

2.9 Implementing Selection and Iteration Algorithms

Lesson Plan

Pacing: 3 class periods of 45 minutes

Learning objectives: 2.9.A

Essential knowledge: 2.9.A.1

Materials

- Projector and the Loop Toolbox slide deck
- Whiteboard and markers
- Mini whiteboards or scrap paper (trace tables)
- IDE on every computer (Days 1 and 3)
- Printed Loop Toolbox puzzle worksheet (answer key for you) and exit ticket

Agenda

Minutes	Activity
5	Day 1 warm-up — one receipt at a time (slide 2)
15	Day 1 direct instruction — the accumulator and the average (slides 3–7)
10	Day 1 direct instruction — counting with an if inside a loop (slides 8–9)
10	Day 1 guided practice — write a total and a count in the IDE, then worksheet question 2
5	Day 1 closure — where does the accumulator start?
5	Day 2 warm-up — worksheet question 1, first rows
10	Day 2 direct instruction — divisibility with % (slides 10–11)
5	Day 2 guided practice — count with two tests (slide 12)
15	Day 2 direct instruction — taking an integer apart (slides 14–16)
7	Day 2 independent practice — worksheet question 3
3	Day 2 closure — which operator?
5	Day 3 warm-up — is 0 the lowest?
12	Day 3 direct instruction — keep the best so far (slides 17–19)
8	Day 3 guided practice — two jobs, one loop (slide 20)
10	Day 3 independent practice — worksheet questions 4 and 8
5	Day 3 closure — pick the right tool (slides 21–22)
5	Day 3 exit ticket

Overview

This topic has one essential knowledge statement (2.9.A.1) and the day split follows its bullet list of standard algorithms. Day 1 is the accumulator (a sum that grows each pass), the average (a sum divided by a count, with the decimals kept), and the frequency counter (an `if` inside a loop that adds 1 when a condition is true). Day 2 is divisibility (`a % b == 0`) and taking an integer apart one digit at a time with `% 10` and `/ 10`. Day 3 is the minimum and maximum: a best-so-far variable that starts with a value the data will replace, and a comparison that updates it. Students also combine two algorithms in one loop.

Students have written `while` and `for` loops (Topics 2.7 and 2.8), `if` statements and compound conditions (Topics 2.3 to 2.5), integer division and the remainder operator (Topic 1.3), and casting (Topic 1.5). Stay inside the topic boundary: no strings (Topic 2.10), no nested loops (Topic 2.11), no run-time analysis (Topic 2.12), and no arrays or methods that return values. The CED does not ask for negative operands with `%`, so every algorithm today assumes numbers that are 0 or positive and a positive divisor.

TEACHER NOTE

The slides are the Loop Toolbox deck (22 slides, with speaker notes); slide numbers below refer to it. Day 1 uses slides 1–9, Day 2 uses slides 10–16, and Day 3 uses slides 17–22. The Loop Toolbox puzzle worksheet (10 questions, about 35 minutes) is split across the three days: question 2 in class on Day 1, questions 1 and 3 on Day 2, questions 4 and 8 in class on Day 3, and questions 5, 6, 7, 9, and 10 as homework (6 and 7 after Day 2, the rest after Day 3).

Files whose names end in `.snippet.java` hold only the statements that go inside `main`; paste them into the `main` of a scratch class to run them. The other files are full programs that read a `Scanner` over a `String`, so they run without typing. All files are in the topic's `examples/` folder.

