

2.4 Nested if Statements

Lesson Plan

Pacing: 2 class periods of 45 minutes

Learning objectives: 2.4.A

Essential knowledge: 2.4.A.1, 2.4.A.2, 2.4.A.3

Materials

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

Agenda

Minutes	Activity
5	Day 1 warm-up — a question inside a question (slide 2)
15	Day 1 direct instruction — nested if and the guard (slides 3–7)
15	Day 1 direct instruction — nesting in else, the inner else, the dangling else (slides 8–11)
8	Day 1 guided practice — what prints for each player (slide 12)
2	Day 1 closure — which else?
5	Day 2 warm-up — nested or chained?
15	Day 2 direct instruction — multiway selection, order, trailing else (slides 13–17)
12	Day 2 guided practice — shipping tiers and write the chain (slides 18–19)
3	Day 2 closure — vocabulary and sentence starter (slides 20–21)
10	Day 2 independent practice and exit ticket

Overview

Day 1 is nesting: an `if` or `if-else` inside the body of another one, and why the inner condition is checked only when the program reaches it (2.4.A.1, 2.4.A.2). Day 2 is multiway selection: an `if`, `else if`, `else` chain, in which at most one body runs, the order of the conditions matters, and the trailing `else` catches everything left over (2.4.A.3).

Students know the relational operators (Topic 2.2) and `if` and `if-else` statements (Topic 2.3). Stay inside the topic boundary: conditions are single comparisons or `boolean` variables, with no `&&`, `||`, or `!` (that is Topic 2.5), and no `while` or `for` loops. Strings appear only in a few guard examples, with `length` and `substring` from Topic 1.15.

TEACHER NOTE

The slides are the Branches on Branches deck (21 slides, with speaker notes); slide numbers below refer to it. Day 1 uses slides 1–12 and Day 2 uses slides 13–21. The Branches on Branches puzzle worksheet (10 questions, about 35 minutes) is split across the two nights: questions 2, 3, and 6 after Day 1; questions 1, 4, 5, 7, 8, 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:** `Parking`, `Shipping`, `PagesGuard`, `GuardWrongOrder`, `StringGuard`, `StringNoGuard`, `InnerElseMisread`, `InnerElseFixed`, `DanglingElse`, `DanglingElseFixed.java`.
- **Day 2 demonstrations:** `SpeedCheck`, `GradeChain`, `GradeWrongOrder`, `GradeLowFirst`, `SeparateIfs`, `HotMildChain`, `HotMildSeparate`, `ElseFirstBug.java`, `ElseInMiddle.java`, `ElseWithCondition.java`.
- **Printed only (exit ticket):** `ExitPool`, `ExitChainOrder`, and `ExitNoBraces`.

Day 1 warm-up: a question inside a question

Show slide 2, Hopper's day plan. Students answer in pairs: what does the plan say on a sunny Tuesday, and on a rainy Sunday? Take answers: school on the Tuesday, a movie on the Sunday. Then ask: on which days does the plan ask about the weather at all? Only on the weekend, because the weather question sits inside the weekend branch. Name it: a question that is asked only after an earlier question was answered.

Day 1 direct instruction: nested if and the guard

1. **Goals** (slide 3). Goals 1–3 are Day 1; goals 4 and 5 are Day 2.
2. **A nested if** (slide 5; 2.4.A.1). A nested `if` is an `if` inside the body of another `if` or `else`. Trace the library check: with `age 15`, the outer condition is true, so Java reaches `if (hasCard)`; with `age 10`, the whole outer body is skipped and `hasCard` is never looked at. Then `Welcome` is displayed either way, because it is outside both bodies.
3. **The guard** (slide 6; 2.4.A.1). An outer condition that makes an inner check safe is a **guard**. With `players 0`, `players > 0` is false, so Java never evaluates `points / players`. Trace it step by step with the table on the slide.
4. **Guard on the inside** (slide 7). Swap the two checks and the division is now the outer condition, so it runs first and throws an `ArithmeticException` (Topic 1.3). Nothing is displayed, not even `Done`.

MISCONCEPTION The guard can go inside the risky check

Students write both checks but put the safety check inside, so the risky one runs first. A guard only protects what is inside its body. The outer condition is always evaluated first, so

the check that can crash must be the inner one. Have students say, for each snippet, "Which condition does Java evaluate first?"

TEACHER NOTE

If computers are open, run `PagesGuard` and then `GuardWrongOrder`. Students see `Report finished` for one and an exception with no output for the other. `StringGuard` and `StringNoGuard` show the same idea with `substring`.

Day 1 direct instruction: nesting in else, the inner else, the dangling else

1. **Nesting in the else branch** (slide 9; 2.4.A.2). An `if` can sit in an `else` body as well. Trace `temp 40, raining true: temp >= 60` is false, so Java runs the `else` body and reaches `if (raining)`, which displays `Raincoat`. One path runs, with three possible results. Draw the decision tree from the slide on the board.
2. **An inner else answers only its own if** (slide 10; 2.4.A.2). With `age 14`, `age >= 16` is false, so the whole body is skipped, inner `else` included; only `Done` is displayed.
3. **Braces and the nearest if** (slide 11; 2.4.A.2). Without braces, an `else` pairs with the nearest earlier `if` that does not already have an `else`, whatever the indentation says. Compare the two columns for `n 3`: the left prints `not positive?`, because the `else` belongs to `if (n > 5)`; the braced version prints only `end` because `n > 0` is true and the inner condition is false.

