# Effective JAVA 78

## Part1 创建和销毁对象

### 考虑用静态工厂方法代替构造器

* 静态工厂方法会有名称，不同类别实例可以加以区分，不再使用参数判断不同的类别情况；
* 不必每次调用就必须创建一个新的实例，静态工厂的单例模式就是这种方法，可以拓展成控制实例个数；
* 可以返回类型的任何子类型的对象，对外隐藏具体使用的是哪个子类，服务提供者框架即是；
  + - 完全交给静态工厂，则不能被继承，所以最好使用在具体的YoungClass中；
    - 于静态方法没有区别，反查实例是如何实例化会非常困难；
    - 静态工厂惯用名称有：valueOf/of/getInstance/newInstance/getType/newType.各个名称习惯用的场景略有不同，getInstance一般是使用于单例模式，type一般用于工厂模式。

### 遇到多个构造器参数时要考虑用构建器

* 具名的可选参数，对于多个参数时，重叠构造器牵制于构造器，很多时候不能根据需要只设定特定的属性，而JavaBeans模式通过setter设置，线程不安全且可能出现属性无效的问题且无法判断，Builder的特点就是可以实现连环套设置，如：People p = new People.Builder().name(“HL”).sex(“女”).age(“23”).build();具体代码例子见书中详情。
* 构建实例，需要构建静态构建器，需要额外的开销，会对性能有影响。

### 用私有构造器或者枚举类型强化Singleton属性

### 通过私有构造器强化不可实例化的能力

### 避免创建不必要的对象

### 消除过期的对象引用

### 避免使用终结方法

## Part2 对象通用方法

### 覆盖equals时需要遵守通用约定

### 覆盖equals时总要覆盖hashCode

### 始终要覆盖toString

### 谨慎地覆盖clone

### 考虑实现Comparable接口

## Part3 类和接口

### 使类和成员的可访问性最小化

### 在公有类中使用访问 方法而非公有域

### 使可变化性最小化

### 复合优先于继承

### 要么继承而设计，并提供文档，否则禁止继承

### 接口优于抽象类

### 接口只用于定义类型

### 类层次优于标签类

### 用函数对象表示策略

### 优先考虑静态成员类

## Part4 泛型

### 不要在新代码中使用原生态类型

### 消除非受检警告

### 列表优先于数组

### 优先考虑泛型

### 优先考虑泛型方法

### 利用有限制通配符来提升API的灵活性

### 优先考虑类型安全的异构器

## Part5 枚举和注解

### 用enum代替int常量

### 用实例域代替序数

### 用enumSet代替位域

### 用EnumMap代替序数索引

### 用接口模拟可伸缩的枚举

### 注解优先于命名模式

### 坚持使用Override注解

### 用标记接口定义类型

## Part6 方法

### 检查参数的有效性

### 必要时进行保护性拷贝

### 谨慎设计方法签名

### 慎用重载

### 慎用可变参数

### 返回零长度的数组或者集合，而不是null

### 为所有导出的API元素编写文档注释

## Part7 通用程序设计

### 将局部变量的作用域最小化

### For-each循环优于传统的for循环

### 了解和使用类库

### 如果需要精确答案，请避免使用浮点数

### 基本类型优于装箱基本类型

### 如果其他类型更适合，则尽量避免使用字符串

### 当心字符串连接的性能

### 通过接口引用对象

### 接口优先于反射机制

### 谨慎地使用本地方法

### 谨慎地进行优化

### 遵守普遍接受的命名惯例

## Part8 异常

### 只针对异常的情况才使用异常

### 对可恢复的情况使用受检异常，对编程错误使用运行异常

### 避免不必要地使用受检异常

### 优先使用标准的异常

### 抛出与抽象相对应的异常

### 每个方法抛出异常都要有文档

### 在细节消息中包含能捕获失败的信息

### 努力使失败保持原子性

### 不要忽略异常

## Part9 并发

### 同步访问共享的可变数据

### 避免多度同步

### Executor和task优先于进程

### 并发工具优先于wait和notify

### 线程安全性的文档化

### 不要依赖于线程调度器

### 避免使用线程组

## Part10 序列化

### 谨慎地实现Serializable接口

### 考虑使用自定义的序列化形式

### 保护性地编写readObject方法

### 对于实例控制，枚举类型优先于readResolve

### 考虑用序列化代理代替序列化实例