

2.3 if Statements

Lesson Plan

Pacing: 2 class periods of 45 minutes

Learning objectives: 2.3.A

Essential knowledge: 2.3.A.1, 2.3.A.2, 2.3.A.3, 2.3.A.4

Materials

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

Agenda

Minutes	Activity
5	Day 1 warm-up — every line, every time? (slides 2–3)
12	Day 1 direct instruction — the one-way if (slides 4–7)
13	Day 1 direct instruction — the condition and the body (slides 8–10)
10	Day 1 guided practice — step challenge (slide 11)
5	Day 1 closure — which lines run?
5	Day 2 warm-up — fork in the road
12	Day 2 direct instruction — if-else (slides 12–14)
6	Day 2 guided practice — parking garage (slide 15)
10	Day 2 direct instruction — one-way or two-way? (slides 16–18)
6	Day 2 guided practice — does the shop open? (slide 19)
6	Day 2 closure and exit ticket (slides 20–21)

Overview

Day 1 is the one-way `if` statement: how it changes the top-to-bottom order statements run in (the **flow of control**), how to trace whether its body runs, and how to write the condition and body, with braces around a body of several statements (2.3.A.1, 2.3.A.2). Day 2 is the two-way `if-else` statement, where exactly one of two bodies runs (2.3.A.3), and choosing between one-way and two-way selection, including how separate `if` statements can affect each other (2.3.A.4).

Students know assignment and compound operators (Topics 1.4 and 1.6), `equals` (Topic 1.15), selection as an algorithm idea (Topic 2.1), and Boolean expressions (Topic 2.2). Stay inside the topic boundary. Conditions are single Boolean expressions; do not use `else if`, nested `if`

statements (Topic 2.4), or `&&`, `||`, and `!` (Topic 2.5 and beyond). If a student asks about checking more than two cases, tell them that is coming in Topic 2.4.

TEACHER NOTE

The slides are the Fork in the Code deck (21 slides, with speaker notes); slide numbers below refer to it. Day 1 uses slides 1–11 and Day 2 uses slides 12–21. The puzzle worksheet (10 questions, about 35 minutes) is split: questions 1, 2, 7, and 8 (rows 1, 2, and 5) after Day 1; questions 3, 4, 5, 6, 9, and 10 and the rest of question 8 after Day 2.

Files ending in `.snippet.java` hold only the statements that go inside `main`; paste them into a scratch class to run them. Examples are in the topic's `examples/` folder.

- **Day 1 demonstrations:** `NoSelection`, `FreeShipping`, `PaidShipping`, `JacketCold`, `JacketWarm`, `JacketFifty`, `SpeedCheck`, `StraySemicolon`, `NoParentheses`, `AssignInCondition`, `CountCompare`, `MemberEqualsTrue`, `MemberDiscount`, `LivesBug`, `MissingBracesFixed`.
- **Day 2 demonstrations:** `PassFail`, `PassFailSeventy`, `PassBoundaryBug`, `TicketPrice`, `ParityCheck`, `DayCheck`, `ElseWithCondition`, `ElseWithoutIf`, `TwoIfsOverlap`, `TwoIfsOtherScores`, `Restock`, `ThanksBoth`, `ThanksAfter`, `UnassignedAfterIf`, `UnassignedFixedInit`, `AssignedBothBranches`.
- **Exit ticket:** reuses `TwoIfsBalance`, `MissingBraces`, `SegmentI`, and `SegmentII`. Keep these out of class so the questions are new to students.

Day 1 warm-up: every line, every time?

Show slide 2: shipping should be free for orders of \$50 or more, but every statement runs once, top to bottom, so the program always prints `Shipping: 7.99`. Partners decide which line should happen only sometimes and how Java could know when. Run `NoSelection` as a second example if you like: both lines print for a temperature of 68, even though the jacket line should not. Show the goals (slide 3).

