

# 1.2 Variables and Data Types

## Lesson Plan

**Pacing:** 2 class periods of 45 minutes

**Learning objectives:** 1.2.A, 1.2.B

**Essential knowledge:** 1.2.A.1, 1.2.A.2, 1.2.A.3, 1.2.B.1, 1.2.B.2

### Materials

- Projector and the Pick the Right Box slide deck
- Whiteboard and markers
- IDE on every computer (Day 2)
- Printed Pick the Right Box puzzle worksheet (answer key for you) and exit ticket

## Agenda

| Minutes | Activity                                                                                           |
|---------|----------------------------------------------------------------------------------------------------|
| 5       | Day 1 warm-up — what fits in each box? (slide 2)                                                   |
| 18      | Day 1 direct instruction — data types, primitive and reference, int, double, boolean (slides 3–10) |
| 14      | Day 1 guided practice — naming and choosing the most appropriate type (slides 11, 18–19)           |
| 5       | Day 1 independent practice — puzzle worksheet question 1                                           |
| 3       | Day 1 closure — what decides a type?                                                               |
| 5       | Day 2 warm-up — name the type of each value                                                        |
| 12      | Day 2 direct instruction — declaring a variable, naming rules (slides 12–14)                       |
| 8       | Day 2 direct instruction — a variable's value can change (slide 15)                                |
| 7       | Day 2 direct instruction — a variable holds only values of its type (slide 16)                     |
| 8       | Day 2 guided practice — break GameStats.java on purpose in the IDE                                 |
| 5       | Day 2 closure — exit ticket                                                                        |

## Overview

Day 1 is about data types: what a data type is (1.2.A.1), the two categories, primitive and reference (1.2.A.1–1.2.A.3), the three primitive types in this course (1.2.A.2, 1.2.B.1), and how to choose the most appropriate type for a piece of information (1.2.A). Day 2 is about variables: declaring them, changing their values, and the rule that a variable holds only values of its own type (1.2.B.2).

One running example ties both days together: a video game that must remember a coin count, a speed, and whether it is paused. The study guide and the slides use the same game, so students will recognize the values `1250`, `4.75`, and `false`.

The slides are the Pick the Right Box deck (21 slides, with speaker notes); slide numbers below refer to it. The deck runs data types, then variables, then choosing a type; this plan teaches choosing a type on Day 1, so Day 1 uses slides 1–11 and 18–19 and Day 2 uses slides 12–16 and 20–

21. The Pick the Right Box puzzle worksheet (10 questions, about 30 minutes) is mostly about variables: question 1 needs only Day 1 material, and questions 2–10 need Day 2.

Stay inside the topic boundary. Students have seen `System.out.println` (Topic 1.1) but not arithmetic, output rules, or assignment rules. When `=` changes a value, say that it stores the value on its right in the variable on its left, and that its full rules are Topic 1.4. Do not show `+` working on numbers, storing a whole number in a `double` variable (Topic 1.5), or declaring `String` variables. `String` appears only as the name of a type.

#### TEACHER NOTE

Before Day 1, have `SevenTwoWays` and `BooleanPlus` ready to run from the topic's examples. Before Day 2, create `GameStats.java` in the IDE and have `MissingDeclaration`, `CoinCount`, `Redeclare`, `DecimalInInt`, `NumberInBoolean`, and `CapitalTrue` ready. 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 exit ticket uses `Tickets` from the same folder; it is printed, not demonstrated.

## Day 1 warm-up: what fits in each box?

Show slide 2, a concert ticket form with four boxes: number of tickets, price paid, an "I agree" checkbox, and a name. Ask pairs what answers each box can accept and what you could do with each answer. Push with the slide's questions: Could a ticket count be 2.5? Could the checkbox be 17? What would it mean to add two names?

Collect answers on the board in four columns. Students usually describe whole numbers, numbers with decimals, yes/no answers, and text. Do not name the types yet; tell students the groups they just made are what Java calls data types.

## Day 1 direct instruction: data types

1. **Goals** (slide 3). Read the five goals aloud; goals 1–3 and 5 are today's, goal 4 (declaring and tracing a variable) is Day 2's.
2. **A data type has two halves** (1.2.A.1, slide 5). A data type is a set of values and the operations that can be done on those values. Use the game table: the coin count and the speed share arithmetic but have different values (4.75 belongs only to the speed); "paused" has two values and no arithmetic.
3. **Values alone are not enough**. A student ID such as 40213 looks like a coin count, but adding two IDs or averaging them is meaningless. Two kinds of information can have look-alike values and still be used in completely different ways.
4. **Primitive or reference** (1.2.A.1–1.2.A.3, slide 6). Primitive types are built into Java and hold simple values: numbers and true/false. A reference type is any type that is not primitive; reference types are used to define objects, which students meet in Topic 1.12. The common reference type is `String`, for text. Point out the clue: the primitive types are lowercase Java words; `String` starts with a capital letter because it is named after a class.

