JDK lib used pattern:

<https://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns-in-javas-core-libraries>

Summary

<http://www.cs.unc.edu/~stotts/COMP723-s15/patterns/sum.html>

<https://medium.com/kkempin/short-summary-of-design-patterns-part-i-creational-patterns-c9304d9237ec>

<https://medium.com/kkempin/short-summary-of-design-patterns-part-ii-structural-patterns-350a8ec49006>

<https://medium.com/kkempin/short-summary-of-design-patterns-part-iii-behavioral-patterns-571e062acc7a>

references

BY : [Derek Banas](https://www.youtube.com/channel/UCwRXb5dUK4cvsHbx-rGzSgw) (good one)

<https://www.youtube.com/watch?v=qG286LQM6BU>

**1. program to an interface not an implementation: Strategy behavioral design pattern (DP). client depends on interface not on implementation.**

**2. Cleanly seprate interface and implementation : internal(under control and changable) and published interface .**

3. Favor object composition over inheritance:

**use interface to change behavior statically at compile time (Animal is a dog etc),.**

**use composition to change behavior dynamically at run time (they use delegation principle).**

**4. Changing behavior – making a system independent of changing requirement.**

**runtime flexibility via composition using strategy ,state,decorator ,proxy design patterns.**

**Compile time flexibility via inheritance : template design pattern.**

**5. Changing object creation – making a system independent of how its objects are created.**

**runtime flexibility via composition using abstract factory ,builder ,prototype design patterns.**

**Compile time flexibility via inheritance : factory Method design pattern.**

**Singleton design pattern.**

**6. Object structure : Working with complex object structures efficiently.**

**Composite ,Iterator , visitor, Flyweight**

**7. Object collaboration : avoid tight coupling between interacting objects.**

**Observer ,method ,chain of responsibility ,command design pattern.**

**8. changing interfaces independently: Adapter, Bridge, facade design pattern.**

**9. storing and restoring object state : Memento design pattern.**

**10. Interpreter/domain specific languages : interpreter design pattern.**

**Design pattern**

1. **Creational :** These design patterns are all about class instantiation or object creation. These patterns can be further categorized into Class-creational patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.
2. Singleton
3. Prototype
4. Factory Method
5. Abstract Factory
6. Builder
7. Object Pool
8. **Structural Design pattern :** These design patterns are about organizing different classes and objects to form larger structures and provide new functionality.
9. Facade Design pattern:
10. Adapter
11. Bridge
12. Composite
13. Decorator
14. Flyweight
15. Private Class Data
16. Proxy.
17. **Behavioral:** Behavioral patterns are about identifying common communication patterns between objects and realize these patterns.
18. Chain of responsibility
19. Command
20. Interpreter
21. Iterator
22. Mediator
23. Memento
24. Null Object
25. Observer
26. State
27. Strategy
28. Template method
29. Visitor

**Façade Design Pattern** : The diagram definition of the Facade pattern is quite simple - all you're really doing is insulating client from the subsystem.

Package 1 + package 2 functionality into  package 3.

References

<https://www.oodesign.com/> (good link for connecting other pattern relation)

<https://www.geeksforgeeks.org/design-patterns-set-1-introduction/>

<https://sourcemaking.com/design_patterns>

https://www.geeksforgeeks.org/object-pool-design-pattern/

#### 5. What is the difference between factory and abstract factory design pattern?

**Answer:**  
Both factory and abstract factories are creational design patterns. The major difference between these two is, a factory pattern creates an object through inheritance and produces only one Product. On the other hand, an abstract factory pattern creates the object through composition and produce families of products.