在JDK1.4中使用Java5的语言特性 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/461/2021_2022__E5_9C_A8JDK14_E4_c104_461632.htm Java 5中提供了一些语法上的特性，使用这些特性的类如果想要在以前版本的虚拟机上运行，会报UnsupportedClassVersionError错误。 原因是*.class文件中的主板本号如果是JDK5编译的话是50，而Jdk1.4编译的是48，所以会报这个不支持的类版本错误。怎么才能在JDK1.4中使用Java 5的特性呢？ JDK的开发人员早就意识到了这个问题，所以在javac中提供了参数，可以编译成Jdk1.4可以识别的版本。一般来说，只要在javac后面提供参数target jsr14即可。下面做一个例子。新建一个类Hello，代码如下：public class Hello{ } 虽然是个空类，但是足矣演示功能。 在Hello.java所在的文件夹调用javac Hello.java 用WinHex打开生成的Hello.class，可以看到在第一行第七列显示的是32，也就是主版本号50。说明这是Jdk5编译的class文件。 同样在Hello.java所在的文件夹调用javac -target jsr14 Hello.java 用WinHex打开生成的Hello.class，可以看到在第一行第七列显示的是30，也就是主版本号48。说明这是Jdk1.4编译的class文件。 知道怎么生成了，还需要知道这个参数的适用范围，javac并不是完全支持jdk5的所有特性的，下面按特性分别列举一下： 泛型和变长参数：编译器在泛型出现的地方插入的强制转换不依赖类库，所以能够在 Java 5 之前的 JVM 上很好地执行。类似的，编译器在出现变长参数列表的地方生成的代码也不依赖类库。 for-each 循环：当迭代数组时，编译器生成归纳变量和标准的数组迭代语法。当在Collection上迭代时，编译器生成标

准的基于迭代器的语法。当在非集合的Iterable 上迭代时，编译器生成错误。 自动装箱：编译器不生成对包装器类的valueOf()方法的调用，而是生成对构造函数的调用。 字符串连接：javac的JSR 14目标模式使编译器生成对StringBuffer的调用而不是对StringBuilder的调用。 枚举：javac JSR 14目标模式对枚举没有特殊支持。尝试使用枚举的代码会失败，在寻找 java.lang.Enum 基类时出现 NoClassDefFoundError。 下面讲一下可以编译成Jdk1.4版本的class的原理。 Java 5 中添加的语言特性 泛型、枚举、注释、自动装箱和增强的 for 循环 不需要修改 JVM 的指令集，几乎全部是在静态编译器（javac）和类库中实现的。当编译器遇到使用泛型的情况时，会试图检查是否保证了类型安全（如果不能检查，会发出“ unchecked cast ”），然后发出字节码，生成的字节码与等价的非泛型代码、类型强制转换所生成的字节码相同。类似的，自动装箱和增强的 for 循环仅仅是等价的“ 语法糖 ”，只是更复杂的语法和枚举被编译到普通的类中。 因为没有修改JVM的指令集，才可以实现上面的那些功能。说明Java多这些特性的支持只是实现在编译水平上，而非虚拟机的水平上，比如范型，只能在编译的时候检查一下是不是符合类型，而没有在JVM指令上来实现这么一个功能，所以安全性上还是会存在隐患。 最后看一下Java是怎样在编译级别上实现这些功能的。for增强循环：Collection fooCollection = ... for (Foo f : fooCollection) { doSomething(f).}Javac会把上面的代码转换为下面的代码：for (Iterator iter=f.iterator(). f.hasNext().) { Foo f = iter.next(). doSomething(f).} 枚举： javac遇到枚举类型的时候，会创建这个类继承自java.lang.Enum。 自动装箱： 比如Integer

i=10. 则javac生成代码的时候会调用Integer.valueof(10).这么一个方法，Integer.valueof相当于Integer的一个静态工厂方法。 变长参数： 把变长的参数转换为数组。 字符串连接： 适用StringBuilder类来连接，代替了StringBuffer类。 