- **Day 1 demonstrations:** `SumOneToFive`, `ReceiptTotal.java`, `QuizAverage.java`, `AverageCastPlacement`, `AverageChoices`, `QuizAverageEmpty.java`, `AverageNoGuard.java` (run-time error), `SumResetInside`, `HotDays.java`, `FreezeCount.java`, `EvenCountBug`, `EvenCountFixed`.
- **Day 2 demonstrations:** `DivisorsOfTwelve`, `DivisibleBackwards`, `BothDivisorsBug`, `BothDivisorsFixed`, `CommonMultiples`, `DigitPieces`, `DigitsRightToLeft`, `DigitSum`, `DigitCountZero`, `DigitCountZeroFixed`, `ZeroDigits`.
- **Day 3 demonstrations:** `HighestScore.java`, `MaxStartFirst.java`, `MaxStartZero.java`, `MinStartZero.java`, `ColdestDay.java`, `TieFirstDay.java`, `TieLastDay.java`, `DigitStats`, `MixedLoop`.
- **Printed only (exit ticket):** `AlgoExitTrace` and `AlgoExitLowest.java`.

Day 1 warm-up: one receipt at a time

Show slide 2: four receipts (\$4, \$7, \$2, \$5) appear one at a time. Students write the total, the cheapest, and how many numbers they had to hold in their head. Most keep one running total (and one "cheapest so far") instead of remembering every receipt; that is exactly what a loop variable does. Leave "keep one running value" on the board.

Day 1 direct instruction: the accumulator and the average

1. **Goals** (slide 3). Read the five goals; Day 1 covers goals 1 and 2.
2. **The running total** (2.9.A.1, slide 5). An **accumulator** collects a result a little at a time. Three parts: start it at 0 once, before the loop; add one value inside the loop with `+=`; use the finished total after the loop. Run `SumOneToFive` (15) and the slide's odd numbers 1 to 9 (25). `ReceiptTotal.java` uses a `double` accumulator started at `0.0` and prints `Total: 11.5`.
3. **The average** (slide 6). $\text{Average} = \text{sum} \div \text{number of values}$. Show the quiz scores 88 92 75 90 81 (sum 426): `sum / 5` prints 85, `(double) (sum / 5)` prints 85.0, and `(double) sum / 5` prints 85.2. Same sum, three answers. Run `AverageCastPlacement` for a second data set.
4. **Cast before you divide** (slide 7). The cast applies to the value right next to it. Put it on `sum`, so the division is already a `double` division. Parentheses around the whole division cast too late, after the decimal part is gone (Topic 1.5).
5. **No values**. If the count can be 0, check `count > 0` before dividing. Run `AverageNoGuard.java`: it throws `ArithmeticException (/ by zero)`. `QuizAverageEmpty.java` guards the division and prints `No scores`.

MISCONCEPTION Resetting the accumulator inside the loop

Students declare `int total = 0;` in the body, so it restarts at 0 on every pass and the output is the last value, not the sum (`SumResetInside` prints 5, not 15). The accumulator starts once, before the loop, and is printed once, after it.

MISCONCEPTION Casting the answer instead of an operand

Students write `(double) (sum / count)` and are surprised to see 85.0 instead of 85.2. The integer division has already dropped the decimals before the cast runs. Cast one operand: `(double) sum / count`.

MISCONCEPTION Dividing by a count that can be zero

An empty data set gives a count of 0 and an `ArithmeticException` when both operands are `int`. Ask "What if there is nothing to average?" for every average students write.

Day 1 direct instruction: counting with an if inside a loop

1. **Frequency** (2.9.A.1, slide 9). To count how often a condition is true, start a counter at 0 before the loop and add 1 only when an `if` inside the loop finds the condition true. The slide's loop counts the multiples of 4 from 1 to 30 and prints 7. A counter is an accumulator that adds 1.
2. **Run** `FreezeCount.java` and `HotDays.java` and have students predict the count before each run.
3. **A near-miss test**. Show `EvenCountBug` (`n / 2 == 0` counts only the value 1, so it prints 1) and ask what operator the test needs. Then `EvenCountFixed`. Even means `n % 2 == 0`.

MISCONCEPTION Using / when you mean %

Students write `n / 2 == 0` to test for even. The `/` operator gives the quotient; only `%` gives the remainder. `EvenCountBug` prints 1 for the numbers 1 to 10.

