

2.5 Compound Boolean Expressions

Lesson Plan

Pacing: 3 class periods of 45 minutes

Learning objectives: 2.5.A

Essential knowledge: 2.5.A.1, 2.5.A.2

Materials

- Projector and the And, Or, Not slide deck
- Whiteboard and markers
- Mini whiteboards or scrap paper
- IDE on every computer (optional demonstrations)
- Printed And, Or, Not puzzle worksheet (answer key for you) and exit ticket

Agenda

Minutes	Activity
5	Day 1 warm-up — who gets into the skate park (slide 2)
10	Day 1 direct instruction — the NOT operator (slides 3–6)
15	Day 1 direct instruction — AND and OR (slides 7–11)
12	Day 1 guided practice — true or false (slide 12)
3	Day 1 closure — "and" or "or"?
5	Day 2 warm-up — read it like Java
17	Day 2 direct instruction — the order of <code>!</code> , <code>&&</code> , and <code> </code> (slides 13–15)
12	Day 2 guided practice — what prints (slide 16)
8	Day 2 independent practice — worksheet questions 1 and 2 in class
3	Day 2 closure — add the parentheses
5	Day 3 warm-up — which side runs?
15	Day 3 direct instruction — short-circuit evaluation and the guard (slides 17–19)
10	Day 3 guided practice — crash or no crash (slide 20)
7	Day 3 independent practice — worksheet question 5 in pairs
3	Day 3 closure — vocabulary and sentence starter (slides 21–22)
5	Day 3 exit ticket

Overview

Day 1 is the three logical operators: `!` (not) flips one `boolean` value (2.5.A.1), and `&&` (and) and `||` (or) join two `boolean` expressions into one value (2.5.A.1). Day 2 is the order they run in, `!` first, then `&&`, then `||`, and how parentheses change it (2.5.A.1). Day 3 is short-circuit evaluation: Java

skips the right side of `&&` or `||` when the left side already decides the answer, and a guard on the left keeps a risky right side from running (2.5.A.2).

Students know the relational operators (Topic 2.2), `if` statements (Topic 2.3), and nested `if` and `else if` chains (Topic 2.4). Stay inside the topic boundary: no truth-table equivalence, De Morgan's laws, or comparing objects with `==` (Topic 2.6), and no loops. Say that rewriting a condition into an equivalent one is the next topic.

TEACHER NOTE

The slides are the And, Or, Not deck (22 slides, with speaker notes); slide numbers below refer to it. Day 1 uses slides 1–12, Day 2 uses slides 13–16, and Day 3 uses slides 17–22. The And, Or, Not puzzle worksheet (10 questions, about 35 minutes) is split across the three nights: questions 2 and 7 after Day 1; questions 1, 4, and 8 after Day 2; questions 3, 5, 6, 9, and 10 after Day 3.

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:** `NotBasics`, `NotBattery`, `NotWithoutParens`, `AndTeen`, `AndDrive`, `OrWeekend`, `OrFree`, `OrBoth`, `ChainedCompare`, `OrShorthand`, `OrIntOperand`, `AndWhenOr`.
- **Day 2 demonstrations:** `PrecedenceAndFirst`, `PrecedenceNot`, `PrecedenceTrace`, `PopcornBug`, `PopcornFixed`, `MonthBug`, `MonthFixed`.
- **Day 3 demonstrations:** `GuardAnd`, `GuardOr`, `WordGuard`, `WordGuardSwapped`, `SafeGuards`, `QuotientCrash`, `NestedGuard`, `MixedGuard`.
- **Printed only (exit ticket):** `ExitLogic`.

Day 1 warm-up: who gets into the skate park

Show slide 2. Students fill in the table for the rule "you need a helmet and you must be signed in": Ana gets in, and Ben, Cy, and Dee do not. Then change the rule to "helmet or signed in": Ana, Ben, and Cy get in, and Dee does not. Ask which word is stricter. Name it: this is Boolean logic, and today's operators are Java's versions of "and", "or", and "not".

