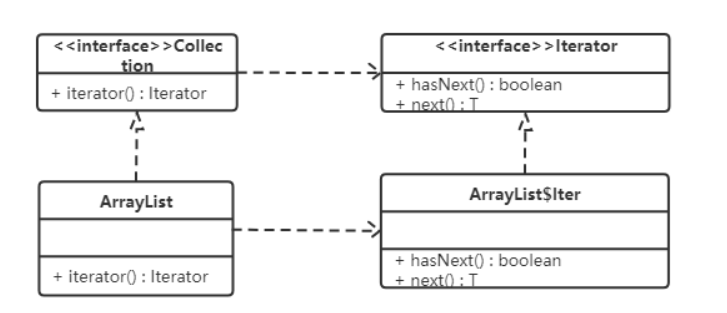
工厂方法（Factory Method）模式，是对简单工厂模式进行了抽象、推广。  
 由于使用了多态性，工厂方法模式保持了简单工厂模式的优点，而且克服了缺点。通过设计一个抽象的Factory类或接口，  
 这个类将不再负责具体的产品生产，而是只制定一些规范，具体的生产工作，推延到其子类去完成。  
  
意图  
 定义一个用户创建对象的接口，让子类决定实例化哪一个类，工厂方法模式，使一个类的实例化，延迟到其子类  
  
  
角色及其职责  
（1）抽象工厂（Creator）角色：  
是工厂方法模式的核心，与应用程序无关。任何在模式中创建的对象的工厂类，必须实现这个接口。  
（2）具体工厂（Concrete Creator）角色：  
是实现抽象工厂接口的具体工厂类，包含与应用程序密切相关的逻辑，并且受到应用程序调用以创建产品对象。  
（3）抽象产品（Product）角色：  
工厂方法模式所创建的对象的超类型，也就是产品对象的共同父类或共同拥有的接口。  
（4）具体产品（Concrete Product）角色：  
实现了抽象产品角色所定义的接口。某具体产品有专门的具体工厂创建，之间往往一一对应。  
  
  
优点:  
（1）用工厂方法在一个类的内部创建对象，通常比直接创建对象更灵活。  
（2）实现了开闭原则可以在不改变工厂的前提下，增加新产品。  
（3）工厂方法模式通过面向对象的手法，将所要创建的具体对象的创建工作，延迟到了子类，  
 从而提供了一种扩展的策略，较好的解决了这种紧耦合的关系  
  
  
jdk源码中的应用:  
1. Collection.iterator 方法



[Collection](https://so.csdn.net/so/search?q=Collection&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/chengqiuming/article/details/_blank)接口是抽象工厂类，ArrayList是具体的工厂类。

[Iterator](https://so.csdn.net/so/search?q=Iterator&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/chengqiuming/article/details/_blank)接口是抽象商品类，ArrayList类中的Iter内部类是具体的商品类。

具体的工厂类中iterator()方法创建具体的商品类的对象。

Collection可以看作是一个总的抽象工厂，它的一些实现这个接口的类，像ArrayList，LinkedHashSet等等可以看作一个个不同的品牌的工厂，而总的产品Iterator接口里面会定义产品所需功能的细节，然后在交给各个品牌不同的工厂来实现。