Day 1 guided practice: write a total and a count

In the IDE, students write a loop that adds the numbers from 1 to 10 and, in the same loop, counts how many are multiples of 3 (the totals are 55 and 3). They run it and compare with a partner. Then they do worksheet question 2 (you're the JVM): `sum` is $3 + 6 + 9 + 12 = 30$ and `count` is 4, so the output is 7 7.5, choice A. The question's "Watch out" box (the accumulator declared inside the loop) is a good example to reread aloud, and its "Fair question" box repeats the empty-data guard.

Day 1 closure

Ask: "Where does the accumulator start, and how often?" Students write: at 0, once, before the loop. Then ask which of the three parts (start, add, use) goes after the loop.

Day 2 warm-up: first rows of question 1

Put `n = 5308` on the board. Students fill in the first three rows of worksheet question 1 on a sticky note: `n % 10` is 8, `n / 10` is 530, `n / 10 % 10` is 0. Ask which one is the last digit.

Day 2 direct instruction: divisibility with %

1. **The test** (2.9.A.1, slide 10). `a % b == 0` is true exactly when `a` is evenly divisible by `b`. Even is `n % 2 == 0`. Run `DivisorsOfTwelve` (output 1 2 3 4 6 12, then `Divisors: 6`) and walk the table for 18 on slide 10: `d` from 1 to 18; the divisors are 1, 2, 3, 6, 9, 18.
2. **Which number goes first?** (slide 11). Read `a % b == 0` as "a divided by b leaves nothing." `96 % 8` is 0 but `8 % 96` is 8. Run `DivisibleBackwards` (prints 7 and `Some left over`, because `7 % 84` and `84 % 7` are swapped).
3. **Two tests at once.** `BothDivisorsBug` and `BothDivisorsFixed` and `CommonMultiples` show compound conditions from Topic 2.5 inside a counting loop: `||` counts numbers divisible by either 2 or 3, while `&&` is needed for "divisible by both".

MISCONCEPTION Swapping the operands of %

Students write `friends % marbles` and expect "does friends divide marbles?" The number being shared is on the left. Have them say the test aloud: "marbles divided by friends leaves nothing."

Day 2 guided practice: count with two tests

Slide 12: count the numbers from 1 to 50 that are divisible by 5 but not by 10. Students list the numbers (5, 15, 25, 35, 45) and write the count, 5. Ask which condition removes 10, 20, 30, 40, and 50 (`n % 10 != 0`).

Day 2 direct instruction: taking an integer apart

1. **The two moves** (slide 14). `number % 10` is the last digit; `number / 10` drops it. For 731: `% 10` gives 1, `/ 10` gives 73, `/ 10 % 10` gives 3 (the tens digit). Run `DigitPieces`.
2. **Visit every digit** (slide 15). Loop while `n > 0`: take the last digit with `% 10`, then drop it with `n /= 10`. For 2581 the digits come out right to left: 1, 8, 5, 2 (prints 1852) and the loop counts 4 digits. Run `DigitsRightToLeft` (4072 prints 2, 7, 0, 4) and `DigitSum`. Have students fill in the slide's table with you: `n` is 2581, 258, 25, 2; then 0 ends the loop.
3. **Zero as an edge case** (slide 16). With `n` starting at 0 the loop never runs, so it says 0 digits. Run `DigitCountZero`, then `DigitCountZeroFixed` (an `if` before the loop). Always test the edges: 0, one digit, and a number with zeros inside, like `ZeroDigits` (70300 has 3 zero digits).

TEACHER NOTE

All digit work assumes a positive number. If a student asks about negative numbers, say that the sign rules for `%` are outside this course and move on.

MISCONCEPTION Dropping the last digit with `%`

Students write `n %= 10` where they need `n /= 10`. After `n %= 10`, `n` holds only the last digit, so the loop sees one digit and then stops. This is worksheet question 9. Use `%` to read a digit and `/` to remove it.

