Java Programming

Zheng-Liang Lu

Department of Computer Science & Information Engineering
National Taiwan University

Java 405
Spring 2024

```
class Lecture2 {

        "Data types, Variables, and Operators"

}

// Keywords:
byte, short, int, long, char, float, double, boolean, true, false,
import, new
```

Example

Given the circle radius, say 10, determine the area.

- Input: how to store the value of the circle radius?
- Algorithm: how to compute the resulting area?
- Output: how to show the result?

```java
1 public class ComputeAreaDemo {
2 
3     public static void main(String[] args) {
4 
5         // INPUT
6         int r = 10;
7 
8         // ALGORITHM
9         double A = r * r * 3.14;
10 
11         // OUTPUT
12         System.out.println(A);
13 
14     }
15 }
```

- In Line 6, we declare the variable *r* an integer (int) with its initial value 10.
- In Line 9, we store the circle area in the variable *A* which is decimal (double).
- The keywords int and double are two of primitive types.

Simple Analog: Variable ≈ Box

Variable Declaration

- First, we name the variable, say price.
- We then determine a proper type for price, for example,

```
... // ignore the common part; the same applies hereinafter

        int price; // price is a variable declared an integer type

...
```

- The rule of variable declaration looks like

  data-type *variable-name*;

  - For example, **String[]** *args* in the main method.
- This rule is similar to C, C++, and C#.

Naming Rules

- The naming rule excludes the following cases:
  - cannot start with a digit;
  - cannot be any reserved word (see the next page);
  - cannot include any blank between letters;
  - cannot contain operators, like $+$, $-$, $*$, $/$.
- Note that Java is case-sensitive, for example, the letter A is different from the letter a.
- These rules are also applicable to methods, classes, etc.
- These rules, again, are similar to C, C++, and C#.

Reserved Words[1]

`abstract`	`double`	`int`	`super`
`assert`	`else`	`interface`	`switch`
`boolean`	`enum`	`long`	`synchronized`
`break`	`extends`	`native`	`this`
`byte`	`final`	`new`	`throw`
`case`	`finally`	`package`	`throws`
`catch`	`float`	`private`	`transient`
`char`	`for`	`protected`	`try`
`class`	`goto`	`public`	`void`
`const`	`if`	`return`	`volatile`
`continue`	`implements`	`short`	`while`
`default`	`import`	`static`	
`do`	`instanceof`	`strictfp*`	

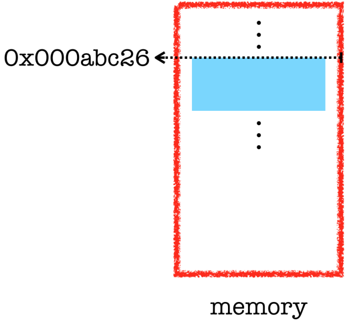
- Coverage: $44 / 50 = 88\%$.

[1] Based on JDK8. You can check the language changes here.

Things behind Variable Declaration

- Variable declaration asks to allocate a proper memory space to the variable (box).
- The size of the allocated space depends on its data type.
- We count the space size in bits or bytes.
    - A bit presents a binary digit.
    - 1 byte is equal to 8 bits.
- For example, an int value occupies 32 bits (or 4 bytes) in the memory.

Variable Name as Alias of Memory Address



- Literals that start with 0x are hexadecimal (hex) integers.[2]
- Hex numbers are widely used to represent, say addresses and colors.[3]

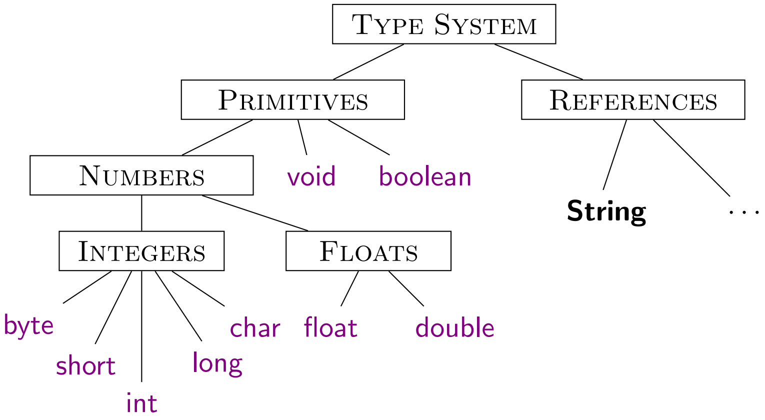
[2]See https://en.wikipedia.org/wiki/Hexadecimal.

[3]Try https://htmlcolorcodes.com/.

