### Java Syntax bles. Data Types, and Opera

Alwin Tareen

#### Variable Declaration

#### What is a variable?

A variable is a named piece of memory that you can use to store information in a Java program.

int average;

#### Declaring a variable

A variable declaration consists of two parts: a **data type**, and an **identifier name**.

data type: int

identifier name: average

## Variable Naming Rules

 A variable name must begin with a letter(not a number or symbol).

```
int total; // Legal
double 2scoops; // Not legal
```

2. The variable name must be a sequence of letters or digits. Symbols (@, #, \$, %, &, etc.) cannot be used at all.

```
double good4you; // Legal boolean work@home; // Not legal
```

3. The length of a variable name is unlimited.



# Variable Assignment

#### The assignment operator: =

The equals sign is used to **assign** a value to a variable.

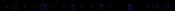
```
total = 58;
```

#### Different from math notation

The assignment operator does **not** possess the same meaning as mathematical equality(even though they seem similar).

```
int distance; // declaration
distance = 42; // assignment
```

- The assignment occurs from right to left.
- ► The value on the **right** is copied into the variable on the **left**.



# Variable Assignment

#### Updating a variable

The assignment operator can be used to **replace** the contents of a variable with a new value.

```
int score; // declaration
score = 0; // assignment
score = 3; // update
score = 5; // update
```

### Initializing a variable

Declaring and assigning a value to a variable can be **combined** into a single step.

```
int velocity = 0; // initialize to 0
```



### User Friendly Output

#### Displaying a variable

When printing out a variable, it is useful to give a small description, so the user can recognize it.

#### Printing without a description:

```
int cost = 21;
System.out.println(cost);
```

### Printing with a description(better):

```
double price = 19.95;
System.out.println("The price is: " + price);
```

# The Concatenation Operator

#### Combining a String and a variable

When the plus sign is used in a println() statement with a String, it concatenates(joins).

```
int grade = 87;
System.out.println("The grade is: " + grade);
```

You can also use concatenation with a numerical value:

```
System.out.println("The price is: " + 19.95);
```

### Primitive Data Types

Data Type	Memory Allocation	Range of Values
boolean	1 bit	true or false
int	4 bytes	max value: $2^{31} - 1$
		min value: $-2^{31}$
double	8 bytes	$-1.79 imes10^{308}$ to
		$+1.79 \times 10^{308}$

#### The integer data type: int

These are represented by a sequence of binary digits in memory.

### The floating-point data type: double

These are composed of two parts: a **mantissa** and an **exponent**. They are subject to rounding errors.



### Arithmetic Operators

Symbol	Operation	
+	addition	
-	subtraction	
*	multiplication	
/	integer division	

- Both int and double data types can be used with these operators.
- ► The multiplication operator takes the form of an asterisk, and not the symbol ×.
- ► The / symbol performs integer division, where the decimal component of the result is discarded.

# The Modulus Operator

### Determining the remainder: %

The operation a%b produces the remainder, when operand a is divided by operand b.

- ▶  $17\%3 \rightarrow 2$
- $> 23\%5 \rightarrow 3$

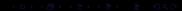
#### Determining even or odd numbers

Take any number and perform a modulus with 2.

- ▶ If the result is  $1 \rightarrow$  the number is odd.
- ▶ If the result is  $0 \rightarrow$  the number is even.

#### Example

 $15\%2 \rightarrow 1$ , therefore 15 is an odd number.



#### Division Behavior

### Integer division

If both of the operands are integers, then integer division is performed, where the decimal component of the result is discarded.

$$10/4 \rightarrow 2$$

#### Floating-point division

If either of the operands is a double, then a regular, calculator-style division is performed.

$$10.0/4 \rightarrow 2.5$$

$$16/5.0 \rightarrow 3.2$$

## Operator Precedence

All expressions are solved according to the same order of operations used in algebra.

```
int result = 14 + 8 / 2;
```

You can change the order of evaluation by using parentheses:

```
int result = (14 + 8) / 2;
```

After the arithmetic operations are complete, the answer is stored in the variable on the left-hand side of the assignment operator.

#### Precedence Table

Precedence	Operator	Operation	Association
1	()	grouping	N/A
	*	multiplication	
2	/	division	left to right
	%	modulus	
3	+	addition	left to right
	_	subtraction	
4	=	assignment	right to left

#### Data Conversion

### Converting numbers

In Java, we are allowed to convert from one numerical primitive data type to another. There are 2 categories of conversion:

#### Widening conversion

This is safest, because information is not lost.

$$\mathtt{int} o \mathtt{double}$$

#### Narrowing conversion

In this scenario, the decimal component of the double number is discarded. It should be avoided, because information is lost(in fact, the compiler will issue a warning).

$$exttt{double} o exttt{int}$$

### Type Casting

A type cast is used to convert a variable from one data type to another. Place the type name in parentheses, in front of the variable to be converted.

Widening conversion(int  $\rightarrow$  double)

```
int sum = 8;
double total = 0.0;
total = (double) sum; // total now contains 8.0
```

Narrowing conversion(double o int)

```
double money = 84.69;
int dollars = 0;
dollars = (int) money; // dollars now contains 84
```

### Updated Precedence Table

Precedence	Operator	Operation	Association
1	()	grouping	N/A
2	(int)	type cast	right to left
	(double)		
	*	multiplication	
3	/	division	left to right
	%	modulus	
4	+	addition	left to right
	_	subtraction	
5	=	assignment	right to left

# Adding or Subtracting 1 from a Variable

#### Increment operator: ++

This adds 1 to any numerical value.

```
int count = 5;
count++; // count now contains 6
```

#### Decrement operator: --

This subracts 1 from any numerical value.

```
int total = 5;
total--; // total now contains 4
```

### Compound Assignment Operators

#### The += operator

Several assignment operators in Java combine a basic operation with assignment. For example, the += operator can be used as follows:

```
int score = 10;
score += 5;
```

The code above causes the value of score to be increased by 5. The code above is equivalent to the following:

```
int score = 10;
score = score + 5;
```

## Java's Compound Assignment Operators

Op.	Description	Example	Equivalence
=	assignment	x = y	x = y
+=	addition & assignment	x += y	x = x + y
-=	subtraction & assignment	x -= y	x = x - y
*=	multiplication & assignment	x *= y	x = x * y
/=	division & assignment	x /= y	x = x / y
%=	remainder & assignment	x %= y	x = x % y