MISCONCEPTION Zero has no digits

The `while (n > 0)` loop treats 0 as having no digits and prints 0. Ask students to test their digit loop with 0 before they call it finished.

Day 2 independent practice

Students do worksheet question 3 (code tiles: digit sum of 4096, answer 19, with `sum += n % 10;` and `n /= 10;` in a `while (n > 0)` loop; the decoy uses `n / 10` alone. Questions 6 (face-off) and 7 (match-up: 6, 14, 3, and 13) are homework.

TEACHER NOTE

Question 7: the multiples of 3 up to 20 are 3, 6, 9, 12, 15, 18 (6 of them). `n % 3 != 0` gives 14. Both 3 and 2 gives 3 numbers (6, 12, 18). Either gives $6 + 10 - 3 = 13$.

Day 2 closure

Ask which operator students would use for each job: "the last digit" (%), "drop the last digit" (/), "is it divisible?" (%).

Day 3 warm-up: is 0 the lowest?

Show slide 19's problem without the code: three prices are 12, 5, and 9, and the program says the cheapest is 0. Students write why. Take two answers, then say the lesson: the starting value must not be a value the data cannot beat.

Day 3 direct instruction: keep the best so far

1. **The pattern** (2.9.A.1, slide 18). Start the best-so-far variable with the first value, or with a value every real value beats; then compare each new value and replace it when the new value is better. Maximum uses `>`, minimum uses `<`. Run `HighestScore.java` (Highest: 96).
2. **Safe starts**. `Integer.MIN_VALUE` for a maximum and `Integer.MAX_VALUE` for a minimum are values nothing can beat. Reading the first value as the start means the loop begins at the second one (`MaxStartFirst.java`).
3. **Starting at 0** (slide 19). The slide prints Cheapest: 0 because no price is below 0. Run `MinStartZero.java` and `MaxStartZero.java`: the warmest of four negative temperatures prints 0, a number that was never in the data.
4. **Ties**. Run `TieFirstDay.java` and `TieLastDay.java`: `>` keeps the first day with the best value, `>=` moves to the last. Ask which is wanted.

MISCONCEPTION Starting the best-so-far at 0

Students start a minimum or maximum at 0 because accumulators start at 0. If every value is above 0, the minimum stays 0; if every value is below 0, the maximum stays 0. Use the first value or `Integer.MAX_VALUE` / `Integer.MIN_VALUE`.

MISCONCEPTION Using the wrong comparison

Students write `<` when searching for the maximum. A maximum replaces the best when the new value is larger; a minimum when it is smaller. Have students trace three values to see which way the comparison needs to point.

Day 3 guided practice: two jobs, one loop

Slide 20: `v` is `k * 5 % 8` for `k` from 1 to 4. Students fill in the table (`v` is 5, 2, 7, 4; `total` is 5, 7, 14, 18; `most` is 5, 5, 7, 7) and write the output, 7 18. Two variables with different starts (`Integer.MIN_VALUE` and 0) work in the same loop because each has its own job. `DigitStats` and `MixedLoop` show more of this.

Day 3 independent practice

Students do worksheet question 4 (snippet pool: find the lowest of five temperatures; two fills work: `Integer.MAX_VALUE, 1, temp < lowest, or console.nextInt(), 2, temp < lowest`) and question 8 (a trace table for a counter and a maximum: output `3 6`) alone. Question 5 (the lie is claim 2: the code prints 0, not -3), question 9, and question 10 are homework.

Day 3 closure

Slides 21 and 22. Review the vocabulary together. Then partners answer the closure prompt: a game wants the total points, the number of perfect rounds, and the best round. Each partner says how every variable starts and changes: the total starts at 0 and adds each round; the perfect-round counter starts at 0 and adds 1 only when the round is perfect; the best round starts at the first round (or `Integer.MIN_VALUE`) and changes when a larger one appears.

Day 3 exit ticket

Give the exit ticket for the last 5 minutes. Collect it as students leave.