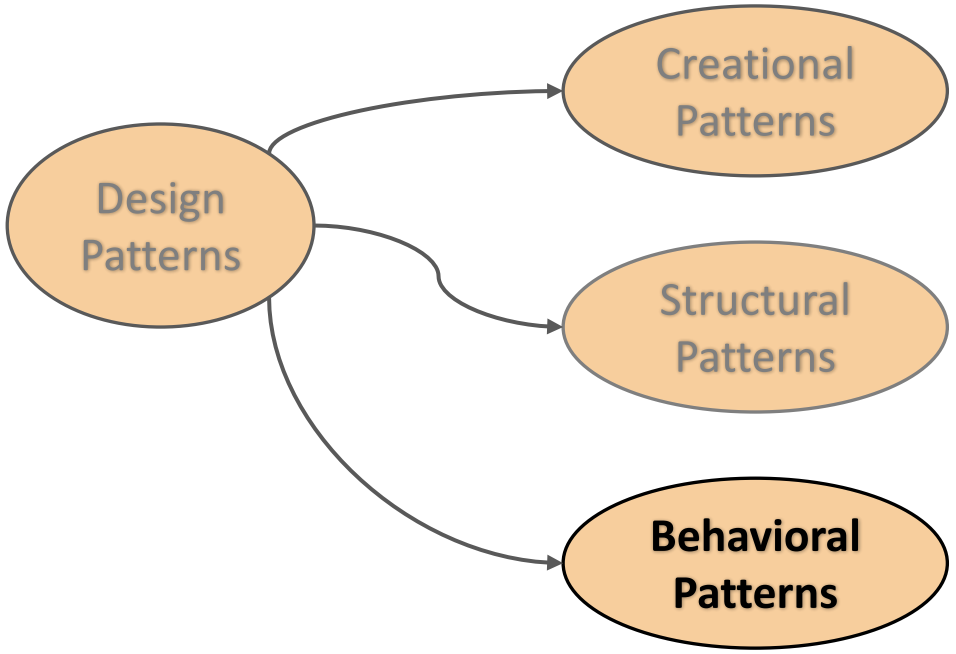
The object that joins these unrelated interface is called an **Adapter**

Examples

- Power Adapter in mobile chargers
- Channel Message to UM

Image Source: http://ecx.images-amazon.com/images/I/71HU5LWrwbL._SY355_.jpg

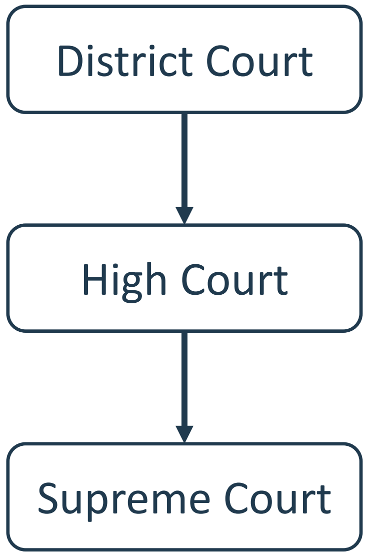
BEHAVIORAL PATTERNS



communication patterns between objects

Chain of Responsibility

A way of passing a requirement between chain of objects.



Examples

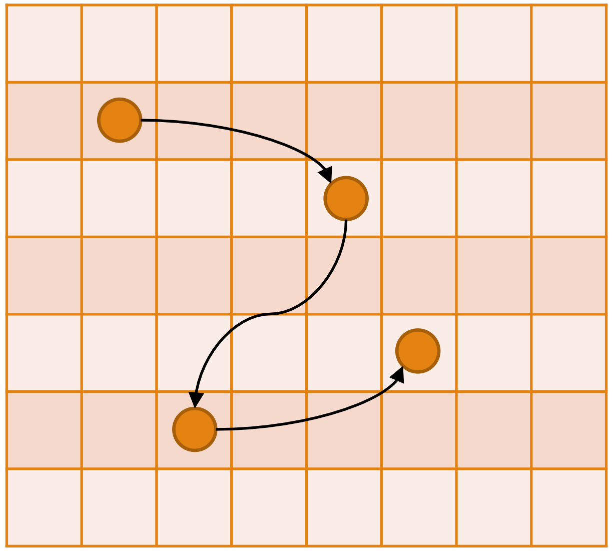
- Exception Handling in Java
- Logger Pattern in Handlers