5. **The three primitive types** (1.2.A.2, 1.2.B.1, slide 8). `int` for integers, `double` for real numbers written with a decimal point, `boolean` for `true` or `false` (lowercase, no quotation marks). Java has five more primitive types (`long`, `short`, `byte`, `float`, `char`) that are not part of the course or the exam.
6. **Predict: SevenTwoWays**. Ask what `System.out.println(7);` and `System.out.println(7.0);` display. Run it: `7` and `7.0`. The decimal point, not the size of the fraction, decides the type. Slide 9 shows the same idea with `12`, `12.0`, and `true`.
7. **true is not 1, and it is not True** (slide 10). Show the two compiler messages on the slide; students meet them again in the IDE on Day 2.
8. **Predict: BooleanPlus**. Ask whether `System.out.println(true + false);` compiles. Run it and read the message: `+` (which means "add," Topic 1.3) cannot work on two `boolean` operands, because addition is not one of `boolean`'s operations. Define **operand**: a value an operator works on.

**MISCONCEPTION** A data type is just a list of allowed values

Students remember the values and forget the operations. Ask, for every type you introduce, "What can you do with it?" The student-ID example is the clearest contrast: look-alike values, but adding two IDs is meaningless.

**MISCONCEPTION** Short text is simple, so it must be primitive

Students think a one-letter `String` such as `"A"` or a short name is "simple" and therefore primitive. Give them one test: is the value a number or `true/false`? If not, it is not primitive in this course. Size and how often a type is used have nothing to do with the category.

**MISCONCEPTION** `true` and `false` add like 1 and 0

Students who have seen `true/false` as 1/0 expect `true + false` to work, or expect 1 to mean `true`. Run `BooleanPlus` so they see the compiler refuse, point to the first message on slide 10, and say plainly that in Java 1 is an `int`, not `true`.

## Day 1 guided practice: naming and choosing the most appropriate type

1. **Name that type** (slide 11). Pairs name the type of each of the six values (`-40`, `98.6`, `false`, `"98.6"`, `0.0`, `100`) and say how they can tell. Check `"98.6"` (quotation marks make it a `String`) and `0.0` (a decimal point makes it a `double`).
2. **The four questions** (slide 18). Teach them, asked in order: Is it yes or no? (`boolean`) Is it a number that can only ever be whole, such as a count? (`int`) Is it a number that can have a fractional part, such as a measurement or price? (`double`) Is it text, or anything else that is not a number or `true/false`? (a reference type, such as `String`) Show the left side first and have pairs choose a type for each school-app item, then reveal the right side and discuss any disagreements.

3. **Pick the right box** (slide 19). Pairs choose a type for each of the five pieces of information and write one reason for each. Save the slide's "write one declaration" step for Day 2.
4. **Change one thing: the temperature.** A thermostat that only shows and sets whole degrees can use `int`; a weather station that measures 68.5° needs `double`. The same kind of information can need a different type, because what decides the type is every value the program must store.
5. **Why not `double` for every number?** A type's set of values is a safety net. If the number of students is an `int`, the compiler stops anyone from storing 24.5 students. A well-chosen type makes impossible values impossible.

**MISCONCEPTION** Choosing the type from one example value

A student sees that today's hoodie costs \$25 and picks `int`. Tomorrow it costs \$24.99, which an `int` variable cannot hold. Train the question "What values could this ever have?" instead of "What is the value right now?"

## Day 1 independent practice

Hand out the puzzle worksheet. Students work Part A question 1: write the type of each of six values. Circulate and look especially at the last value: students often call `"true"` in quotation marks a `boolean`. Ask them what the quotation marks do.

**TEACHER NOTE**

Only question 1 needs just Day 1 material; the rest of the worksheet uses variables. Students keep it for Day 2 homework.

## Day 1 closure

Ask each student to write one sentence: "To choose a data type, I ask ..." Look for answers about the set of values the information could ever have (and, from strong students, what operations it needs).

## Day 2 warm-up: name the type

Show four values: `12`, `12.0`, `false`, `"12"`. Students write the type of each and whether it is primitive or reference. Answers: `int`, `double`, `boolean`, `String`. The last one is in quotation marks, so it is text, and `String` is a reference type. This also re-teaches the most common Day 1 slip, `12` versus `12.0`.

## Day 2 direct instruction: declaring a variable

1. **Why variables.** A game cannot write `1250` everywhere; it must remember the coin count and update it. Draw a labeled box on the board: the label is the name, the box is made for one type, and it holds exactly one value.
2. **Definition** (1.2.B.2, slide 13). A **variable** is a storage location that holds a value. Every variable has a name and a data type. A variable of a primitive type holds one value from that type.
3. **Declaring** (slide 13). A **variable declaration** gives the type and the name: `int coins;`. Usually it also stores a starting value: `int coins = 1250;`. The `=` stores the value on its right in the variable on its left; its full rules are Topic 1.4. A variable declared with no starting value cannot be used until it gets one (also Topic 1.4).
4. **Show** `GameStats.java` (slide 14, then run it in the IDE). Point to the type on the left and the kind of value on the right of each declaration. Run it. Ask why line 8 has no quotation marks around `coins`: the name means "the value stored in `coins`."
5. **Naming rules** (slide 13). No spaces; letters and digits, starting with a letter; not a word Java already uses, such as `int`; case-sensitive. Convention: **camel case** (`highScore`, `milesWalked`), and a name that says what the value means.
6. **Predict: MissingDeclaration.** `coins = 1250;` with no type. Run it: `cannot find symbol`, because no statement declared `coins`. Fix: put the type in front.
7. **Back to slide 19.** Each student writes one declaration for one of the five items they typed on Day 1, with a sensible starting value.

