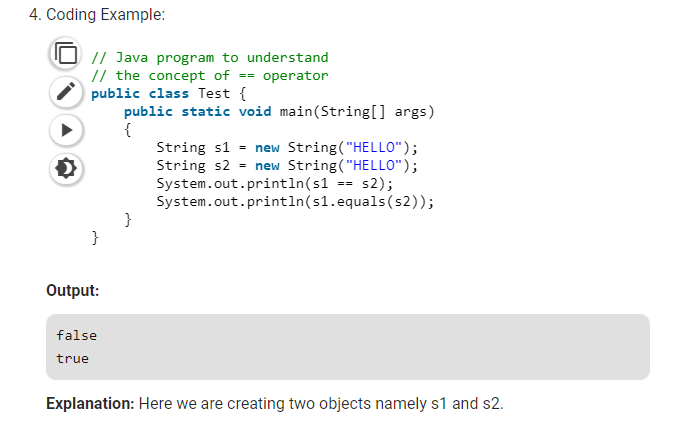
In general both equals() and “==” operator in Java are used to compare objects to check equality but here are some of the differences between the two:

1. Main difference between .equals() method and == operator is that one is method and other is operator.
2. We can use == operators for reference comparison (**address comparison**) and .equals() method for **content comparison**. In simple words, == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.
3. If a class does not [override the equals method](https://www.geeksforgeeks.org/overriding-equals-method-in-java/), then by default it uses equals(Object o) method of the closest parent class that has overridden this method. See [this](https://www.geeksforgeeks.org/override-equalsobject-hashcode-method/)for detail

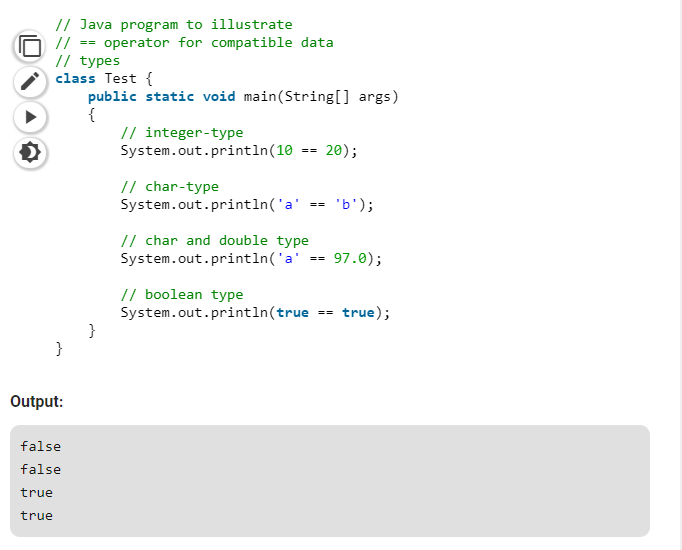


* Both s1 and s2 refers to different objects.
* When we use == operator for s1 and s2 comparison then the result is false as both have different addresses in memory.
* Using equals, the result is true because its only comparing the values given in s1 and s2.

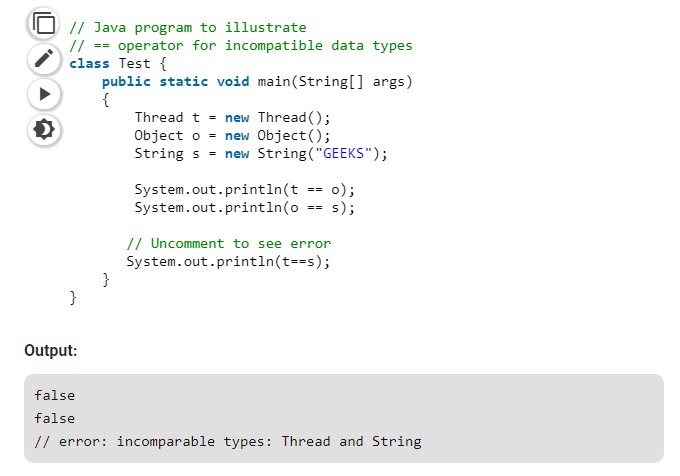
Let us understand both the operators in detail:

**Equality operator(==)**

We can apply equality operators for every primitive types including boolean type. we can also apply equality operators for object types.

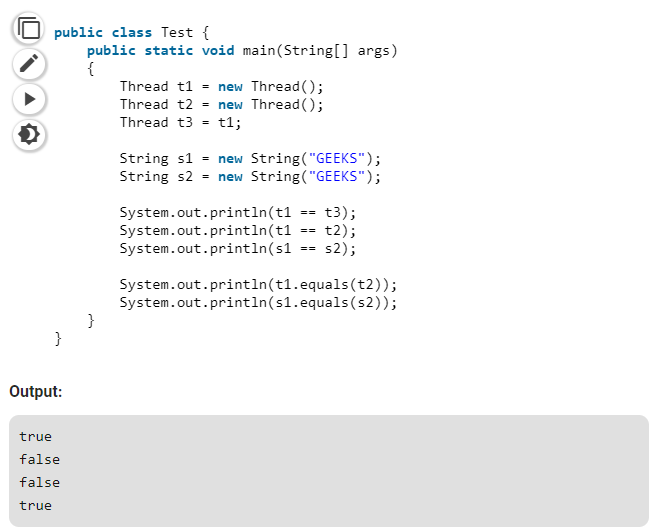


If we apply == for object types then, there **should be compatibility between arguments types** (either child to parent or parent to child or same type). Otherwise we will get compile time error.



**.equals()**

In Java, string equals() method compares the two given strings based on the data/content of the string. If all the contents of both the strings are same then it returns true. If all characters are not matched then it returns false.



**Explanation:** Here we are using .equals method to check whether two objects contains the same data or not.

* In the above example, we are creating 3 Thread objects and 2 String objects.
* In the first comparison, we are checking that t1 == t3 or not. As we know that both **t1 and t3 pointing to same object** that’s why it returns true.
* In second comparison, we are using operator “==” for comparing the String Objects and not the contents of the objects. Here, both the objects are different and hence the outcome of this comparison is “False”
* When we are comparing 2 String objects by .equals() operator then we are checking that is both objects contains the same data or not.
* Both the objects contains the same String i.e. GEEKS that’s why it returns true.