如何在DOS下面编译有包的Java程序_百度经验

以HelloWorld.java 为例(假设该文件的位置是/home/HelloWorld.java):

```
package a.b;
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello World!");
    }
}
```

下面根据不同的编译方法来讨论：

第一种编译方法(编译直接生成class 文件，执行需先创建包的路径)

假设当前目录为/src/java/ ， 则编译命令为：

```
javac HelloWorld.java
```

假设当前目录为/src/ ， 则编译命令为：

```
javac /src/java/HelloWorld.java
```

或者使用相对路径:

```
javac java/HelloWorld.java
```

执行完该命令后, 在/src/java/ 目录下生成一个HelloWorld.class 文件。执行文件（在java 目录下新建目录a ， 在a 目录下新建目录b 将HelloWorld.class 至于b 目录下；执行java a.b.HelloWorld ），必须要按照包的结构先创建目录。

第二种编译方法(编译直接生成包的路径)

假设当前目录为/src/java/ ， 则编译命令为：

```
javac -d . HelloWorld.java
```

说明："." 为指定编译路径为当前目录；生成的HelloWorld.class 所有目录为/src/java/a/b/HelloWorld.class 。

```
javac -d c/d HelloWorld.java
```

说明：c/d 为指定编译路径为/src/java/c/d ， 同样也可以写成绝对路径如javac -d d:/ HelloWorld.java ， 前提是路径必须先存在；生成的HelloWorld.class 所有目录为/src/java/c/d/a/b /HelloWorld.class 。

假设当前目录为/src/ ， 则编译命令为：

```
javac -d . java/HelloWorld.java
```

说明：生成的HelloWorld.class 所有目录为/src/a/b/HelloWorld.class 。

```
javac -d java/c/d java/HelloWorld.java
```

说明：生成的HelloWorld.class 所有目录为/src/java/a/b/HelloWorld.class 。

第三种编译方法(先把源文件按照包结构放到指定的目录中，然后执行编译命令)

假设当前目录为/src/java/，先在目录中创建目录/a/b，然后编译命令：

```
javac a/b/HelloWorld.java
```

下面总结一下对于带包的类进行编译和执行时的一些要点：

1、编译时可以不考虑包结构的问题，不论用哪种方法,其实本质都是一样的，只需要让javac命令找到所需要编译的原文件(*.java)即可。编译时可以用相对或者绝对路径来为javac命令提供源文件的位置信息。

2、初学者易混淆classpath的作用,对于java命令的-cp选项和javac命令的-classpath选项，以及配置环境变量时的CLASSPATH.其作用是不变的：都是指定所需要的class文件的位置。所不同的是,执行javac编译时的-classpath选项用于指定被编译的源文件需要调用另外的用户自定义类的位置.。执行java命令是根据classpath来寻找所需要执行的class文件的位置；而javac命令不能根据classpath来找源文件,只能根据classpath来寻找所需要用到的类。

下面举例来说明该问题：

假设以下代码(位置:/src/java/code/a/b/TestT.java):

```
package a.b;
import c.d.T;
public class TestT {
   public static void main(String[] args) {
```

```
    T t = new T();
    t.p();
  }
}
package a.b;
import c.d.T;
public class TestT {
   public static void main(String[] args) {
    T t = new T();
    t.p();
  }
}
```

引入的文件(位置:/src/java/code/tmp/c/d/T.java)

```
package c.d;
public class T {
   public void p(){
    System.out.println("class：T");
   }
}
package c.d;
public class T {
   public void p(){
    System.out.println("class：T");
   }
}
```

假设现在编译两个文件(目录：/src/java/)，则编译命令为：javac -classpath code/tmp code/a/b/TestT.java 执行命令为：java -cp code;code/tmp a/b/TestT

如果当前目录为：/src/java/code/，则编译命令为：javac -classpath tmp a/b/TestT.java执行命令为：java -cp .;tmp a/b/TestT

假设现在编译不同磁盘的三个文件(目录：e:/src/java/)，则编译命令为：

假设以下代码(位置:e:/src/java/code/a/b/TestT.java):

view plaincopy to clipboardprint?

```
package a.b;
import c.d.T;
import e.f.T1;
public class TestT {
  public static void main(String[] args) {
   T t = new T();
   t.p();
   T1 t1 = new T1();
   t1.p();
  }
}
package a.b;
import c.d.T;
import e.f.T1;
public class TestT {
  public static void main(String[] args) {
```

```
   T t = new T();
   t.p();
   T1 t1 = new T1();
   t1.p();
  }
}
```

引入的文件1(位置:d:/java/code/tmp/c/d/T.java)

view plaincopy to clipboardprint?

```
package c.d;
public class T {
  public void p(){
   System.out.println("class：T");
  }
}
```

```
package c.d;
public class T {
  public void p(){
   System.out.println("class：T");
  }
}
```

引入的文件2(位置:c:/code/tmp/e/f/T1.java)

view plaincopy to clipboardprint?

```
package e.f;
public class T1 {
  public void p(){
   System.out.println("class：T1");
  }
}
```

```
package e.f;
public class T1 {
  public void p(){
   System.out.println("class：T1");
  }
}
```

如果当前目录为：e:/src/java/

编译命令为：javac -classpath d:/java/code/tmp;c:/code/tmp code/a/b/TestT.java

执行命令为：java -cp code;d:/java/code/tmp;c:/code/tmp a/b/TestT

说明：javac命令中的classpath必须指定引入类的路径；同样java命令中的cp必须引入引入类的class的路径也需指定执行类的路径

实例：

```
package test;
enum T{
    HELLO,WORLD,HAA;
}
```

```
package test;
import static test.T.*;
public class A {
    private T t;
```

```
    public A(T t){
        this.t = t;
    }
    public static void main(String[] args){
        System.out.println(new A(HELLO));
        System.out.println(new A(HAA));
    }
    @Override
    public String toString(){
        return this.t + " ";
    }
}
```

编译命令： javac -d . T.java

 javac -d . A.java

执行命令: java test.A