MISCONCEPTION The inner else runs whenever the outer condition is false

Students read an inner `else` as "otherwise, for anything that failed." Each `else` answers exactly one `if`. The inner `else` runs only when the outer condition is true and the inner one is false. This is the Think First question on worksheet question 2.

MISCONCEPTION The indentation shows which if the else belongs to

Students trust the layout. The compiler never looks at indentation, and code with a dangling `else` compiles without any warning. Teach the routine: ignore indentation, find each `else`, and pair it with the nearest unpaired `if` above it; then add braces if you want a different pairing.

Day 1 guided practice: what prints for each player

Show slide 12 and give students two minutes with mini whiteboards, one row at a time: `hours 9` with `rested true` displays `Ready`; `9` with `false` displays `Nap first`; `6` with `true` displays `Sleep more`, because the inner `if-else` is skipped. For the third row ask: is `rested` ever checked? No. If time remains, students draw the decision tree on their whiteboard.

TEACHER NOTE

Hold up a whiteboard check after each row. If many students write `Nap` first for the third row, return to slide 10 and the inner `else`.

Day 1 independent practice and closure

Puzzle worksheet questions 2, 3, and 6 are homework: the ride sign, the code tiles that guard the division, and two truths and a lie about the `else` with no braces. Question 3 settles the guard-order idea from today.

For closure, write one sentence on the board: "An inner `else` belongs to ____." Students finish it with their partner: the nearest `if` without an `else`, which is not always the outer one.

Day 2 warm-up: nested or chained?

Show this on the board: a movie ticket costs one price for children and a different price for everyone else. Ask students whether they need a nested `if` or an `if-else`. Then add a third group, seniors. Take two answers on how the code changes. This introduces the `else if` chain.

Day 2 direct instruction: multiway selection, order, trailing `else`

1. **The first true condition wins** (slide 14; 2.4.A.3). A multiway selection is an `if`, `else if`, `else` chain. Trace `speed 72`: `72 > 80` is false, so Java goes on; `72 > 65` is true, so `Warning` is displayed and the rest is skipped. The `else` is never checked.
2. **Order matters** (slide 15). Reverse the first two conditions: `speed 90` is over 65, so it displays `Warning` and `Ticket` is unreachable, because every value over 80 is also over 65. Rule: put the more specific or narrower condition first, or order the tests so each one needs only a single comparison.
3. **The trailing `else`** (slide 16). With no trailing `else`, `coins 3` matches neither condition and nothing is displayed from the chain; the next line, `Coins: 3`, still runs. With the trailing `else`, `No chest` is displayed first.
4. **The plain `else` goes last** (slide 17). An `else` never has a condition, and nothing can follow it in the chain, so `else` followed by `else if` does not compile ('`else`' without '`if`').
5. **Vocabulary** (slide 20) can wait until closure.

MISCONCEPTION Every true condition runs its body

For `score 95`, students see that `>= 90`, `>= 80`, and `>= 70` are all true and predict several lines. Each later condition is reached only through an `else`, so only the first true one runs. This is the question in worksheet question 1 (answer A, not B).

MISCONCEPTION A row of separate ifs is the same as a chain

Leave out the word `else` and Java checks every `if`, so several bodies can run, or none. Show `SeparateIfs` next to `GradeChain`, or `HotMildSeparate` next to `HotMildChain`. Worksheet question 7 (one token: add `else`) and question 5 (the face-off) cover it.

MISCONCEPTION A plain else can go before an else if

Once an `if` has its plain `else`, the statement is complete, so another `else if` cannot attach. Students need to see the compiler message once, on slide 17 or with `ElseFirstBug.java`.

Day 2 guided practice: shipping tiers and write the chain

1. **Shipping tiers** (slide 18). Students write the value printed for each weight on mini whiteboards: 25 displays 25.0; 20 displays 12.0; 5 displays 6.0; 1 displays 6.0. Ask about 20 and 5: neither is above 20 or above 5 (the boundaries), so each falls to the next branch.
2. **Write the chain** (slide 19). Students write their own `if`, `else if`, `else` chain that prints Freezing, Cold, Mild, or Hot for an `int temp`. A model: `if (temp < 0) Freezing, else if (temp <= 15) Cold, else if (temp <= 25) Mild, else Hot`. Follow the hint on the slide: each condition needs one comparison because the order does the work. Have two students share different valid orders (for example, starting from `temp > 25`).

TEACHER NOTE

Check the boundaries, 0, 15, and 25, with the class. They are the usual off-by-one errors, and each lands in a different branch.

Day 2 independent practice

Puzzle worksheet questions 1, 4, 5, 7, 8, 9, and 10 are homework: the points chain, the grade snippet pool, the chain versus separate `if` face-off, the one-token fix (add `else`), who am I, you're the compiler (snippets 2 and 3 return to the guard order), and the wrap-up sentence.

Day 2 closure and exit ticket

Show slide 20 (vocabulary), then slide 21 (closure) and have students finish the sentence starter with a partner. Give the exit ticket (about five minutes). Question 1 checks tracing a nested `if-else` (2.4.A.1, 2.4.A.2), question 2 checks that only the first true branch of a chain runs, in order (2.4.A.3), and question 3 checks pairing an `else` without braces (2.4.A.2). Re-teach the weakest one at the start of Topic 2.5.