Data Types

- Every variable needs a type.
- Also, every statement (or expression) has a final type.
- The notion of data types is vital to programming languages.
  - I would say that, the role of data types acts like the physics law in the universe.
- Java is a static-typed language, similar to C, C++, and C#.
  - A variable is available after declaration and cannot changed in runtime.
- We now proceed to introduce the two categories of data types: primitive types, and reference types.

Type System: Overview



Digression: Binary System[5]

- We have been familiar with the decimal system. (Why?)
- Computers know only the binary system because of its nature: only two states, on and off.[4]
- However, both systems are equivalent except that they differ in representations.
- For example,
$$999_{10} = 9 \times 10^2 + 9 \times 10^1 + 9 \times 10^0.$$
- Similarly,
$$111_2 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 7_{10}.$$
- In most cases, we don't need to deal with binary codes directly because we are using high-level languages.

[4] How about the quantum computers? Spin up and down. See Qubit.
[5] See How Exactly Does Binary Code Work? by José Américo NLF Freitas.

Integers

Name	Bits	Range	Approx. Range
byte	8	0 to 255	<= 255
short	16	-32768 to 32767	$\pm 3 \times 10^4$
int	32	-2147483648 to 2147483647	$\pm 2 \times 10^9$
long	64	-9223372036854775808 to 9223372036854775807	$\pm 9 \times 10^{18}$

- The range is limited to its finite size of storage.
- If a value is out of the feasible range, an overflow occurs.
- The int type is the most used unless otherwise noted.
- If you want to write down a long-type literal, say 9876543210, you should write 9876543210L, where the suffix L indicates the long type.

Floating-Point Numbers

Name	Bits	Range
float	32	$1.4e{-}045$ to $3.4e{+}038$
double	64	$4.9e{-}324$ to $1.8e{+}308$

- The notation e (or E) represents the scientific notation, based 10.
  - For example, $1e2 = 100$ and $-1.8e{-}3 = -0.0018$.
- We use floating-point numbers when evaluating expressions that require fractional precision, say sqrt() and log().
- In this sense, integers seem redundant because floating-point numbers could represent integers and also decimals.
- However, the floating-point system can only approximate the real-number arithmetic! (Why?)

Machine Epsilon[7]

```
...

        System.out.println(0.5 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
        // Output? Why?

...
```

- We relieve the machine epsilon by proper algorithm design.
- In critical applications, we even avoid to use the floating-point numbers but integers.[6]

[6] Also read https://news.cnyes.com/news/id/3680649.

[7] See Machine Epsilon and https://0.30000000000000004.com/.

Another Example

```
...
        System.out.println(3.14 + 1e20 - 1e20);   // Output?
        System.out.println(3.14 + (1e20 - 1e20)); // Output?
...
```

- Floating-point arithmetic (FP)[8] is arithmetic using formulaic representation of real numbers as an approximation to support a trade-off between range and precision.[9]

[8] See https://en.wikipedia.org/wiki/Floating-point_arithmetic.

[9] You may also read this article What Every Computer Scientist Should Know About Floating-Point Arithmetic.

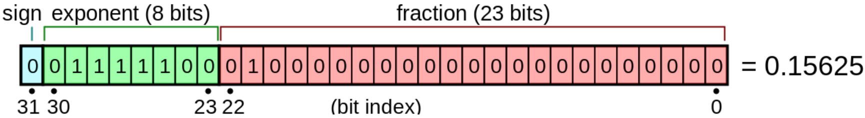
IEEE Floating-Point Representation[10]

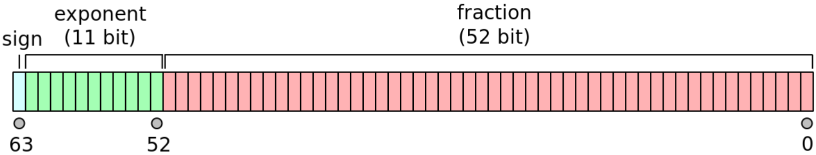
$$x = (-1)^s \times M \times 2^E$$

- The sign bit s determines whether the number is negative ($s = 1$) or positive ($s = 0$).
- The mantissa M is a fractional binary number that ranges either between 1 and $2 - \varepsilon$, or between 0 and $1 - \varepsilon$.
- The exponent E weights the value by a (possibly negative) power of 2.

[10] William Kahan (1985): IEEE754; See also Double-Precision FP Format.

Illustration





- That is why we call a double value.
- Double values have at least 16 significant digits in decimal!

