System.out.println("p1.equals(p2)? " + p1.equals(p2));

The output of this code snippet is true, because the object p1 uses the class(Person) equals implementation method that only checks the name field

System.out.println("p2.equals(p1)? " + p2.equals(p1));

The output of this code snippet is false, because the p2 is the sub class(PersonWithJob) overrides equals of the super class(Person) thereby checking for name and salary fields ,so since p1 has only name the comparison fails.

```
public class PersonWithJob {

    private double salary;  2 usages
    private Person person;  3 usages

    public double getSalary() { return salary; }
    PersonWithJob(String n, double s) {  1 usage
        person = new Person(n);
        salary = s;
    }

    @Override
    public boolean equals(Object aPerson) {
        if(aPerson == null) return false;
        if(!(aPerson instanceof PersonWithJob)) return false;
        PersonWithJob p = (PersonWithJob)aPerson;
        boolean isEqual = this.person.getName().equals(p.person.getName()) &&
                this.getSalary()==p.getSalary();
        return isEqual;
    }
    public static void main(String[] args) {
        PersonWithJob p1 = new PersonWithJob( n: "Joe",  s: 30000);
        Person p2 = new Person( n: "Joe");
        //As PersonsWithJobs, p1 should be equal to p2
        System.out.println("p1.equals(p2)? " + p1.equals(p2));
        System.out.println("p2.equals(p1)? " + p2.equals(p1));
    }
```