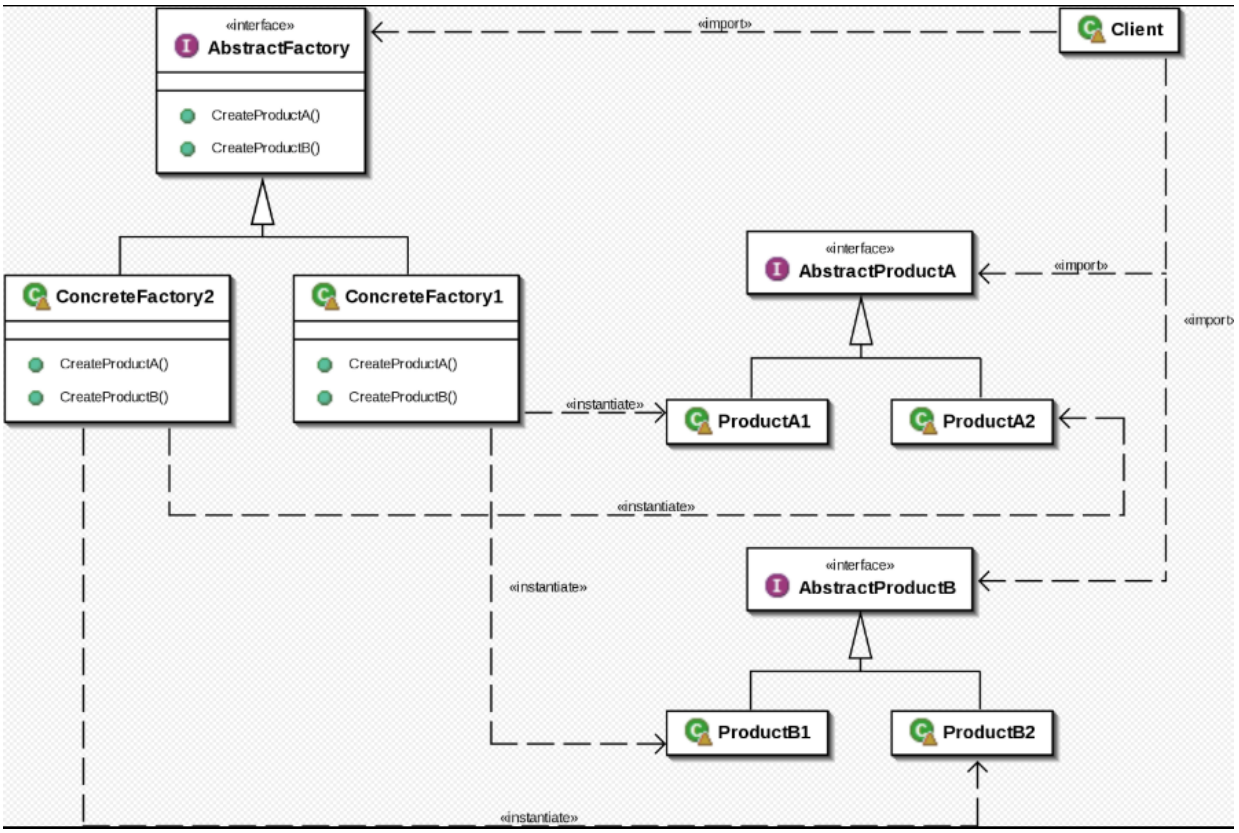
抽象工厂模式

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：

```
package com.tuling.designpattern.abstractfactory;

/**
 * @author 腾讯课堂-图灵学院 郭嘉
 * @Slogan 致敬大师，致敬未来的你
 */
public class AbstractFactoryTest {
 public static void main(String[] args) {
 IDatabaseUtils iDatabaseUtils=new OracleDataBaseUtils(); //
  IConnection connection=iDatabaseUtils.getConnection();
  connection.connect();
  ICommand command=iDatabaseUtils.getCommand();
  command.command();


  }
}

// 变化: mysql , oracle. ...
// connection , command ,

interface IConnection{
 void connect();
}
interface ICommand{
 void command();
}
interface IDatabaseUtils{
 IConnection getConnection();
 ICommand getCommand();
}
class MysqlConnection implements IConnection{

 @Override
 public void connect() {
 System.out.println("mysql connected.");
  }
}
```

```
class OracleConnection implements IConnection{

 @Override
 public void connect() {
 System.out.println("oracle connected.");
 }
}


class MysqlCommand implements ICommand{

 @Override
 public void command() {
 System.out.println(" mysql command. ");
 }
}

class OracleCommand implements ICommand{

 @Override
 public void command() {
 System.out.println("oracle command.");
 }
}

class MysqlDataBaseUtils implements IDatabaseUtils{

 @Override
 public IConnection getConnection() {
 return new MysqlConnection();
 }

 @Override
 public ICommand getCommand() {
 return new MysqlCommand();
 }
}

class OracleDataBaseUtils implements IDatabaseUtils{

 @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
```