Assignments

- The equal sign (=) is used as the assignment operator.
- An assignment statement designates a value to the variable.

```java
int x, y;     // Variable declaration.
x = 0;        // Assign 0 to x.
y = x + 1;    // y = 1 (trivial?)
x = x + 1;    // Is this weird?
```

- Direction: from the right-hand side to the left-hand side.[11]
  - Copy a value from the right-hand side (value or expression) to the space indicated by the variable in the left-hand side.
- You cannot write codes like $\boxed{1 = x}$ because 1 cannot be resolved to a memory space.

[11] The variable x can be a l-value and r-value, but 1 and other numbers can be only r-value but not l-value. See Value.

Two-Before Rule

```java
int x;
...
x = 0;
...
x = x + 1;
```

- Rule 1: a variable must be declared before any assignment.
- Rule 2: a variable must be initialized with a value before being used.

Arithmetic Operators

Operator	Operation	Example	Result
+	Addition	12 + 34	46
−	Subtraction	56 − 78	−22
*	Multiplication	90 * 12	1080
/	Division	3.0 / 2.0	1.5
%	Remainder	20 % 3	2

- What if 3 / 2?
- The result depends on the types of its operands!

Concept Check

```java
...
        double x = 1 / 2;
        System.out.println(x); // Output? Why?
...
```

- What is the output?

Two of Program Stages

- Compile time (or compilation period):
  - memory allocation for x,
  - constant literals (in this case 1, 2),
  - linking the println method, etc.
- Run time (or execution period):
  - execution of arithmetic operation
  - output the result, etc.

Compatibility and Type Conversion

- If a type is compatible to another, then the compiler will perform the implicit conversion.
  - For example, the integer 1 is compatible to a double value 1.0.
- Clearly, Java is a weakly-typed language.[12]
- However, there is no automatic conversion from double to int. (Why?)
- To do so, you must use a cast, which performs an explicit conversion.
- Similarly, a long value is not compatible to int.

[12] See Statically vs. Dynamically Typed Languages.

Casting

```java
...
        int x = 1;
        double y = x; // Compatible; implicitly converted.
        x = y;        // Not allowed unless casting.
        x = (int) y;  // Succeeded!!
...
```

- Note that the Java compiler does only type checking but no real execution before compilation.
- In other words, the actual values of x and y are unknown until the program is executed.

Compatibility and Type Conversion (Concluded)

- Small-size types $\rightarrow$ large-size types.
- Small-size types $\not\leftarrow$ large-size types (need a cast).
- Simple types $\rightarrow$ complicated types.
- Simple types $\not\leftarrow$ complicated types (need a cast).

Text: Characters & Strings

- Each character is encoded in a sequence of 0's and 1's.
  - For example, ASCII. (See the next page.)
- The char type denotes characters, which are represented in Unicode, a 16-bit unsigned value.[13]
- However, we often use **String** to present texts, as shown before.
- As an analogy, a molecule (string) consists of atoms (characters).[14]

---

[13] Unicode defines a fully international character set that can represent all of the characters found in all human languages.

[14] A **String** object comprises characters equipped with plentiful tools.

ASCII (7-bit version)

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Example

```
1 ...
2         char c = 'a';  // A char value should be single-quoted.
3         System.out.println(c + 1); // Output 98!! (why?)
4         System.out.println((char)(c + 1)); // Output b.
5
6         String s = "Java"; // A string should be double-quoted.
7         System.out.println(s + 999); // Output Java999.
8 ...
```

- We may apply arithmetic operators to characters, say Line 4 for some purposes.[15]
- In Line 7, the result of applying the + operator to string is totally different from Line 3 & 4. (Why?)

[15] For example, https://en.wikipedia.org/wiki/Cryptography.

Boolean Values[17]

- Programs are expected to do decision making by itself, say self-driving cars.[16]
- Java provides the boolean-type flow controls (branching and iteration).
- The boolean type allows only two values: true and false.
- Note that boolean values cannot be cast to non-boolean type, and vice versa. (Why?)

[16] See https://www.google.com/selfdrivingcar/.

[17] George Boole (1815–1864) is the namesake of the branch of algebra known as Boolean algebra. See https://en.wikipedia.org/wiki/George_Boole.

Relational Operators

Operator	Name
$<$	less than
$<=$	less than or equal to
$>$	greater than
$>=$	greater than or equal to
$==$	equal to
$!=$	not equal to

- Relational operators take two operands and return a boolean value.
- Note that the mathematical equality operator is $==$, not $=$ (assignment).

