**FACTORY PATTERN** (Important)

Explanation: <https://www.youtube.com/watch?v=7g9S371qzwM&list=PL6W8uoQQ2c61X_9e6Net0WdYZidm7zooW&index=7>

The Factory Pattern in Java is like ordering food from a menu at a restaurant. You tell the waiter what you want, and the kitchen prepares it for you.

In programming, it's used to create objects without exposing the instantiation logic to the client. Instead of directly calling the constructor of a class, you use a factory method to create objects based on certain conditions or parameters.

Use when:

* **Encapsulate Object Creation**: Use the Factory Pattern when you need to encapsulate the object creation process, hiding the details of instantiation from the client code.
* **Flexibility in Object Creation**: It's useful when you want to provide flexibility in creating different types of objects based on certain conditions or parameters.
* **Decouple Client and Concrete Classes**: Helps decouple client code from concrete classes, allowing for easier maintenance and future modifications.
* **Create Objects Dynamically**: Use it when you need to create objects dynamically at runtime, based on runtime conditions or configurations.
* **Centralized Object Creation Logic**: When you have complex object creation logic that you want to centralize in one place, the Factory Pattern can be a good choice.

**ABSTRACT FACTORY PATTERN**

Factory of factory.

The Abstract Factory Pattern in Java is like visiting a restaurant where you can order a set menu. You choose a menu, and then you receive a collection of dishes that belong to that menu.

In programming, it's used to create families of related or dependent objects without specifying their concrete classes. Instead of dealing with individual objects directly, you work with a factory that produces families of objects.