**MISCONCEPTION** Using a variable without declaring it

Students write `coins = 1250;` and expect Java to create the variable. Java only knows about a name after a declaration gives it a type. The same `cannot find symbol` message appears when the declared name and the used name differ only in capitalization (`LapCount` versus `lapcount`), so tell students to compare names letter by letter.

## Day 2 direct instruction: a variable's value can change

1. **Trace** `CoinCount` (1.2.B.2, slide 15). Slide 15 shows its first four lines and their trace table; build the table with the class one line at a time before revealing it, then run `CoinCount` for the last change to 40. Line 3 does not add 1300 to 1250; it replaces 1250. Each `println` displays whatever is in the box at that moment.
2. **Same rule for every type.** A `boolean` variable that starts `false` can later hold `true`; it still holds one value at a time.
3. **Predict: Redeclare.** The second line is `int coins = 1300;`. Run it:  
`variable coins is already defined`. Writing the type asks for a new variable, and one with that name already exists. Nothing runs, not even the first line (the compiler checks the whole program first, Topic 1.1).

**MISCONCEPTION** Writing the type again when changing a value

Students copy the declaration line and edit the number. Give them one rule: the type is written exactly once, in the declaration. Every later change is just the name, `=`, and the new value.

## Day 2 direct instruction: a variable holds only values of its type

1. **Predict: DecimalInInt** (slide 16 shows the same error with 2.5 slices). `int pizzaSlices = 3.5;` Run it and read the message in pieces: `incompatible types` (the value and the variable do not match) and `possible lossy conversion from double to int` (the `.5` would be lost). Java refuses to throw part of a value away silently. The right fix depends on the meaning: if half slices are possible, the variable should be a `double`.
2. **Two more mismatches.** Run `NumberInBoolean` and `CapitalTrue`, the IDE versions of slide 10. `boolean lightsOn = 1;` gives `int cannot be converted to boolean`. `boolean lightsOn = True;` gives `cannot find symbol`, because Java reads `True` as the name of a variable that does not exist.
3. **Connect to Day 1.** The rule exists because a type is a set of values: an `int` variable promises that its value is always an integer, and the compiler checks every value you store before the program runs.

### MISCONCEPTION Capitalizing true or false

`True` and `False` are not values in Java; they look like variable names, so the compiler reports `cannot find symbol`. Students who have used other languages make this mistake often. Point out that the message names `True` as a variable, which is the clue.

## Day 2 guided practice: break GameStats on purpose

Students type `GameStats.java` and run it. Then, one at a time, they make each change, predict, run, record, and undo:

1. Change `1250` to `1250.5` on line 5.
2. Change `false` to `False` on line 7.
3. Change `coins` to `Coins` on line 8.
4. Add `int speed = 5;` as a new line after line 7.
5. Add `paused = true;` as a new line after line 9.

For each, students record: Did it compile? If not, what did the compiler say? If so, what was displayed? Debrief: change 1 is

`incompatible types: possible lossy conversion from double to int`; changes 2 and 3 are `cannot find symbol` (for `False` and for `Coins`); change 4 is `variable speed is already defined`; change 5 compiles and displays `1250, 4.75, true`.

### TEACHER NOTE

Some IDEs underline these errors as students type and offer fixes. Ask students to ignore the suggestions and run anyway, so they see the compiler's own message. If computers are not available, run the five changes as a whole-class demo and have students write predictions first.

## Day 2 independent practice

Puzzle worksheet questions 2–10 (be the JVM, be the compiler, snippet pool, code tiles, the int-versus-double face-off, tracing a changing variable, a one-token fix, two truths and a lie, and who am I) are homework, about 25 minutes. Students who finish the IDE activity early can start them in class. Question 2 ends with a Think First prediction that question 7 settles, so ask students to work the questions in order.

## Day 2 closure and exit ticket

If a minute is left, use slide 21's sentence starter (why not one box for everything?) as a quick pair-share; slide 20 lists the vocabulary. Then give the exit ticket (about five minutes). Question 1 checks choosing a type and declaring a variable of it (1.2.A, 1.2.B.1), question 2 the primitive and reference categories (1.2.A.1–1.2.A.3), and question 3 that a variable holds one value at a time (1.2.B.2). Re-teach the weakest one at the start of Topic 1.3.