## Structural Design Patterns

## Overview

* Provides different ways to create a complex object or class structure
* It provides ways to realize relationships between entities
* Structural patterns define how each component or entity should be structured to have very flexible interconnecting modules which can work together in a larger system.

## Adapter

* Used to interface 2 or more unrelated / incompatible objects
* Examples in JDK
  + java.util.Arrays#asList()
  + java.io.InputStreamReader(InputStream) (returns a Reader)
  + java.io.OutputStreamWriter(OutputStream) (returns a Writer)

## Composite

* Used to create tree structure,
* Example: Drawing consist of using several shapes. Shapes can be created and added. Color can be passed

### Proxy

* Used to delegate certain access to another object
* Used to provide controlled access

### Flywheet

* Used when we need to create lot of objects for a class in an optimized way (less memory usage)
* To apply flyweight pattern, we need to divide Object property into **intrinsic** and **extrinsic** properties
* We need to create a Factory of objects

### Facade

* Used for client applications to easily access service
* Suppose if we want to get HTML, PDF reports dynamically from the server, we can use this pattern
* Used to provide a unified interface to a set of interfaces in a subsystem
* Used to create Helper classes

### Bridge

* Used to decouple interface from implementation
* Publish interface in an inheritance hierarchy, and bury implementation
* Used to hide implementation details from client

### Decorator

* Used to modify the functionality of an object at run time with out affecting the other instances of the same class