Benefits

- Less Coupling

Iterator

Sequentially access the elements of a collection



Examples

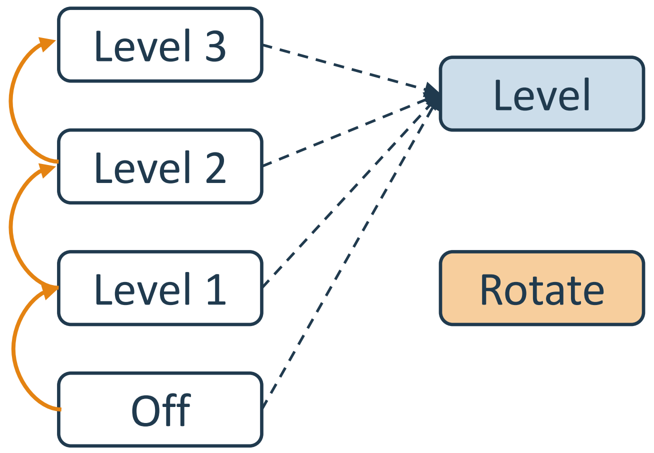
- Iterator in Java

Benefits

- Abstracts out the underlying type of collection
- hasNext() will give next element no matter how the collection is implemented on memory.
- Collection could be Array, ArrayList or HashMap

State

Alters the behavior of an Object when its stage changes



Examples

- Fan Controller

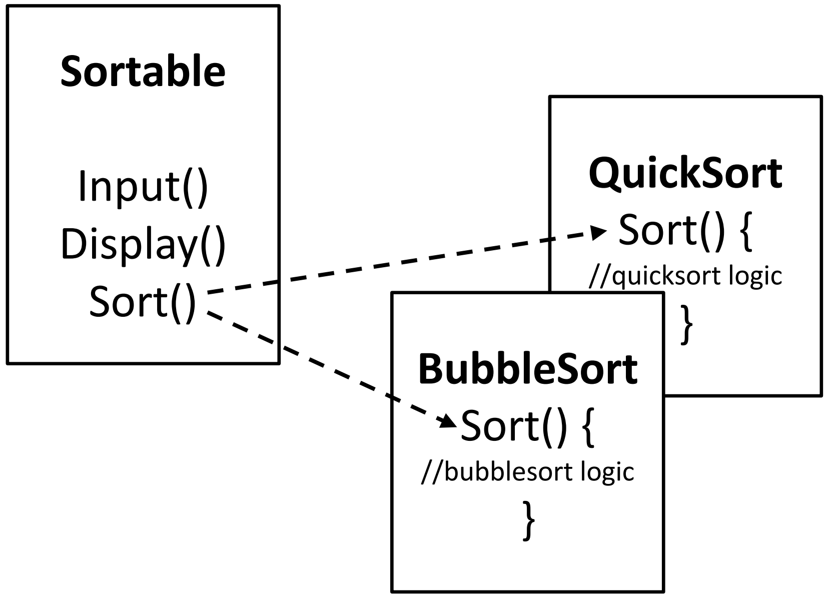
Benefits

- No if-else conditions required
- More modular

Strategy

Encapsulated an algorithm inside a class.

An interface with common procedures for a family of algorithm



Examples

- compare method in Java.util.Comparator class

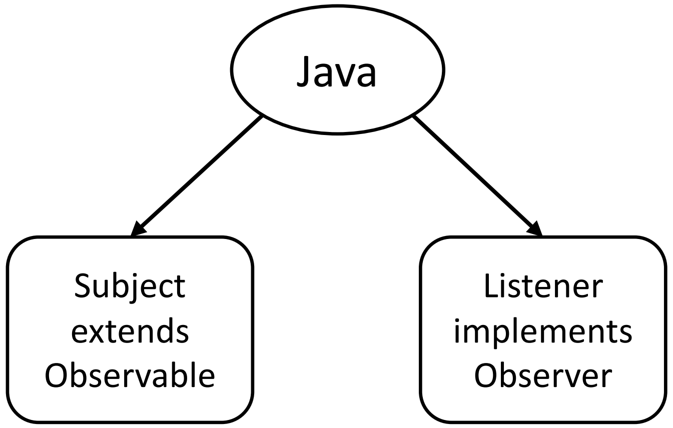
Benefits

- Each algorithm can implement or change in its own way

Observer

A way of notifying change to a number of classes

Implemented by default in Java 7.

