# 静态工厂方法与构造器不同的优势在于：

## 静态工厂方法有名称：

如果构造器的参数本身没有确切地描述正被返回的对象，那么具有适当名称的静态工厂会更容易使用。

**例如：**

public static BigInteger probablePrime(int bitLength, Random rnd)

## 可以在每次调用他们的时候复用已存在的对象

这使得不可变类可以使用预先构造好的实例，或者将构建好的实例缓存起来，进行重复利用，从而避免创建不必要的重复对象。

**例如：**

public static final Boolean *TRUE* = new Boolean(true);  
public static final Boolean *FALSE* = new Boolean(false);

public static Boolean valueOf(boolean b) {  
 return (b ? *TRUE* : *FALSE*);  
}

## 帮助单例模式实现延迟实例化：

// volatile关键字保证：当uniqueInstance遍历被初始化成Singleton实例时，多个线程正确地处理uniqueInstance变量  
 // 保证多个线程操作uniqueInstance变量的原子性  
 private volatile static Singleton uniqueInstance;  
  
private Singleton() {}  
  
 */\*\*  
 \* 双重检查实现单例模式  
 \** ***@return*** *\*/* public static Singleton getInstance(){  
 if(uniqueInstance == null){  
 // 第一次实例化的时候才需要加锁  
 synchronized (Singleton.class) {  
 if (uniqueInstance == null) {  
 uniqueInstance = new Singleton();  
 }  
 }  
 }  
 return uniqueInstance;  
 }

## 4.可以返回原返回类型的任何子类型的对象

**例如：**

1.Arrays.java:

返回的ArrayList为内部类

public static <T> List<T> asList(T... a) {  
 return new ArrayList<>(a);  
}

private static class ArrayList<E> extends AbstractList<E>  
 implements RandomAccess, java.io.Serializable

2.java.util.EnumSet没有公有构造器，只有静态工厂方法。他们返回两种实现类之一，具体则取决于底层枚举类型的大小：如果元素小于等于64个，返回一个RegalarEnumSet实例，用单个long进行支持。如果元素大于65个，返回JumboEnumSet实例，用long数组进行支持。 客户端永远不知道也不关心他们从工厂方法中得到的对象的类；他们值关系他是EnumSet的某个子类即可。

**为了提升软件的可维护性和性能，返回对象的类也可能随着发行版的不同而不同。**

静态工厂方法的一些惯用名称：

valueOf、of、newInstance、getInstance、getType(Type表示工厂方法返回的对象类型)…

参考：

《Effective java》