

2.6 Comparing Boolean Expressions

Lesson Plan

Pacing: 2 class periods of 45 minutes

Learning objectives: 2.6.A, 2.6.B

Essential knowledge: 2.6.A.1, 2.6.A.2, 2.6.B.1, 2.6.B.2, 2.6.B.3

Materials

- Projector and the Same or Equal? slide deck
- Whiteboard and markers
- Mini whiteboards or scrap paper
- IDE on every computer (optional demonstrations)
- Printed Same or Equal? puzzle worksheet (answer key for you) and exit ticket

Agenda

Minutes	Activity
5	Day 1 warm-up — two signs at the pool (slide 2)
15	Day 1 direct instruction — equivalent expressions and truth tables (slides 3–7)
15	Day 1 direct instruction — De Morgan's laws (slides 9–12)
8	Day 1 guided practice — equivalent or not, push the ! inside (slides 8 and 13)
2	Day 1 closure — what is a counterexample?
5	Day 2 warm-up — same text, same object?
15	Day 2 direct instruction — == on objects, aliases, equals (slides 14–17)
8	Day 2 direct instruction — null and the null check (slide 18)
7	Day 2 guided practice — what prints (slide 19)
5	Day 2 closure — vocabulary and sentence starter (slides 20–21)
5	Day 2 exit ticket

Overview

Day 1 is comparing Boolean expressions: two expressions are equivalent when they have the same value in every case, so one counterexample disproves it and a complete truth table proves it (2.6.A.1), and De Morgan's laws rewrite `!(a && b)` as `!a || !b` and `!(a || b)` as `!a && !b` (2.6.A.2). Day 2 is comparing objects: `==` and `!=` on references ask whether two variables refer to the same object, aliases share one object, and `null` means no object (2.6.B.1, 2.6.B.2); `equals` is the method a class uses to decide when two objects count as equivalent, and `String` compares characters (2.6.B.3).

Students know `!`, `&&`, and `||` and short-circuit evaluation (Topic 2.5), references and `null` (Topics 1.13 and 1.14), and `equals` on `String` (Topic 1.15). Stay inside the topic boundary: no loops, and no

writing an `equals` method of your own (classes and methods come in Unit 3). Truth tables have two variables on Day 1; mention that three variables have eight rows.

TEACHER NOTE

The slides are the Same or Equal? deck (21 slides, with speaker notes); slide numbers below refer to it. Day 1 uses slides 1–13 and Day 2 uses slides 14–21. The Same or Equal? puzzle worksheet (10 questions, about 35 minutes) is split across the two nights: questions 1, 4, 5, and 8 after Day 1; questions 2, 3, 6, 7, 9, and 10 after Day 2.

Files whose names end in `.snippet.java` hold only the statements that go inside `main`. All files are in the topic's `examples/` folder. Everything here is optional to run; the slides carry the examples.

- **Day 1 demonstrations:** `FreezingCheck`, `PassingCheck`, `WrongNegation`, `DeMorganTable`, `NotBothTable`, `ImpliesTable`, `CounterExample`, `OrVersusNotEqual`, `TeenCheck`, `TeenWrong`, `StoreHoursWrong`, `StoreHoursFixed`, `MissingParens`.
- **Day 2 demonstrations:** `SameObject`, `AliasTrace`, `ReassignAlias`, `EqualsVsDoubleEquals`, `PasswordWrong`, `PasswordFixed`, `NullCheck`, `NullGuard`, `EmptyNotNull`, `EqualsOnNull`, `EqualsNullArgument`, `LiteralFirstEquals`, `GuardWrongOrder`.
- **Printed only (exit ticket):** `ExitRefs`.

Day 1 warm-up: two signs at the pool

Show slide 2. Sign 1 says "swim unless it is raining or cold", and sign 2 says "swim if it is not raining or not cold". Students fill the four rows: with raining and cold both yes, neither sign says swim; with raining yes and cold no, sign 1 says no and sign 2 says yes; with raining no and cold yes, sign 1 says no and sign 2 says yes; with neither, both say swim. Ask: do the two signs always agree? No, so they are not saying the same thing, and today's lesson is how to tell.

