The code is given:  
  
3.  public class Bertha {  
4.    static String s = "";  
5.    public static void main(String[] args) {  
6.        int x = 4; Boolean y = true; short[] sa = {1,2,3};  
7.        doStuff(x, y);  
8.        doStuff(x);  
9.        doStuff(sa, sa);  
10.       System.out.println(s);  
11.   }  
12.   static void doStuff(Object o) { s += "1"; }  
13.   static void doStuff(Object... o) { s += "2"; }  
14.   static void doStuff(Integer... i) { s += "3"; }  
15.   static void doStuff(Long L) { s += "4"; }  
16. }  
  
What is the result?  
A. 212  
B. 232  
C. 234  
D. 312  
E. 332  
F. 334  
G. Compilation fails  
  
**Answer:  A is correct.**  
  
**Explanation:** It's legal to autobox and then widen. The first call to doStuff(), x is boxed from int to an Integer then passes these two objects (Integer,Boolean) to varargs (Object.... 0).  
  
The second call cannot widen and then box (making the Long method unusable), so it boxes the int to an Integer. And then it goes to doStuff(Object o) because Integer is-A Object. As always, a  
var-args method will be chosen only if no non-var-arg method is possible.  
  
The third call is passing two objects—they are of type 'short array.'  
  
  
1) Widening beats boxing  
2) Widening beats var-args

3) Boxing beats var-args

Now see the rules for overloading methods using widening,boxing and varargs...

1) Primitive widening uses the "smallest" method argument possible.

2) Used individually, boxing and var-args are compatible with overloading.

3) You CANNOT widen from one wrapper type to another. (IS-A fails.)

4) You CANNOT widen and then box. (An int can't become a Long.)

5) You can box and then widen. (An int can become an Object, via Integer.)

6) You can combine var-args with either widening or boxing

public class Boxing1 {  
public static void main(String[] args) {  
Integer i = null;  
method(i);  
}  
static void method(int k){  
System.out.println(k);  
}  
}  
  
What is the output of the above program?  
1)Null Pointer Exception  
2)Number Format Exception  
3)null  
4)0

Resposta: Null Pointer Exception  
  
Explanation :  
When wrapper type is null, we cannot do the boxing conversion. It will throw the NullpointerException when its is trying the convert to primitive type.

public class Boxing2 {  
public static void main(String[] args) {  
byte b = 10;  
method(b);  
}  
static void method(int i){  
System.out.println("Primitivae Type call");  
}  
static void method(Integer i){  
System.out.println("Wrapper Type Call");  
}  
}  
  
What will be output for the above program?  
1)Wrapper Type Call  
2)Primitive Type Call  
3)Compiler Error  
4)Compiles fine, throws runtime exception

2)Primitive Type Call  
  
Explanation :  
When comes to method overloading in Java 5.0, it works like the previous versions. First JVM will check for the matching primitive types, then it will search for the Wrapper types.