Day 1 direct instruction: the NOT operator

1. **Goals** (slide 3). Goals 1 and 2 are Day 1, goal 3 is Day 2, and goals 4 and 5 are Day 3.
2. **! flips a value** (slide 5; 2.5.A.1). `!a` is true when `a` is false, and false when `a` is true. Show `!muted` with `muted` true, giving `false`. To flip a comparison, wrap the whole comparison in parentheses: `!(volume < 20)` is `!(false)`, which is true for `volume` 35.
3. **! grabs only the value next to it** (slide 6). `!volume < 20` is read as `(!volume) < 20`, and `!` cannot flip an `int`, so it does not compile (bad operand type `int` for unary operator `!`). The fix is the parentheses.

MISCONCEPTION ! can go in front of a comparison without parentheses

! has higher precedence than every comparison, so it grabs only the value right next to it. Whenever students want to negate a comparison, they write !(first and close it after the comparison. Show the compiler message once (NotWithoutParens).

Day 1 direct instruction: AND and OR

1. **Both, or at least one** (slide 8; 2.5.A.1). `a && b` is true only when both sides are true. `a || b` is true when at least one is true, including both. Walk the four rows of the table on the slide, and point out that the whole expression is one `boolean` value.
2. **A range and a choice** (slide 9). `minutes >= 30 && minutes <= 60` displays `Just right` for 45. `day.equals("Sat") || day.equals("Sun")` displays `Sleep in` for "Sun". A range uses `&&`; a choice among values uses `||`.
3. **Each side must be a whole boolean** (slide 10). `30 <= minutes <= 60` does not compile, because `30 <= minutes` is a `boolean` that cannot then be compared with `60`; write `30 <= minutes && minutes <= 60`. `day.equals("Sat") || "Sun"` does not compile either, because "Sun" is not a `boolean`.
4. **When English "and" means ||** (slide 11). "Students and teachers get 10% off" means a student or a teacher; `isStudent && isTeacher` is true only for someone who is both. Ask students which question to ask: must one person pass both tests, or is either test enough?

MISCONCEPTION Leaving out the second comparison

Students write `x > 5 || 10`, `month == 12 || 1`, or the chained `1 <= x <= 12`, the way they would say it in English or write it in math. Java never fills in the missing half. Each side of `&&` and `||` must be a complete `boolean` expression, which means repeating the variable. This is worksheet question 6, snippets 1 and 2.

MISCONCEPTION English "and" always means &&

Sentences like "kids under 5 and seniors ride free" use "and" for two groups, which is `||` in Java. Nobody is both under 5 and over 64, so `&&` lets nobody in. Teach the question from slide 11 and use it again on worksheet question 7.

Day 1 guided practice: true or false

Show slide 12. Students work out each value on mini whiteboards, using `hasKey true`, `locked true`, code `1234: hasKey && !locked` is `false`; `!hasKey || locked` is `true`; code `== 1234 && hasKey` is `true`; `!(code == 1234) || !hasKey` is `false`. Ask for the value of each side before the whole expression.

TEACHER NOTE

Insist that students write the value of each side underneath the expression, as in `true && false`, before the answer. It builds the habit they need for precedence tomorrow.

Day 1 independent practice and closure

Puzzle worksheet questions 2 and 7 are homework: the pencil check on `!`, `&&`, and `||`, and the one-token fix (change `&&` to `||`) for the weekend sale. For closure, ask for one rule from school that needs `&&` and one that needs `||`.

Day 2 warm-up: read it like Java

Write `a || b && c` on the board with `a` true, `b` false, `c` false. Students decide the value and write which part they did first, then compare. Some will go left to right and get `false`. Do not resolve it yet; the first slide of the lesson does.

Day 2 direct instruction: the order of `!`, `&&`, and `||`

1. **The order** (slide 14; 2.5.A.1). Parentheses first, then `!`, then the comparisons, then `&&`, then `||`. Compare it with `2 + 3 * 4`: `&&` is the "times" and `||` is the "plus". `a || b && c` is read as `a || (b && c)`, which is `true` for the warm-up values.
2. **Say it with parentheses** (slide 15). `member || total > 50 && thisWeek` with `member` true, `total` 20.0, and `thisWeek` false displays `true`, because `&&` goes first and a member is true no matter what. `(member || total > 50) && thisWeek` displays `false`. The two lines mean different rules; the parentheses say which one is meant.
3. Add parentheses even where they are not needed when it makes the meaning clear.

