Purpose: Design patterns are solutions to general problems that software devs faced during software development.

1. Type of design pattern:  
    Three categories: Creational, Structural, Behavioral pattern.

* Creational patterns: these design patterns provide a way to create objects while hiding creation logic, rather than instantiating objects directly using new operator.
* Structural patterns: these design patterns concern class and object composition.
* Behavioral patterns: these design pattern are specifically concerned with communication between objects.

1. Creational patterns.

**Abstract Factory pattern**: An interface responsible for creating factory of related object. Each generated factory can give the object as per Factory pattern.

**Singleton pattern**: Create an object while making sure that only single object gets created. This class provides a way to access.

**Builder pattern**: Separate the construction of a complex object from its representation so that the same construction process can create different representations (Tách rời quá trình tạo object với nội dung và cấu trúc bên trong của nó, nhờ vậy một object có thể có nhiều cách tạo – nhiều biểu hiện khác nhau)

Instantly Note:

**Strategy pattern** (behavioral pattern) gần tương tự ý nghĩa với Builder pattern(creational pattern). Strategy pattern tạo behavior thay đổi theo từng logic của từng class.

Buider pattern tạo Oject nhưng có những default (hơi khó giải thích nhưng builder giúp giảm bớt quá trình tạo class khi có quá nhiều trường hợp muốn dung default value)

**Strategy pattern**: Nhiều class con kế thừa class cha nhưng khác behavior của class cha -> behavior class dung interface. Trong interface có nhiều interface hành động khác. Ai dùng cái nào thì kế thừa cái đó. -> flexible system. – Nói chung có quá nhiều behavior -> chia thành interface, muốn dùng cái nào, implement cái đó (reuse behavior code.)

Builder pattern: Class có những optional field. Lúc muốn nhập, lúc muốn dùng default value -> dùng builder pattern. – Có quá nhiều value mà làm biếng nhập.

**Singleton**: Một lớp chỉ có duy nhất 1 thể hiện và một biến toàn cục để truy cập nó. Không dùng toán tử new.

**Rắc rối**: trong trường hợp đa luồng gọi đến biến toàn cục singleton (Đoạn mã kiểm tra sự tồn tại) -> dùng synchronized (đồng bộ hóa) A xong B mới vào.

-Builder pattern: <https://www.journaldev.com/1425/builder-design-pattern-in-java>

-DAO design pattern.