

NAME _____

DATE _____

PERIOD _____

UNIT 2 · TOPIC 2.6 · WORKSHEET

Same or Equal?

Prove two conditions are twins, untangle a not, and find out which Strings are really the same object.

About 35 minutes

10 questions

Cumulative through 2.6

TOPIC

2.6

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints. Some puzzles include pieces you won't need.

PART A · 3 QUESTIONS

A

Warm up your brain

1**PENCIL CHECK**

a and b are boolean variables. Fill in the truth table with true or false, then answer the question under it.

a	b	!(a && b)	!a !b	!a && !b
true	true			
true	false			
false	true			
false	false			

Which two expressions are equivalent? Circle their headers.

2 YOU'RE THE JVM

You are the Java Virtual Machine. Run this code in your head. What prints?

```
String p = new String("frog");
String q = p;
String r = new String("frog");
System.out.println((p == q) + " " + (p == r) + " " + p.equals(r));
```

- | | |
|---------------------------|---------------------------|
| A true false true | B true true true |
| C false false true | D true false false |

3 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
String a = new String("lily");
String b = a;
String c = new String("lily");
b = c;
```

- 1 After the last line, `a == b` is false.
- 2 After the last line, `b == c` is true.
- 3 `a.equals(b)` is false, because `a` and `b` now refer to different objects.

THE LIE, REWRITTEN



THINK FIRST

Is `!(x > 5)` the same as `x < 5` for every `int` value of `x`? Guess in the margin; question 4 will show you how to check.

PART B · 3 PUZZLES

B Puzzles

4 MATCH-UP

Each expression on the left has exactly one equivalent in the list below: the same value for **every** pair of `int` values `x` and `y`. Write its letter. Two letters in the list are look-alikes and aren't used.

	Expression	Equivalent to
1	<code>!(x > 5 && y > 5)</code>	
2	<code>!(x > 5 y > 5)</code>	
3	<code>!(x >= 5)</code>	
4	<code>!(x == 5 y != 5)</code>	

- **P** `x <= 5 && y <= 5`
- **Q** `x <= 5 || y <= 5`
- **R** `x < 5`
- **S** `x != 5 && y == 5`
- **T** `x < 5 || y < 5`
- **U** `x <= 5`

5 SNIPPET POOL

Teen Night at the pond is for ages 13 through 19. The sign-up program keeps everyone else out with `!(age < 13 || age > 19)`. Fill the holes so `allowed` holds the same value as original for **every** age, with no `!` in sight.

TeenNight.java

```
// age already holds the sign-up age
boolean original = !(age < 13 || age > 19);
boolean allowed = age (1) 13 (2) age (3) 19;
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

<	>	<=	
>=	&&		



FAIR QUESTION

If the two versions match for my test age, aren't they equivalent?

Not yet. Equivalent means the same value in **every** case. A few matching tries prove nothing, but one case that differs proves they aren't equivalent. The values right on a boundary, like the cutoff number itself, are where look-alikes usually split apart.

6

CODE TILES

Two variables can share an object and then stop sharing it. The first line stayed on the fridge. Put the tiles in order so the program prints the output below. One tile is a decoy.

```
String a = new String("pad");
```

```
System.out.println("after: " + (a == b));
```

```
b = new String("pad");
```

```
String b = new String("pad");
```

```
System.out.println("text: " + a.equals(b));
```

```
String b = a;
```

```
System.out.println("before: " + (a == b));
```

Target output

```
before: true
```

```
after: false
```

```
text: true
```

YOUR PROGRAM

PART C · 4 QUESTIONS

C

Catch the mistake

7

YOU'RE THE COMPILER

You are the compiler. For each snippet, decide whether it compiles. If it does, write what it prints, or write *crashes* and the exception. If it doesn't compile, say why in a few words.

Snippet	Compiles?	Output, or why not
1 String s = null; System.out.println(s.length() > 0);		
2 String s = null; boolean ok = s != null && s.length() > 0; System.out.println(ok);		
3 String s = ""; boolean ok = s == null s.length() == 0; System.out.println(ok);		
4 String s = null; System.out.println("hi".equals(s));		
5 int n = 5; System.out.println(n == null);		
6 String s = "pad"; System.out.println(!s == null);		

8

ONE-TOKEN FIX

Ana tried to rewrite `!(score >= 70 && score <= 90)` without the `!`. Her version on line 3 is false for every score. Change exactly one token on line 3 so rewrite equals original for **every** score.

```

1 int score = 65;
2 boolean original = !(score >= 70 && score <= 90);
3 boolean rewrite = score < 70 && score > 90;
4 System.out.println(original + " " + rewrite);

```

OLD TOKEN, NEW TOKEN, AND WHY ANA'S VERSION IS NEVER TRUE



WATCH OUT

A truth table proves equivalence only when it's complete. With two boolean variables there are 4 rows, and with three there are 8. Checking just the rows where the columns happen to agree proves nothing: `a == b` and `a != b` match in three of their four rows, and they are not equivalent.

9 WHO AM I?

Some ideas from this topic are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
We're two Boolean expressions that give the same value in every single case. We're...	
I list every combination of <code>true</code> and <code>false</code> so nothing gets skipped.	
I turn <code>!(a b)</code> into <code>!a && !b</code> .	
I'm a second variable holding a reference to the very same object.	
I mean "this variable refers to no object at all."	
I'm the <code>String</code> method that compares characters, not references.	

10 WRAP-UP



REFLECTION

In one sentence: when do you compare two `String` variables with `==`, and when with `equals`?

YOUR ANSWER