Factory pattern :

* Optimal way to create objects

Intent :

* Define an interface to create an object and let the subclass decide which factory class to instantiate.

Solution :

* Mainly solves the problem of interface selection
* Creation process is performed in subclasses

Application Example:

Hibernate only need the dialect and driver to change the database.

**Advantages**: 1. If a caller wants to create an object, it only needs to know its name. 2. High scalability. If you want to add a product, you only need to extend a factory class. 3. The specific implementation of the shielding product is shielded, and the caller only cares about the interface of the product.

**Disadvantage**: Every time a product is added, a specific class and object implementation factory need to be added, which doubles the number of classes in the system, increases the complexity of the system to a certain extent, and also increases the specific class of the system. rely. This is not a good thing.

**Note**: The factory method pattern can be used anywhere complex objects need to be generated. Simple objects, especially objects that can be created by just using new ones, do not need to use the factory pattern. If you use the factory pattern, you need to introduce a factory class, which will increase the complexity of the system.

* [java.util.Collections](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/Collections.html) — singletonList(), singletonMap()
* [java.util.ResourceBundle](https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/util/ResourceBundle.html) — getBundle()