Java Binary compatible

Binary-code compatibility（二进制兼容性），指升级系统时，不必重新编译使用这个库的可执行文件或使用这个库的其他库文件，程序的功能不被破坏，简单的来说即某个程序依赖其他库A，当A库升级时可以做到向后兼容，则程序功能不受影响。

下面是几种Java programming支持的binary compatible changes:

1. 为了提高性能，重新实现已有的方法，构造器和初始化器
2. 方法或者构造器添加返回值（可能原来是抛出异常），而不引起无限循环或者死锁
3. 在已有的类或接口中添加新fields,methods及constructors
4. 删除类中的私有fields,methos及contstructors
5. 当升级整个package时，删除包中package-only的access fields,methods、类的构造器及接口
6. 重新改变已有的type declaration中的fields,methods及constructors的顺序
7. 在类层次中将方法提升层次
8. 改变类或接口的superinterfaces的顺序
9. 在type hierarchy中插入新类或者interface types

# Evolution of Packages

A new top level class or interface type may be added to a package without breaking compatibility with pre-existing binaries, provided the new type does not reuse a name previously given to an unrelated type.

If a new type reuses a name previously given to an unrelated type, then a conflict may result, since binaries for both types could not be loaded by the same class loader.

Changes in top level class and interface types that are not public and that are not a superclass or superinterface, respectively, of a public type, affect only types within the package in which they are declared. Such types may be deleted or otherwise changed, even if incompatibilities are otherwise described here, provided that the affected binaries of that package are updated together.

# Evolution of Classes

下面描述了类、其成员及构造器的变化对已有binary的影响。

## abstract Classes

非Abstract Class变为Abstract Class，已有binaries在创建实例的时候在link time时会抛出InstantiationError，在Runtime时，反射方法会抛出InstantiationException。但是将Abstract Class变为非Abstract Class不会影响兼容性。

## final class

非final class变成final类型。在class loaded时可能会抛出VerifyError，因为final 类不能有子类。但是final class变成非final class没有影响。

## public class

非public class变成public，不会造成pre-existing binaries的兼容性。但是public class变成非public,可能在使用时抛出IllegalAccessError。

## Superclasses and Superinterfaces

改变超类或者接口，在加载的时候可能抛出ClassCircularityError。

## Class Type Parameters

增加或者删除class的type parmaters不会对binary compatibility有影响。

## Class Body and Member Declarations

在超类中增加同名的成员变量，不会对pre-existing binaries有影响。但是删除私有成员或者constructor，会抛出linkage error。

## Access to Members and Constructor

成员变量或者构造器的可见性降低会造成pre-existing binaries。

## Field Declarations

类中增加filed，可能会造成pre-existing binaries的不兼容问题。例如在子类中增加和父类中同名但是不同类型的方法，可能会造成linkage error，还有其他问题：

1. 新的field可见性降低
2. 新的field变成static

## field fields和constants

一个声明为非final的field变为final，会造成不可兼容的问题。抛出的异常为IllegalAccessError。

## 2.10static Fields

Field没有被声明为private，而且非static，但是改为static，或者返回来，会抛出IncompatibleClassChangeError。

## 2.11transient Fields

去除field的transient标识，会造成兼容性问题

## 2.12 Method和Constructor Declarations

新增方法或者Constructor声明，如果和superclass中的声明方法冲突，当出现新方法的可见性降低或者声明为static时，会出现linkage error异常。

参考链接：

<https://en.wikipedia.org/wiki/Binary-code_compatibility>

http://docs.oracle.com/javase/specs/jls/se7/html/jls-13.html