java是一个近乎纯洁的面向对象的编程语言，但是为了编程的方便还是引入了基本数据类型，但是为了能够将这些基本的数据类型当作对象操作，java为每一个基本数据类型都引入了对应的包装类型（wrapper class），int的包装类就是integer，从java 5开始引入了自动装箱/拆箱机制，使得两者可以互相转换。

java为每个原始类型提供了包装类型：

-原始类型：boolean、char、byte、short、int、long、float、double

-包装类型：Boolean、Character、Byte、Short、Integer、Long、Float、Double

```
public class Box_Manage {
    public static void main(String [] args) {
        Integer a = new Integer(3);
        Integer b = 3;  //将3自动装箱成Integer类型
        int c = 3;
        System.out.println(a == b); //false
        System.out.println(a == c); //true
    }
}
```

a == b因为两个引用没有引用同一个对象。a == c 是true a自动拆箱int类型再和c比较。

```
public class Box_Manage {
    public static void main(String [] args) {
        Integer f1 = 100, f2 = 100, f3 = 150;
        Integer f4 = 150;
        System.out.println(f1 == f2); //true
        System.out.println(f3 == f4); //false
    }
}
```

简单的说，如果整型字面量得值在-128到127之间，那么不会new一个新的Integer对象，而是直接引用常量池中的Integer对象，所以上面的题目中f1 == f2的结果是true，而f3 == f4的结果是false

```java
public class Box_Manage {
    public static void main(String [] args) {
        Integer f1 = new Integer(100);
        Integer f2 = 100, f3 = 150;
        Integer f4 = 150;
        System.out.println(f1 == f2); //false
        System.out.println(f3 == f4); //false
    }
}
```

当有装箱机制的时候不一定创建对象，如果在范围内会引用常量池中的对象，如果不是利用装箱机制，利用new Integer（）来创建一个对象时，则一定会创建一个对象。