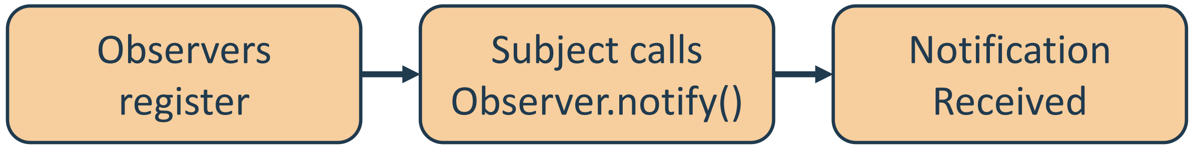


Examples

- Online Bidding

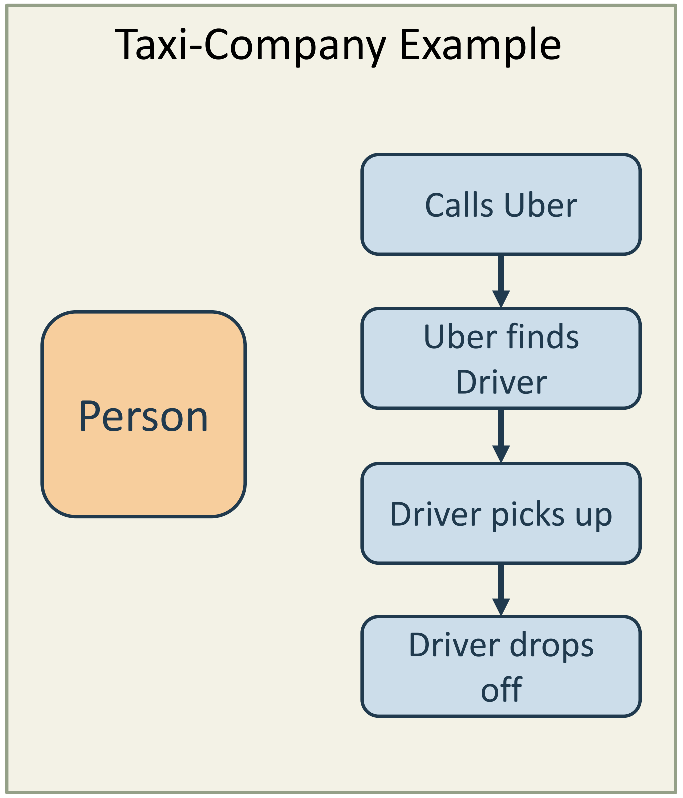
Cons

- Subject class cannot extend any other class



Visitor

Defines a new operation on a class without change in it.



About Example

- Driver is a operator
- While driving, Person class has no control.

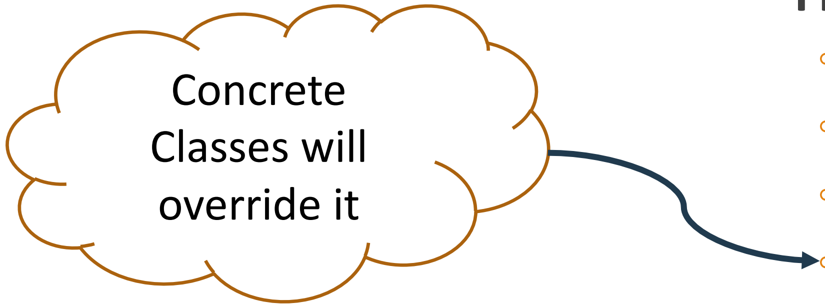
Points

- Can be done with Inheritance as well
- But it will complicate the structure

Template Method

Consist of templates which leaves the exact implementation of methods to subclasses

Taxi-Company Example



FileHandler Class

- openFile(String filepath)
- closeFile(File file)
- writeToFile(File file, String msg)
- performOperation(File file) ← abstract method

Command

Encapsulates a command request as an object

All clients of Command objects treat each object as a "black box”.

They simply invoke object's virtual execute() method.

- Java.lang.Runnable → you implement only the run() method
- Elastic Search Request Object

sprinklr