MISCONCEPTION Java reads `&&` and `||` left to right

Students apply the left-to-right rule they learned for `+` and `*` of the same precedence, but `&&` and `||` have different precedence. Write `a || b && c` and ask where the invisible parentheses go. This is worksheet question 1 (answer A, not B) and the lie in question 8 (claim 2).

Day 2 guided practice: what prints

Show slide 16 with `x` 4. Students write the three lines and underline the operator that runs first on each. Line 1, `x > 0 || x > 10 && x < 2`, displays `true`: `&&` goes first, giving `false`, and `true || false` is `true`. Line 2, `!(x > 3) && x < 10`, displays `false`: the parentheses make `x > 3` go first, `!` flips `true` to `false`, and `false && true` is `false`. Line 3, `!(x > 3 && x < 10)`, displays `false`: the `&&` inside the parentheses is `true`, and `!` flips it. Contrast lines 2 and 3, which differ only in where the parentheses sit.

Day 2 independent practice and closure

Students begin worksheet questions 1 and 2 in class (about eight minutes) while you circulate. Finish questions 1, 4, and 8 tonight: the JVM trace, the code tiles for the Lily Pad Plunge (a variable must exist before it is used, and the decoy uses `||`), and two truths and a lie about `hasPass || weekend && !rainy`. For closure, write `a && b || c` and ask a student to add the parentheses that show how Java reads it.

Day 3 warm-up: which side runs?

Ask: for `false && something`, do we need to know `something` to know the answer? And for `true || something`? Take a few answers, and say that today's lesson is about what Java does with that fact.

Day 3 direct instruction: short-circuit evaluation and the guard

1. **Skipping the right side** (slide 18; 2.5.A.2). Java evaluates the left side first. `false && ...` is always false and `true || ...` is always true, so Java skips the right side. That is **short-circuit evaluation**. Trace `word` as the empty string: `word.length() > 0` is false, so `word.substring(0, 1)` is never evaluated, and `No h at the start` is displayed.
2. **The guard goes on the left** (slide 19). With the two comparisons swapped, `word.substring(0, 1)` runs first on the empty string and throws a `StringIndexOutOfBoundsException`. It compiles, so the compiler cannot warn you; it crashes only when it runs. Connect it to the guard in Topic 2.4: there the guard was an outer `if`, here it is the left side of `&&`.

MISCONCEPTION Java evaluates both sides every time

Students believe the right side always runs, so they see no reason to care about the order of the two sides. Show `SafeGuards`: with `n 0`, `n != 0 && 10 / n > 1` and `n == 0 || 10 / n > 1` run without crashing, while `QuotientCrash` (division on the left) crashes. This is what worksheet question 3 (face-off) and question 5 (snippet pool) settle.

MISCONCEPTION Short-circuiting makes the guard order unimportant

Short-circuiting only skips the right side. A risky expression on the left runs before the guard is ever checked, so the guard must be the left operand of `&&` (or, with `||`, a left side that is true exactly when the right side is unsafe). Ask students every time: "Which side runs first, and can it crash?"

Day 3 guided practice: crash or no crash

Show slide 20 with `boxes 0` and `cookies 24`. Students write each line's output or `crash`. A, `boxes == 0 || cookies / boxes > 5`, displays `true`: the left side is true, so the division is skipped. B, `boxes != 0 && cookies / boxes > 5`, displays `false`: the left side is false, so the division is skipped. C, `cookies / boxes > 5 || boxes == 0`, crashes with an `ArithmeticException`, because the division is on the left and runs first. Ask why A and C use the same two tests and behave differently.

Day 3 independent practice

In pairs, students do worksheet question 5, the grade-book snippet pool:

`count > 0 && total / count >= 80`. They must say what happens when `count` is 0 and why. The rest of the worksheet is homework: questions 3, 6, 9, and 10 (the face-off, you're the compiler, who am I, and the wrap-up). Question 5 settles the Think First prediction on question 4 (done the night before; students can check it then).

Day 3 closure and exit ticket

Show slide 21 (vocabulary), then slide 22 and have students finish the sentence starter, "I use `&&` when __, and `||` when __." Give the exit ticket (about five minutes). Question 1 checks values and the order of operators (2.5.A.1), question 2 checks short-circuit evaluation and the guard (2.5.A.2), and question 3 checks building a compound condition correctly (2.5.A.1). Re-teach the weakest one at the start of Topic 2.6.