Day 1 direct instruction: equivalent expressions and truth tables

1. **Goals** (slide 3). Goals 1–3 are Day 1; goals 4 and 5 are Day 2.
2. **Same value in every case** (slide 5; 2.6.A.1). Two Boolean expressions are equivalent when they evaluate to the same value in all cases. One case where they differ, a counterexample, proves they are not. Test the value right on the boundary: with `temp 32`, `!(temp < 32)` is `true` and `temp > 32` is `false`, so they are not equivalent; `temp >= 32` is `true` and matches. The opposite of `<` is `>=`, not `>`.
3. **A truth table proves it** (slide 6; 2.6.A.1). List every combination of values. If two columns match in every row, the expressions are equivalent. Count the rows: 4 for two variables, 8 for three. Walk the table for `!(a || b)` and `!a && !b`: they match in all four rows.
4. **Three matching rows prove nothing** (slide 7). `a && b` and `a == b` agree in rows 1–3 and differ in row 4, so they are not equivalent.

MISCONCEPTION It matched for my test values, so they are equivalent

Students test one or two values, see a match, and stop. Matching cases prove nothing; a single mismatch proves they differ, and a complete truth table is the only way to prove equivalence. Boundary values are where look-alikes split apart. This is the Fair Question on worksheet question 5, and the warning below worksheet question 8.

MISCONCEPTION The opposite of $x < 5$ is $x > 5$

Students negate a comparison by flipping the symbol and forget the boundary. The opposite of $<$ is $>=$, of $>$ is $<=$, and of $=$ is $!=$. Use `WrongNegation` or `PassingCheck` to show the failing value (the boundary itself). This is the Think First prediction on worksheet question 4.

Day 1 direct instruction: De Morgan's laws

1. **Negate each part, flip the operator** (slide 10; 2.6.A.2). $!(a \ \&\& \ b)$ is the same as $!a \ || \ !b$, and $!(a \ || \ b)$ is the same as $!a \ \&\& \ !b$. Put the table of opposite comparisons on the board: $<$ and $>=$, $>$ and $<=$, $=$ and $!=$.
2. **Rewrite it without the !** (slide 11). $!(level > 3 \ || \ lives == 0)$: flip $||$ to $\&\&$, then negate each part, giving $level <= 3 \ \&\& \ lives != 0$. The check at the bottom shows both are true for `level 2` and `lives 3`.
3. **Half a De Morgan is wrong** (slide 12). With $x \ 1$ and $y \ -1$, $!(x > 0 \ \&\& \ y > 0)$ is true, but $x <= 0 \ \&\& \ y <= 0$ is false. The correct rewrite is $x <= 0 \ || \ y <= 0$.

MISCONCEPTION Negate the parts, but keep the operator

Students negate each comparison and leave $\&\&$ or $||$ alone, or flip the operator and forget to negate the parts. Both steps are required. Teach the two-step routine on slide 11 and have students check a rewrite with one test case before they trust it. This is worksheet question 8.

Day 1 guided practice: equivalent or not, push the ! inside

1. **Equivalent, or not?** (slide 8). Pairs on mini whiteboards: 1, $!(n > 0)$ and $n <= 0$, are equivalent. 2, $!(n == 3)$ and $n > 3$, are not: `n 1` is a counterexample (true and false). 3, $!(done \ || \ late)$ and $!done \ \&\& \ !late$, are equivalent. 4, $!(ready \ \&\& \ !tired)$ and $!ready \ || \ tired$, are equivalent (the $!$ on `tired` cancels).
2. **Push the ! inside** (slide 13). Students rewrite with no $!$ in front of the parentheses: 1, `isOpen || !hasRoom`. 2, `age < 16 && !hasPermit`. 3, `count == 0 || total <= 100`.

TEACHER NOTE

Run slide 8 first, before slide 10 if possible, so students lean on counterexamples. In pairs, ask each pair to find a counterexample for every "not equivalent" answer before showing the class.