Example

```
1 ...
2         int x = 2;
3         System.out.println(x > 1);     // Output true.
4         System.out.println(x < 1);     // Output false.
5         System.out.println(x == 1);    // Output false.
6         System.out.println(x != 1);    // Output true.
7         System.out.println(1 < x < 3); // Sorry?
8 ...
```

- In Line 7, $1 < x < 3$ is syntactically wrong.
- You need to split a complex statement into several basic statements and joint them by proper logical operators.
- For example, $1 < x < 3$ should be

$$1 < x \,\&\&\, x < 3,$$

where && represents the AND operator.

Conditional Logical Operators[18]

Operator	Name
!	NOT
&&	AND
||	OR
^	EXCLUSIVE-OR

- We often use XOR to denote the exclusive-or operator.

[18] The bit-wise operators are ignored in my course because most of Java programmers do not use those directly. See Bitwise and Bit Shift Operators if necessary.

Truth Table

- Let X and Y be two boolean variables.
- The truth table for logical operators is shown below:

X	Y	!X	X&&Y	X || Y	X $\wedge Y$
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

Life Applications Using Boolean Logic

- Basic instructions, such as arithmetic operators, are implemented by Boolean logic.
- For example, 1-bit adder can be implemented by using the XOR operator.[19] (Try!)
- Can you image that the combination of these very basic elements (0, 1, AND, OR, NOT) with jumps produces so-called Artificial Intelligence (AI) like AlphaGo which beat human beings in 2016 and ChatGPT which starts a new era in the end of 2022?

[19] See also logic gates in digital circuit designs.

"Logic is the anatomy of thought."

– John Locke (1632–1704)

"This sentence is false."

– anonymous

"I know that I know nothing."

– Plato

(In Apology, Plato relates that Socrates accounts for his seeming wiser than any other person because he does not imagine that he knows what he does not know.)

Arithmetic Compound Assignment Operators

- For simplicity, let x and k be any number.

Operator	Description
$x{+}{+}$	Increment by one
$x{+}{=}k$	Cumulative increment by k
$x{-}{=}k$	Cumulative subtraction by k
$x{*}{=}k$	Cumulative multiplication by k
$x/{=}k$	Cumulative division by k
$x\%{=}k$	Cumulative modulus by k
$x{-}{-}$	Decrement by one

Example: Integers

```java
...
        int x = 1;
        System.out.println(x); // Output 1.

        x = x + 1;
        System.out.println(x); // Output 2.

        x += 2;
        System.out.println(x); // Output 4.

        x++; // Equivalent to x += 1 and x = x + 1.
        System.out.println(x); // Output 5.
...
```

Example: Characters and Strings

- Some of the aforesaid operators are also applicable to char values and **String** objects.
- For example,

```
...
        char x = 'a';
        x += 1;
        System.out.println(x); // Output b.
        x++;
        System.out.println(x); // Output c.

        String y = "Java";
        y += 999
        System.out.println(y); // Output Java999.
...
```

Discussion: ++x vs. x++

```java
...
        int x = 0;
        int y = ++x;
        System.out.println(y); // Output 1.
        System.out.println(x); // Output 1.

        int w = 0;
        int z = w++;
        System.out.println(z); // Output 0.
        System.out.println(w); // Output 1.
...
```

- `x++` first returns the old value of *x* and then increments itself.
- Instead, `++x` first increments itself and then returns the new value of *x*.
- We will use these notations very often.

Operator Precedence[20]

Precedence	*Operator*
↓ (highest)	var++ and var-- (Postfix)
	+, - (Unary plus and minus), ++var and --var (Prefix)
	(type) (Casting)
	! (Not)
	*, /, % (Multiplication, division, and remainder)
	+, - (Binary addition and subtraction)
	<, <=, >, >= (Comparison)
	==, != (Equality)
	^ (Exclusive OR)
	&& (AND)
	|| (OR)
↓ (lowest)	=, +=, -=, *=, /=, %= (Assignment operator)

[20]See Table3-10 in YDL, p. 116.

Tip: Using Parentheses

- The program always evaluates the expression inside of parentheses first.
- If necessary, using parentheses in expressions could change the natural order of precedence among the operators.

Scanner: Example of Reference Types

- To reuse your program, it is inconvenient to modify and recompile the source code for every radius.
- Reading inputs from the user's keyboard in the console is the easiest way to interact with programs.
- Java provides the **Scanner** object with easy-to-use input methods.
- Note that **System**.in refers to the standard input device, by default, the keyboard.

