

2--In java, **string objects are immutable**. Immutable simply means unmodifiable or unchangeable.

Once string object is created its data or state can't be changed but a new string object is created.

3---Overridings Equals Method:

**class** Person

{

**private** [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) fname;

**private** [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) lname;

**private** **int** age;

**public** Person([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) fname, [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) lname, **int** age)

{

**this**.fname = fname;

**this**.lname = lname;

**this**.age = age;

}

*//Overriding equals*

**public** **boolean** equals([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object) obj)

{

**if** (**this** == obj) **return** **true**;

**if** (obj == **null**) **return** **false**;

**if** (**this**.getClass() != obj.getClass()) **return** **false**;

Person that = (Person) obj;

**if** (**this**.age != that.age) **return** **false**;

**if** (!**this**.fname.equals(that.fname)) **return** **false**;

**if** (!**this**.lname.equals(that.lname)) **return** **false**;

**return** **true**;

}

}

**public** **class** EqualsDemo

{

**public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) str[])

{

Person p1 = **new** Person("Anushka", "Krishan", 7);

Person p2 = **new** Person("Anushka", "Krishan", 7);

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(p1.equals(p2));

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(p1.equals(p1));

[System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(p2.equals(p1));

}

}

OUTPUT

======

D:\JavaPrograms>javac EqualsDemo.java

D:\JavaPrograms>java EqualsDemo

**true**

**true**

**true**

In above piece of code class Person has an overridden equals() method, which took the following step-by-step approach:

1. If the reference to this object is the same as the reference to the argument object, return true. This test saves the work of doing all the other checks in this case.
2. If the argument is null, return false.
3. If the objects are not from the same class, return false. To determine an object's class, we use getClass(). Note that we can use == to tell us whether two objects of type Class are equal because getClass() is guaranteed to return the same reference for all objects in any given class.
4. Cast the argument from Object to Person (this cast must succeed because of the previous test).
5. Return false if any instance variables do not match.

Once you have overridden equals(), you have made your class comparable