Day 1 direct instruction: the one-way if

1. **Run the body only when the condition is true** (2.3.A.1, slide 5). A **selection statement** changes the **flow of control**, the order statements run in, based on a Boolean expression. Break `if (total >= 50) { shipping = 0.0; }` into three parts: the **condition** in parentheses, the **body** in braces, and the rest of the program after the closing brace, which runs either way.
2. **A true condition.** Run `FreeShipping`: `Free shipping!`, then `Total: 62.5`. The body ran and changed shipping.
3. **A false condition** (slide 6). Run `PaidShipping`: the body is skipped, and the output is `Total: 47.5`. The flowchart has one arrow that skips the body and one that goes through it, and both rejoin.
4. **Boundaries.** `JacketCold` (45), `JacketWarm` (68), and `JacketFifty` (50) show that `temperature < 50` is false at exactly 50. Students predict each before the run.

5. **A semicolon after the condition** (slide 7). `StraySemicolon` prints `Wear a jacket.` for 68. The `;` ends the `if` statement, so the braces after it are an ordinary block that always runs. The code compiles, so only the wrong output shows the bug.

MISCONCEPTION A semicolon after the condition

Most Java lines end with `;`, so students add one after `if (...)`. It becomes an empty body, and the braces that follow always run. Give students a rule: no semicolon after the closing parenthesis of an `if`.

MISCONCEPTION The body always runs once

Students expect a statement to run once, as in Unit 1. With an `if`, the body runs once or not at all. Ask "Does the body run for this value?" before every trace.

Day 1 direct instruction: the condition and the body

1. **The condition must be a boolean** (slide 9). Three kinds of condition: a `boolean` variable (`if (raining)`), a method call that returns a `boolean` (`day.equals("Sat")`), and a comparison (`score == 100`). Run `MemberDiscount` for two separate conditions, one a `boolean` variable and one `equals`.
2. **Parentheses are required.** `NoParentheses` does not compile (`'('` expected).
3. **One = is a mistake here too.** `AssignInCondition` and `CountCompare` do not compile:
`incompatible types: int cannot be converted to boolean`. Remind students of Topic 2.2: `=` stores, `==` asks. `MemberEqualsTrue` compiles but `isMember == true` is the same as `isMember`.
4. **Braces for several statements** (slide 10). With no braces, the body is only the next statement; indentation means nothing to Java. Show the slide's `lives` example. Then show `LivesBug`: `Ouch!` prints even though `hitByEnemy` is `false`, because only `lives--;` was the body. `MissingBracesFixed` shows the fix.

MISCONCEPTION Indenting a line puts it in the body

Java ignores indentation. Without braces the body is exactly one statement. Teach the habit: always write braces, even for one statement.

MISCONCEPTION Using = instead of == in the condition

`if (score = 100)` does not compile because `score = 100` is an `int`, not a `boolean`. Read `==` aloud as "is equal to".

MISCONCEPTION Comparing Strings with ==

Use `day.equals("Sat")`, not `day == "Sat"` (Topic 2.2). `equals` is case-sensitive: `"sat"` is not `"Sat"`.

Day 1 guided practice: step challenge

Slide 11. Students trace the `points` code for `steps` of 12000, 10000, and 9999 and write what prints: 55, 55, 5. The boundary case 10000 is the one to discuss: `steps >= 10000` is true. Then run `SpeedCheck` with `speed = 30` and `limit = 35` (prints only `Speed: 30`); students change `speed` to 36 and predict first.

TEACHER NOTE

If you have no mini whiteboards, students write answers on scrap paper and show a partner. Keep each trace to about a minute, and have students say "true, so the body runs" or "false, so the body is skipped" aloud.

Day 1 independent practice

Puzzle worksheet questions 1, 2, and 7 (be the JVM with a one-way `if`, a table of fees for five ages, and a one-token fix to a stray semicolon) are homework. Question 2 previews Day 2 by putting two `if` statements one after the other. Question 1's Think First (can two separate `if` statements skip both bodies?) is answered by question 2: yes, for age 13 to 64.

Day 1 closure

Ask: "In `if (total >= 50) { shipping = 0.0; }` which part is the condition, which is the body, and what runs after?" Students answer on a sticky note.

Day 2 warm-up: fork in the road

Write on the board: "If you have a ticket, go in. Otherwise, buy one." Ask how many of the two actions happen every time (exactly one). Introduce today's idea: an `if` can have an `else`, so the program picks one path.

