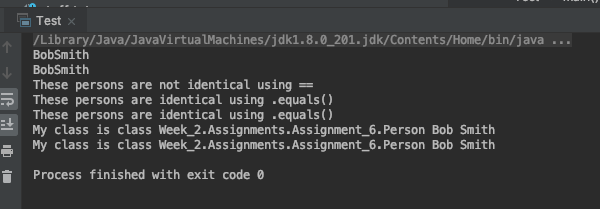


The reason that the == or the .equals() will not work is because you are comparing object references at this point. Since they are different objects they will not match in either instance. As shown with the .toString() method, it shows the reference ID for each instance of the person class that is being used. If you extrapolate it out to use person1.firstName you can begin to compare the values that are assigned to each instance of person.



The reason for the change in the output is because the equals() method has now been overridden to look at the contents of the object. By giving it a reference to go look at person.firstname and person.lastname, you can now compare the values that are assigned to each of these instances instead of just the reference ID associated with the object. The reason that the == still fails is that it compares memory locations for the object and equals() compares the values. Since == is still comparing the object memory locations, it will still fail.

Overriding methods like equals() and toString() gives you some flexibility unto how you want the method to react in certain projects. By doing so, you take existing work and expand it to meet the needs of your project and give you some versatility. Allowing custom outputs for things like .toString() and .equals() allows your classes to have specific outputs and allows for better error handling and reusability later in the project.