Variables and Data Types

Jessica's Office Hours

- **Fridays 1:45–2:45 p.m. in Levine 260**
- (This is slightly shifted from the initial announcement.)
- Come by with questions about the course, or just to chat!

Logistics

- HW00: **Due Wednesday September 11, 2024 @ 11:59 p.m. ET**
- Recitations start next week
- Regular office hour schedule starts next week, announcement on Ed soon

Learning Objectives

- To be familiar with primitive data types
- To be able to write expressions using primitive data types
- To know what a variable is
- To be able to declare variables
- To be able to solve problems using primitive type variables

Overview

One role of a computer program is to model and manipulate real or imaginary world entities. To do this, the computer must store some **data** to model these entities.

In this module, we will learn how to represent the properties (or attributes) of the entities that our program will manipulate

Example:

- Entity: `student`
- Properties: `name, age, height, etc.`

Data

Data is a piece of information. We use data to model entities & solve problems.

All data (in Java) has a **data type**

- Defines the set of possible values a piece of data can have
- Defines the possible operations that can be performed on that data

Two types of data types in Java

- Primitive types (today!)
- Object types (later!)

Primitive types

`int`: stores whole numbers (positive or negative) like `3`, `-5`, `19000`

- "int" is short for Integer

`double`: stores decimal numbers (positive or negative) like `3.5`, `-5.1`, `19000.1`

- Note: not infinitely precise. Computers are physical and finite.

`boolean`: stores Boolean values, either `true` or `false`

There are others we will introduce later.

Operations on int

Type of operand 1	Operator	Type of operand 2	Type of result	Example	Result
`int`	`+`	`int`	`int`	`3 + 5`	`8`
`int`	`-`	`int`	`int`	`4 - 6`	`-2`
`int`	`*`	`int`	`int`	`2 * 3`	`6`
`int`	`/`	`int`	🤯 `int` 🤯	`3 / 2`	`1`

Testing Operator Behavior

If you want to verify the result of some operation, you can place it in a print statement:

```
System.out.println(3 + 5); // prints 8 when program is run

System.out.println(3 / 2); // prints 1
```

No quotation marks (`"`) are needed since we're not printing text literally.

The modulo (%) operator

The mod operator `(x % y)` returns the remainder after you divide `x` (first number) by `y` (second number)

```
5 % 2 --> 1
```

```
4 % 2 --> 0
```

Properties of Modulo

Observe the following pattern:

```
0 % 3 --> 0
1 % 3 --> 1
2 % 3 --> 2
3 % 3 --> 0
4 % 3 --> 1
5 % 3 --> 2
6 % 3 --> 0
```

- The result of `x % y` is always between `0` and `y - 1` (inclusive) *when* `x` *is positive*
- When `x` is a multiple of `y`, the result is `0`

Properties of Modulo

Pattern holds on other values!

```
0 % 4 --> 0
1 % 4 --> 1
2 % 4 --> 2
3 % 4 --> 3
4 % 4 --> 0
5 % 4 --> 1
6 % 4 --> 2
7 % 4 --> 3
8 % 4 --> 0
9 % 4 --> 1
```

- The result of `x % y` is always between `0` and `y - 1` (inclusive) *when* `x` *is positive*
- When `x` is a multiple of `y`, the result is `0`

Operations on double

Type of operand 1	Operator	Type of operand 2	Type of result	Example	Result
`double`	`+`	`double`	`double`	`3.5 + 5.5`	`9.0`
`double`	`-`	`double`	`double`	`4.0 - 6.0`	`-2.0`
`double`	`*`	`double`	`double`	`2.5 * 1.0`	`2.5`
`double`	`/`	`double`	`double`	`3.0 / 2.0`	`1.5`

Operations on double and int

When one of the operand is of type `double`, the result is of type `double` always.

Type of operand 1	Operator	Type of operand 2	Type of result	Example	Result
`double`	`+`	`int`	`double`	`3.5 + 5`	`8.5`
`int`	`-`	`double`	`double`	`4 - 6.0`	`-2.0`
`double`	`*`	`int`	`double`	`2.5 * 1`	`2.5`
`double`	`/`	`int`	`double`	`3.0 / 2`	`1.5`

Logical Operations for Booleans

Booleans are either `true` or `false`, so the set of operations we can do with these values is different than numeric types.

- "and": evaluates to `true` only when both operands are `true`
  - "Today is Wednesday and this class is CIS 1100" is `true`
  - "Today is Wednesday and this class is CIS 1600" is `false`, even though the first part is `true`.
- "or": evaluates to `true` only when at least one operand is `true`
  - "Today is Wednesday or this class is in the Art History Department" is `true`.
- "not": negates the value of one boolean.

Operations on boolean

Type of operand 1	Operator	Type of operand 2	Type of result	Example	Result
`boolean`	`&&`	`boolean`	`boolean`	`true && false`	`false`
`boolean`	`||`	`boolean`	`boolean`	`true || false`	`true`
`boolean`	`!`	`N/A`	`boolean`	`!true`	`false`

Comparison: Equality

The == operator is used to check for equality.

