

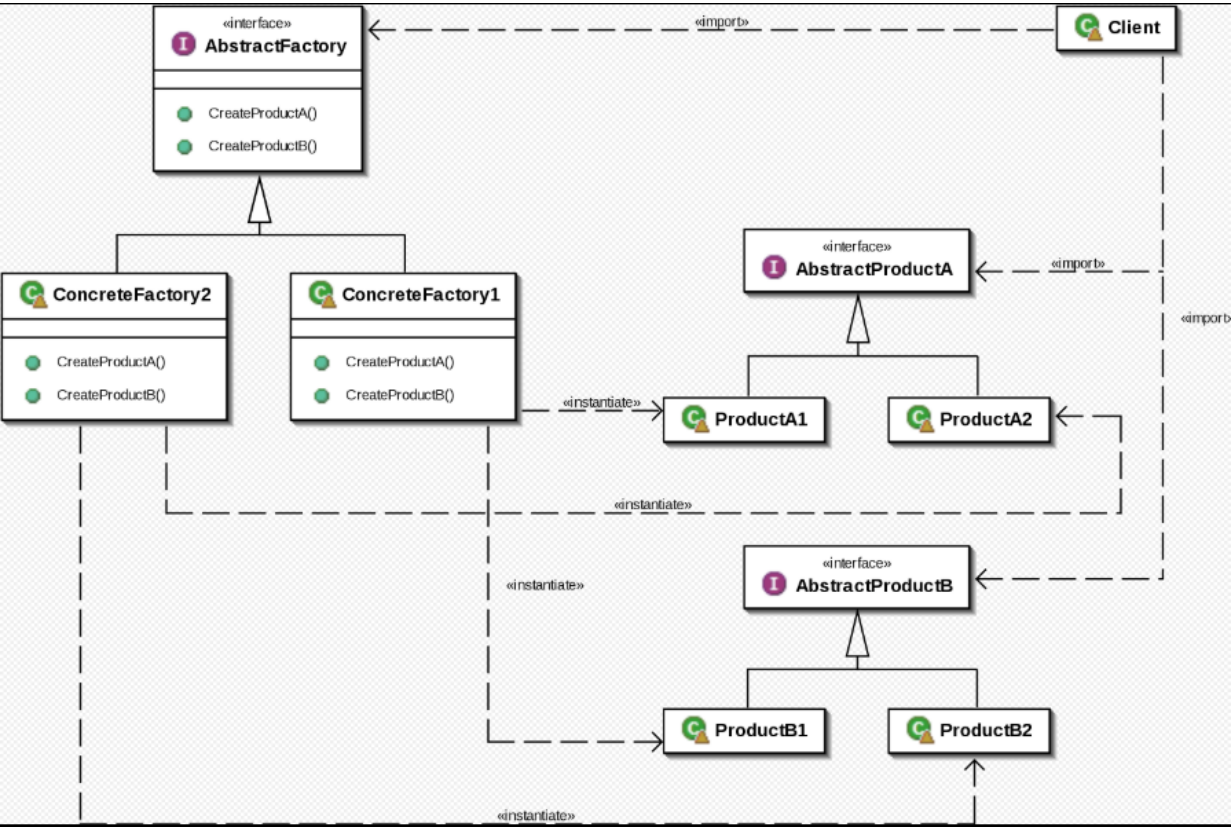
抽象工厂模式

Abstract Factory

郭嘉

模式定义：

提供一个创建一系列相关或互相依赖对象的接口，而无需指定它们具体的类



代码示例:

```
1  package com.tuling.designpattern.abstractfactory;
2
3  /**
4   * @author 腾讯课堂-图灵学院 郭嘉
5   * @Slogan 致敬大师，致敬未来的你
6   */
7  public class AbstractFactoryTest {
8      public static void main(String[] args) {
9          IDatabaseUtils iDatabaseUtils=new OracleDataBaseUtils(); //
10         IConnection connection=iDatabaseUtils.getConnection();
11         connection.connect();
12         ICommand command=iDatabaseUtils.getCommand();
13         command.command();
14
15
16     }
17 }
18
19 // 变化: mysql , oracle. ...
20 // connection , command ,
21
22 interface IConnection{
23     void connect();
24 }
25 interface ICommand{
26     void command();
27 }
28 interface IDatabaseUtils{
29     IConnection getConnection();
30     ICommand getCommand();
31 }
32 class MySqlConnection implements IConnection{
33
34     @Override
35     public void connect() {
36         System.out.println("mysql connected.");
37     }
38 }
```

```
39 class OracleConnection implements IConnection{
40
41     @Override
42     public void connect() {
43         System.out.println("oracle connected.");
44     }
45 }
46
47
48 class MysqlCommand implements ICommand{
49
50     @Override
51     public void command() {
52         System.out.println(" mysql command. ");
53     }
54 }
55
56 class OracleCommand implements ICommand{
57
58     @Override
59     public void command() {
60         System.out.println("oracle command.");
61     }
62 }
63
64 class MysqlDataBaseUtils implements IDatabaseUtils{
65
66     @Override
67     public IConnection getConnection() {
68         return new MysqlConnection();
69     }
70
71     @Override
72     public ICommand getCommand() {
73         return new MysqlCommand();
74     }
75 }
76
77 class OracleDataBaseUtils implements IDatabaseUtils{
78
79     @Override
```

```
80 public IConnection getConnection() {
81     return new OracleConnection();
82 }
83
84 @Override
85 public ICommand getCommand() {
86     return new OracleCommand();
87 }
88 }
89
```

应用场景：

程序需要处理不同系列的相关产品，但是您不希望它依赖于这些产品的具体类时，
可以使用抽象工厂

优点：

1. 可以确信你从工厂得到的产品彼此是兼容的。
2. 可以避免具体产品和客户端代码之间的紧密耦合。
3. 符合单一职责原则
4. 符合开闭原则

JDK源码中的应用:

```
1 java.sql.Connection
2 java.sql.Driver
```

