Object类的equals方法 想等测试与继承

类型检测部分：

1. 隐式参数(this)和显式参数(Object otherObject)属于同一个类时，用getClass进行检测

class Employee

{

…

public Boolean equals(Object otherObject)

{

if(this == otherObject) return true;

if(otherObject == null) return false;

**if(getClass != otherObject.getClass()) return false;**

Employee other = (Employee) otherObject;

return Objects.equals(name, other.name) && salary == other.salary && Object.equals(hireDay, other.hireDay);

}

}

1. 隐式参数和显式参数不属于同一类时，如果用instanceof则貌似违反了Java语言规范中对equals方法的**对称性**的要求（x.equals(y)返回true则 y.equals(x)也返回true）,而用getClass有人认为违反了置换原则，对应策略：
   1. 如果子类能够拥有自己的相等概念，则对称性需求将强制采用getClass进行检测，不能用instanceof否则违反对称性

例如：在Employee和Manager比较的例子中，**需要的规则（Employee）**是只要对应的域相等，就认为两个对象相等；**但是Manager中重写了equals方法**：两个Manager对象，对应的姓名、薪水、雇佣日期和奖金相等，才是同一个对象。因此，可以使用getClass检测。

* 1. 如果由超类决定相等的概念，那么就可以使用instanceof进行检测，这样可以在不同类的对象之间进行相等的比较

接上a），如果使用雇员的ID作为检测的标准，那么这个相等的概念适用于所有的类，就可以使用instanceof进行检测，并应该将Employee.equals方法声明为final