The result is a `boolean` value (`true` or `false`) and the input operands must both be values of the same type.

```
4 == 5;          // evaluates to false
5.0 == 5.0;      // evaluates to true
true == false;   // evaluates to false
false == false;  // evaluates to true
```

The result of the comparison can be printed:

```
System.out.println(4 == 5); // prints false
```

Comparison: Inequality

The != operator is used to check for inequality (not equals).

The result is a `boolean` value (`true` or `false`).

```
4 != 5;          // evaluates to true
5.0 != 5.0;      // evaluates to false
true != false;   // evaluates to true
false != false;  // evaluates to false
```

The result of the comparison can be printed

```
System.out.print(4 != 5);  // prints true
```

Comparison: Others

For types like `int` and `double`, we can also perform other comparisons

The result is a `boolean` value (`true` or `false`);

Operator Name	Syntax	Example	Example Output
Less than	`<`	`5 < 6`	`true`
Less than or equal to	`<=`	`5 <= 5`	`true`
Greater than	`>`	`2 > 3`	`false`
Greater than or equal to	`>=`	`5 >= 1`	`true`

Operator Chaining & Priority

You can chain multiple operators together in one line:

`110 + 120 + 160 + 121 + 240`

- Sometimes the order of operations is unclear. Example:
  - `110 + 120 * 2 == 2`
- To avoid confusion, use parentheses to specify the order of operations:
  - `(110 + (120 * 2)) == 2`

Parenthesis are recommended for general use.

Expressions

A sequence of ***operators*** and their ***operands*** (values to act on) that specifies a computation. **Has a resulting value.**

Examples:

- `1 + 2 + 3`
- `240 != 240`
- `((4 - 8) / (1 * 2)) >= 0`
- `3.14 * 6.02 - 1000.00`
- `!false && true == false`

Live Coding DEMO (Part 1)

`LeapYear.java`: a program that will determine if a year is a leap year.

A leap year takes place every four years.

BUT! If the year is divisible by 100, it's not actually a leap year.

BUT! If the year is divisible by 400, it is again a leap year!

Print `true` if a given year corresponds to a leap year, and `false` otherwise.

Variables

Variables are a **portion of computer memory** used to store a value (data).

- Allows us to store data and the result of computations for later usage.
- A way for the computer to “remember” data.

Every variable has a **name** that we can use to refer to the variable.

Every variable has a **data type** that defines which data can be stored in that variable.

Variable Vocabulary

- **Declaring a variable** happens when we write its type and its name together for the first time. This brings the variable into the program and assigns it a default value based on its type. It can only be done once per variable.
- **Assigning a value to a variable** happens when we use the `=` operator to store a value in a variable. This can be done at the same time as declaring the variable—or not—and it can be done many times after that.
- **Initialization** is the process of giving a variable its first value.

Variable declaration

- Creates a variable
- Associates a variable to a type
  - The type determines how much space (bits) the computer will use to store the value associated with the variable.
- Done by writing the type followed by the variable name

Examples

```
// declaring the variable score
double score;

// declaring the variable age
int age;
```

Variable initialization

Assigns a value to a variable: using the `=` sign

- The value and the type of the variable must be compatible

```
// declaring and initializing the variable name (one line)
double score = 98.3;
// declaring the variable age (two lines)
int age;
age = 14;
// declaring and initializing variable isTakingCIS1100 (one line)
boolean isTakingCIS1100 = true;
```

Operations on variables

- Assignment statement (`=`) initializes or changes the value of a variable previously declared
- Operators can be applied to values to perform computation
  - Variables store values!

```
// initialize variable x and put the value 1100 in it.
int x = 1100;

// update the value of x to be the result of 2400 + 1400.
x = 2400 + 1400;
```

Variables in Expressions

Variables can be named in expressions, which will use the value stored in the variable as part of the computation:

```
int x = 12;
int y = x * 30;  // results in y being 360
int z = 20 + y;  // z equals 380
x = x + 1;       // x equals 13
```

The value of the expression on the right hand side depends on the value of the variable at the moment the expression is evaluated—changing `x` after `y` is assigned does not change the value of `y`.

Compound Assignment Operators

Shortcuts that do a math operation and assignment in one step!

+ shortcuts	- shortcuts	* shortcut	/ shortcut	% shortcut
x = x + 1;	x = x - 1;	x = x * 2;	x = x / 2;	x = x % 2;
x += 1;	x -= 1;	x *= 2;	x /= 2;	x %= 2;
x++;	x- -;			

Printing a variable

Put the variable name without the quotes in the print command

```
double score = 43.5;
System.out.print(score);
```

Prints `43.5` ✅

Printing a variable

Using quotes will just print out the characters literally—you'll get the variable name rather than its value.

```
double score = 43.5;
System.out.print("score");
```

Prints `score` 😞

Printing a variable

Use the `+` operator to append the value of a variable to a text in the print command

```
System.out.print("Score in game: " + score);
```

Prints `Score in game: 43.5`

Operator Type Errors

Sometimes mixing variable types and values will result in compiler errors:

```
// Wrong value for the specified variable type
int pi = 3.14159;
double x = true;

// Using operators with incompatible/mismatching types
int y = 1 + false;
boolean z = 110 && 120;
```

Live Coding DEMO (Part 2) w/ Variables!

LeapYear.java

Program that will determine if a year is a leap year.