Day 1 independent practice and closure

Puzzle worksheet questions 1, 4, 5, and 8 are homework: the truth table, the match-up, the snippet pool (age >= 13 && age <= 19), and the one-token fix (change && to ||). For closure, ask: what is a counterexample, and why is one enough?

Day 2 warm-up: same text, same object?

Show two pictures on the board: two sticky notes that both say "pond", and one sticky note with two arrows pointing to it. Ask which is "the same." Take answers, then say that Java has both meanings, and it uses == for one and equals for the other.

Day 2 direct instruction: == on objects, aliases, equals

1. **== asks: the same object?** (slide 15; 2.6.B.1). `new String("pond")` twice makes two objects, so `home == away` is `false`. After `away = home`, both variables refer to the same object, so `home == away` is `true`. Two variables that refer to the same object are **aliases**. Draw the before and after diagram on the board.
2. **Never compare text with ==** (slide 16). `typed == saved` with two separate objects displays `Try again`, even though the characters match; `typed.equals(saved)` displays `Welcome`.
3. **Each class decides what equal means** (slide 17; 2.6.B.3). `a == b` asks about the same object, `a != b` asks about different objects, and `a.equals(b)` asks whether the contents are the same. A class defines its own `equals` method, usually by comparing attributes; `String` compares the characters. Students do not write `equals` in this topic.

MISCONCEPTION == compares the text of two Strings

Students used == on numbers, so they expect it to compare contents. For objects it compares references, and two separate objects with the same characters give `false`. The fix is `equals`. Worksheet question 2 (answer A, not B) and question 10 return to this.

MISCONCEPTION Reassigning one alias changes the other variable too

After `b = c`, students think `a` follows `b`. A reassignment changes only that variable's reference. `a` still refers to its original object. This is the lie in worksheet question 3 (claim 3) and the code tiles in question 6.

TEACHER NOTE

Use `new String("...")` in demonstrations, as the guide does, so two equal Strings are really separate objects. String literals can share one object, so == can give a surprising `true`; do not go into that now.

Day 2 direct instruction: null and the null check

Show slide 18 (2.6.B.2). `x == null` is `true` when `x` refers to no object. Check it before calling a method, or the program throws a `NullPointerException`. `pick != null && pick.equals("blue")` is safe because of short-circuit evaluation (Topic 2.5); with `pick` as `null`, the right side never runs. An empty string `""` is not `null`: it is a real object with no characters, so `pick.length()` gives 0.

MISCONCEPTION An empty string is null

Students use the two words as if they meant the same thing. `null` means no object at all, so calling `length()` on it crashes; `""` is a real `String` with length 0. Worksheet question 7, snippets 1 and 3, shows the contrast.

MISCONCEPTION The null check can go after the method call

Students write `c.length() > 2 || c == null`. The left side runs first and crashes. Same rule as the guard in Topic 2.5: the null check must be on the left. Use `GuardWrongOrder` to show the exception.

Day 2 guided practice: what prints

Show slide 19 with `a` as `new String("hop")`, `b = a`, and `c` as `null`. Students write each line's output on mini whiteboards. A, `a == b`, is `true` (aliases). B, `a.equals(new String("hop"))`, is `true` (same characters). C, `c == null || c.length() > 2`, is `true`, because the left side is true and the call is skipped. D, `c.length() > 2 || c == null`, crashes with a `NullPointerException`, because `c.length()` runs first. Ask what small change would make D safe.

Day 2 independent practice

Puzzle worksheet questions 2, 3, 6, 7, 9, and 10 are homework: the JVM trace on `==` and `equals`, two truths and a lie about aliases, the code tiles (alias, then a new object), you're the compiler (null checks, and `n == null` for an `int`, which does not compile), who am I, and the wrap-up sentence.

Day 2 closure and exit ticket

Show slide 20 (vocabulary), then slide 21 and have students finish the sentence starter. Give the exit ticket (about five minutes). Question 1 checks De Morgan's laws (2.6.A.2), question 2 checks `==`, aliases, and `equals` (2.6.B.1, 2.6.B.3), and question 3 checks proving or disproving equivalence (2.6.A.1). Re-teach the weakest one at the start of Topic 2.7.