Types, Variables, & Expressions

By: Megan Avery
Updated July 2018

Before We Begin

- Make sure you have Java downloaded on your computer
  - can run **`java -version`** in terminal to check
- Make sure you have IntelliJ downloaded on your computer

- *Suggested:* Watch the “IntelliJ & First Java Program” tutorial

Types & Variables

Vocab Alert!

type: category or set of data values

- Affects the types of operations that can be performed on the data
- In Java always need to specify the type for data

statically typed: the type for variables is checked at compile time

- Java is statically typed

dynamically typed: the type for variables is checked when program runs aka at run time

primitive types: the 8 simple types for numbers, text etc.

Why do languages need types?

Computers see information as a long stream of 1s and 0s, types tell the computer how to view those ones or 1s.

Each type has distinct properties, value ranges, and number of 1s or 0s it takes up.

Vocab Alert!

binary: numbers written in base 2, basically a collection of 1s and 0s

- “normal” numbers are written in base 10

bit: a single 1 or 0 in a binary number or in the computer’s memory

byte: a collection of 8 bits

nibble: a collection of 4 bits

hexadecimal: a number in base 16

- values switched to letters after 9 (i.e. A = 10, B = 11, etc.)
- 1 nibble in binary = 1 “number” in hex

8 Primitive Types

byte: 8-bit signed whole number
- from -128 to 127

short: 16-bit signed whole number
- from -32,768 to 32,767

int: 32-bit signed whole number
- from -2^{31} to $2^{31}-1$

long: 64-bit signed whole number
- from -2^{63} to $2^{63}-1$

float: 32-bit non-whole (floating point) number
- used when wanting to save memory

double: 64-bit floating point number
- default choice for non whole numbers

boolean: truth value
- possible values are true and false

char: 16-bit unicode character, e.g. ‘A’
- can also set values by ASCII values

Vocab Alert!

variable: A piece of the computer's memory that is given a name and type, and can store a value.

variable declaration: Sets aside memory for storing a value.

- ex: `int x;`

memory address: where in the the computer's memory a variable lives, chosen for a variable during declaration

- when printed often shown as a hexadecimal number

assignment: store a value for a variable

- ex: `int x = 3;` or `x = 3;` if x already exists

cast: force a variable to be a different type

Example Assignments

```
byte b = 0;

short s = 0;

int i = 0;

long l = 0l;
```

```
float f = 0.0f;

double d = 0.0;

boolean truthy = true;

char c1 = 'A';

char c2 = 65;
```

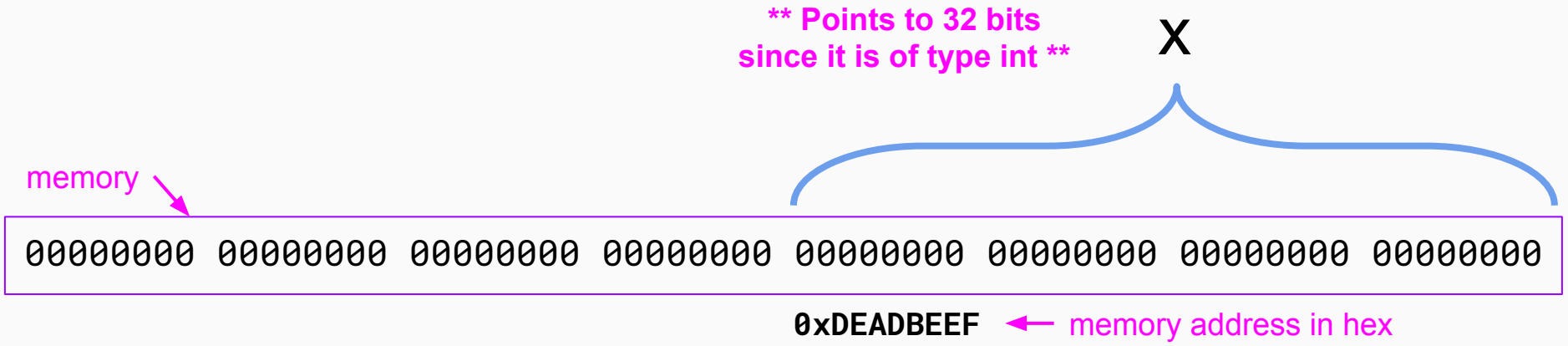
A Note on Variable Names

- Different languages have different conventions for variables names
- For all languages variable names...
  - must start with a letter
  - can contain letters and numbers
  - cannot contain spaces
  - should be descriptive and easy to understand
- In Java…
  - regular variable names are “camel case”
    - ex: helloWorld
  - constants (discussed later) are all caps “snake case”
    - ex: HELLO_WORLD

Putting it all together: declaring & assigning x

Execute Code:

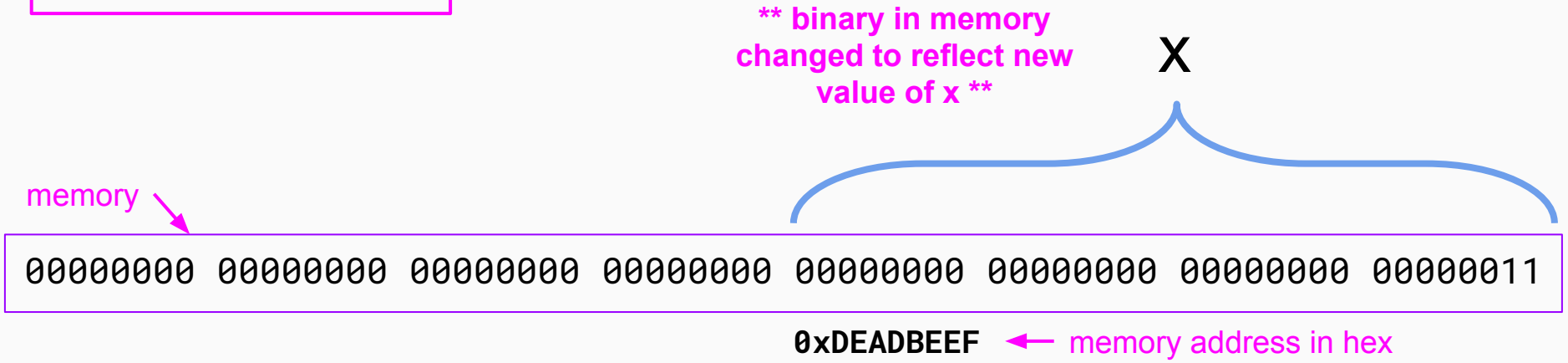
```
int x = 0;
```



Putting it all together: updating x

Execute Code:

```
x = 3;
```



Cast Demo

The “final” keyword

If you know you won’t be changing the value of a variable you can mark it with the final keyword upon declaration.

- ex: `final int i = 0;`

Last Note on Types

When deciding which type to use for a variable try to pick the one that makes the most sense. Don’t use a double when you aren’t going to use the decimal part of it.

Examples:

- Number of days it rained last year → int
- Average of a group of numbers → double
- If the current month has a y in it → boolean

Expressions

Vocab Alert!

operator: combines multiple values or expressions

- ex: + - / * %

expression: a combination of values and / or operations that results in a new value

- ex: 1 + 4 * 5 (7 + 2) * 6 / 3

The Modulus (%) Operator

Modulus is a way to get the division remainder between 2 numbers.

Examples:

- `14 % 3 = 2`
  - since 3 * 4 + 2 = 14
- `8 % 2 = 0`
  - Modding by something that goes into the first number evenly will always give 0
- `230857 % 10 = 7`
  - Able to get the last digit of any number by modding with 10 (get last 2 by modding 100 and so on)
- `3 % 7 = 3`
  - Modding a small number by a larger one will always give the smaller number again
- `12.4 % 3.0 = 0.40000000000000036`
  - Confused by the extra digits? Doubles aren't represented in an exact way in the computer so sometimes you'll get output like this with them

Integer Division

When 2 ints are divided the result is another int that is the exact number of times one number goes into the other

Examples:

- 21 / 5 = 4
- 84 / 10 = 8
- 30 / 6 = 5
- 8 / 9 = 0

Note: If want to get exact division with integers will need to cast one to a double

- ex: (double) 21 / 5 = 4.2

Operator Precedence - PEMMDAS

Please - parenthesis

Excuse - exponents

My - multiplication

Marvelously - modulus

Deranged - division

Same precedence so apply left to right

Aunt - addition

Sally - subtraction

Same precedence so apply left to right

Printing Variables & Expressions Demo

Note: printing variables/ expressions with words

When you are printing variables/expressions with words you **must** always include a + between the variable/ expression and the words in quotes, you will get a compile error otherwise.

Also, once you have done “some words” + variable/expression it becomes a non number type and any other operations will no longer work as they normally would unless put inside of parenthesis.

- ex: `System.out.println("hello world" + 3 + 4);` → hello world34

Practice Problem

Write a program to calculate your total at a restaurant based on the subtotal. You must calculate both tax and tip and add them to the subtotal to get your final total.

Note: tax is 8% and the tip should be 15% of the subtotal *after* tax has been added

Practice problem solution demo

Practice for Later:
Write a program to these specs

- Has a variable for your age
- Has a variable for how many decades old you are calculated from your age variable using integer division
- Has a variable for how many years you are into your current decade calculated from your age variable using modulus
- Has a variable for how close you are to 100 years old as a percentage calculated from your age variable using non-integer division
- Has a variable for if you're over 30
- Prints all these values out with appropriate words surrounding them to give them context

Example Output:

Hi! My name is Grace Hopper.

I was 85 years old when I died in 1992.

That's 8 decades!

I'm was only 5 years into my latest decade.

I was .85 of the way to 100 years old!

Was I over 30? true!

Grace Hopper was an amazing woman in tech, learn more about her here: https://en.wikipedia.org/wiki/Grace_Hopper

The End