Day 2 direct instruction: if-else

1. **Exactly one body runs** (2.3.A.3, slide 13). `if (guess == secret) { ... } else { ... }`. The condition is checked once. If it is true the first body runs; otherwise the `else` body runs. Both paths rejoin afterward, so `Thanks for playing` always prints.
2. **Pass or fail**. Run `PassFail` (64: Try again next time., Score: 64), then `PassFailSeventy` (70) and `PassBoundaryBug` (`score > 70` for 70 is a boundary bug that gives Try again next time.).
3. **A value from either path**. `TicketPrice` sets `price` in both branches. `ParityCheck` and `DayCheck` ("`sat`" is not "`Sat`", so `Weekday`) are short alternatives.
4. **else has no condition** (slide 14). `ElseWithCondition` does not compile (not a statement): the `else` body runs exactly when the `if` condition is false, so the question has already been asked. `ElseWithoutIf`, the stray semicolon again, gives 'else' without 'if'.

MISCONCEPTION else needs a condition

Students write `else (age < 16)`. `else` runs when the `if` condition is false and has no condition of its own (`ElseWithCondition`).

MISCONCEPTION A semicolon after the condition, now with an else

The `;` ends the `if`, so the `else` has nothing to attach to: `'else' without 'if'` (`ElseWithoutIf`). Delete the semicolon.

MISCONCEPTION Both bodies of an if-else can run

Exactly one runs, never both and never neither. Ask students to name which body ran before they trace anything after it.

Day 2 guided practice: parking garage

Slide 15. Students write the value of `cost` for 1, 2, and 5 hours: 0, 0, 15. Discuss the boundary (2 hours is free) and why `cost` has a value on both paths. Then show `ThanksBoth` and `ThanksAfter`: both print the same thing for `total = 18.0`, but the second writes `Thanks for shopping!` only once, after the `if-else`. Ask: "Which version is easier to change?"

Day 2 direct instruction: one-way or two-way?

1. **Separate ifs are independent** (2.3.A.4, slide 17). Run `TwoIfsOverlap` (93): `Excellent work!` and `You passed.`, both bodies run because both conditions are true. `TwoIfsOtherScores` shows 80 (only `You passed.`) and 50 (nothing). The same logic as an `if-else` runs exactly one body, so 95 prints `Honor roll` only.
2. **One if can change what the next one sees**. Run `Restock`: the first `if` raises `stock`, so the second one's condition is now true: `Restocked.`, `In stock.`, `Stock: 14`.
3. **A value on only one path** (slide 18). `UnassignedAfterIf` does not compile (`variable message might not have been initialized`); the compiler checks the path where the body is skipped. Two fixes: give the variable a starting value (`UnassignedFixedInit`) or assign it in both branches (`AssignedBothBranches`).

MISCONCEPTION Two ifs in a row are the same as an if-else

Separate `if` statements can both run, or neither. An `if-else` always runs exactly one body. The shape of the problem decides: "choose one of two" is an `if-else`; "check this, and also check that" is two `if` statements.

MISCONCEPTION The second if only sees the original values

Each condition is checked when its statement runs, using the current values. Restock shows the first `if` changing what the second one sees.

MISCONCEPTION A variable assigned in one branch is always initialized

Java checks every path. If a path skips the assignment, using the variable afterward is a compile-time error.

Day 2 guided practice: does the shop open?

Slide 19. Students write the output of version A (two separate `if` statements) and version B (`if-else`) and explain why they differ. Version A: `coins` becomes 13, the second `if` is now true, so Shop open and 13. Version B prints only 13: `coins < 10` was true, so the first body ran and the `else` body was skipped, so Shop open never prints. Close by asking students to describe, in one sentence, what an `if-else` guarantees.

Day 2 independent practice

Puzzle worksheet questions 3, 4, 5, 6, 8 (the remaining rows), 9, and 10 are homework: rebuild a ticket booth with code tiles (question 3), fill the snippet pool (question 4), match four variants to their output (question 5), two truths and a lie about `if-else` (question 6), and the reflection on when to use two separate `if` statements (question 10).

Day 2 closure and exit ticket

Show slide 21; partners finish the sentence starter. Give the exit ticket (about five minutes). Question 1 checks tracing two separate `if` statements (2.3.A.1, 2.3.A.4), question 2 checks the body of an `if` without braces (2.3.A.2), and question 3 checks `if` versus `if-else` (2.3.A.3, 2.3.A.4). Re-teach the weakest one at the start of Topic 2.4. Slide 20 (vocabulary) is a reference for students.