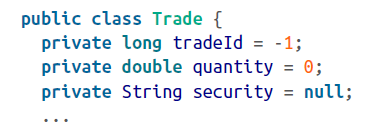
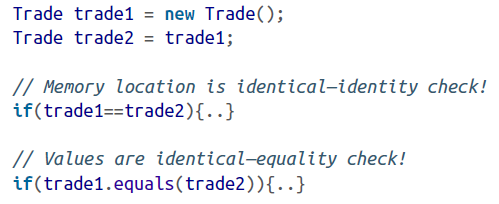
**Identity Mismatch**

The objects in a java application, for example, have both identity and equality. For example, see the definition of the Trade POJO described here:



Two objects represent the same memory location, so they are treated as identical:



When we are comparing the trades using the == operator, we are comparing bases on the memory location. This check will compare the memory address of two trade objects, and if they have the same address (like trade1 and trade2), they are identical.

On the other hand, if the values of an object (trade1) are equal to another (trade2) object, we say these objects are equal. In the preceding example, the second if block using the equals method indicates this. As developers, we are responsible for writing a valid equals method.

In the java world, we know the difference between using == versus *equals* method for equality.

In a relational schema, however, there are no identity and equality concepts. The rows (or records) are identical with their values in the columns. To overcome the identity and equality feature deficiencies, databases tend to employ primary key strategies. These primary identifiers, which represent individual rows, are then modeled as one of the object properties to bridge the gap. For Example, the trade’s traded attribute will be mapped to a TRADE\_ID primary key on the TRADES table.