Example

```java
import java.util.Scanner;
...
        // Create Scanner object to receive data from keyboard.
        Scanner input = new Scanner(System.in);

        // INPUT
        System.out.println("Enter r?");
        int r = input.nextInt(); //

        // ALGORITHM
        double A = r * r * 3.14;

        // OUTPUT
        System.out.println(A);
        input.close(); // Cleanup: reclaim the resource.
...
```

Discussions (1/2)

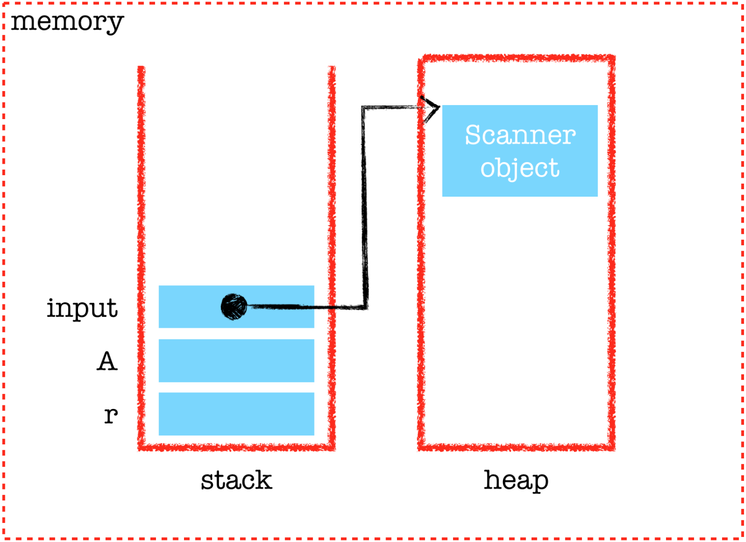
- In Line 1, we include the **Scanner** class, which belongs to the java.util package, by using the import statement.
  - We put these import statements in the beginning of the file.
  - Note that we can't leave these import statements in any class.
- In Line 4, the new operator followed by **Scanner** is to create a **Scanner** object.
- This object works as an agent between the keyboard and your program.
- In Line 9, the nextInt method of **Scanner** is used to convert the input to an int value.

Discussions (2/2): General Concepts

- All runtime objects are created dynamically and resided in the heap. (See the figure in the next page.)
- Before manipulating the **Scanner** object, its address is assigned to the variable *input*, which is allocated in the stack.
- Hence *input* is called a reference to the **Scanner** object.[21]
- Clearly, the memory contains human data and also references (i.e., memory addresses).

[21] If you have programming experiences in C/C++, then this reference is similar to the concept of pointers.

Illustration: Simplified Memory Model



Methods Provided by Scanner[22]

Method	*Description*
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.
`next()`	reads a string that ends before a whitespace character.
`nextLine()`	reads a line of text (i.e., a string ending with the *Enter* key pressed).

[22] See Table 2-1 in YDL, p. 38.

Exercise: Body Mass Index (BMI)

Write a program to take user name, height (in cm), weight (in kgw) as input, and then output the user name attached with his/her BMI, which is

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}.$$

- Be careful about unit conversion!

```
1  ...
2          Scanner input = new Scanner(System.in);
3
4          // INPUT
5          System.out.println("Enter your name?");
6          String name = input.nextLine();
7
8          System.out.println("Enter your height (cm)?");
9          double height = input.nextDouble();
10
11         System.out.println("Enter your weight (kgw)?");
12         double weight = input.nextDouble();
13
14         // ALGORITHM
15         double bmi = 10000 * weight / height / height;
16
17         // OUTPUT: name (bmi)
18         System.out.println(name + " (" + bmi + ")");
19 ...
```

- Make sure that you understand Line 18.

Exercise: Two Descriptive Statistics

Write a program to take 3 numbers as user's input and output the arithmetic average with its standard deviation.

- Let a, b, c be the double variables.
- Then its standard deviation is

$$\sqrt{\frac{\sum (x_i - \overline{x})^2}{3}},$$

where $x_i = \{a, b, c\}$ and $\overline{x} = (a + b + c) / 3$.

- You may use two of **Math** methods:[23] **Math**.pow(double x , double y) for x^y and **Math**.sqrt(double x) for $\sqrt{x}$.

[23] See https://docs.oracle.com/javase/tutorial/java/data/beyondmath.html.

```java
...
        // INPUT
        Scanner input = new Scanner(System.in);
        System.out.println("a = ?");
        double a = input.nextDouble();
        System.out.println("b = ?");
        double b = input.nextDouble();
        System.out.println("c = ?");
        double c = input.nextDouble();
        input.close();

        // ALGORITHM
        double mean = (a + b + c) / 3;
        double std = Math.sqrt((Math.pow(a - mean, 2) +
                                Math.pow(b - mean, 2) +
                                Math.pow(c - mean, 2)) / 3);

        // OUTPUT
        System.out.println("Mean = " + mean);
        System.out.println("Std = " + std);
...
```