ORACLE
Academy

Java Foundations

3-2
Numeric Data

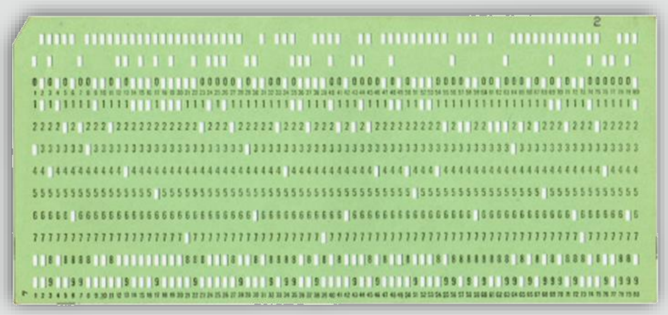
ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Differentiate integer data types (byte, short, int, long)
  - Differentiate floating point data types (float, double)
  - Manipulate and do math with numeric data
  - Use parentheses and order of operations

A Bit About Data

- In the early days of computing, data was stored on punch cards



- Each slot had 2 possible states:
  - Punched
  - Not punched

Reading Punch Card Data

- An AutoPiano reads punch cards
- A column represents a key on the piano
- The punch card scrolls through the piano, triggering keys
- Each slot has 2 possible states with 2 possible results:

An 1800s piano roll

State	Result
Punched	Play note
Not punched	Don’t play note

ORACLE
Academy

A Bit About Modern Computing

- Modern data processing still needs to represent 2 states:
  - This is interpreted as binary code: **10011101**
  - A single **1** or **0** is called a bit

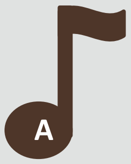
	AutoPiano	**Modern Computing**
Bit	Hole punched/Not punched	1/0
Bits are instructions for …	Mechanical components	The processor
Medium	Mechanical	Electro-Magnetism
Bits store data about…	Piano keys	Numbers

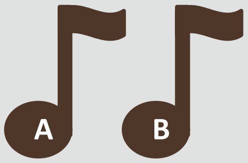
Let's take a closer look at this

ORACLE Academy

Bits of Data

- One AutoPiano key is represented by 1 bit
  - 0: Don’t play
  - 1: Play
- Two keys require 2 bits
  - There are 4 possible combinations of keys
  - We can calculate this as 2^2



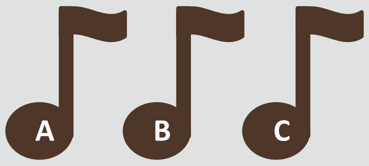


	A key	B key
Silence	0	0
B only	0	1
A only	1	0
Both A and B	1	1

ORACLE
Academy

Bigger Bits of Data

- Three keys require 3 bits
  - There are 8 possible combinations of keys
  - We can calculate this as 2^3
- Eight keys require 8 bits
  - There are 256 possible combinations
  - We can calculate this as 2^8

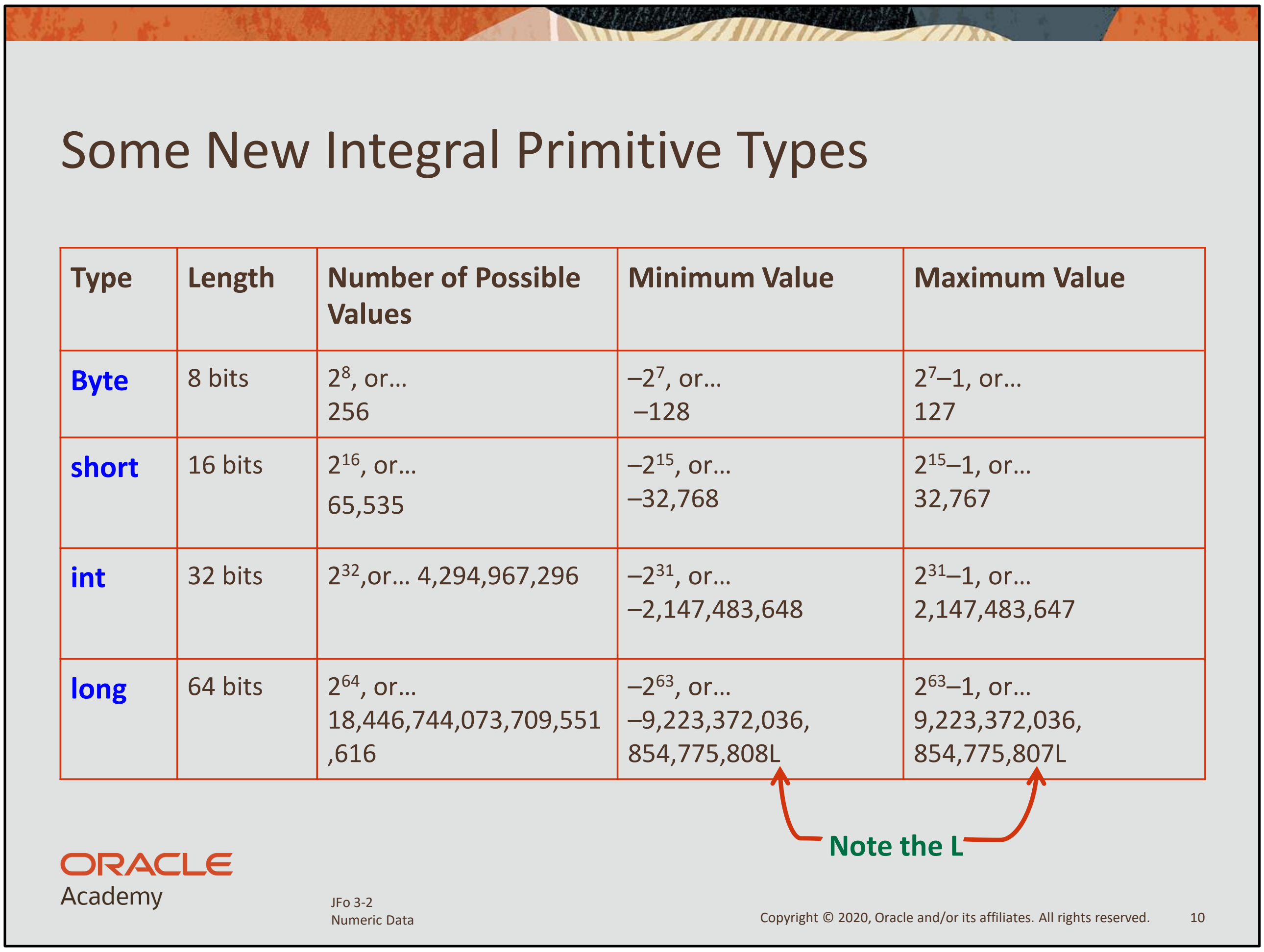


A key	B key	C key
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Bits and Bytes

- Eight bits are called a byte
- A Java byte can store 256 possible values
  - Possible values are from -128 to 127
  - 128 values below 0
  - 127 values above 0
  - 1 value equal to 0

✓
```
byte x = 127;
```
✗
```
byte z = 128;     //Too high
```



Some New Integral Primitive Types

Type	Length	Number of Possible Values	Minimum Value	Maximum Value
Byte	8 bits	2^8, or… 256	-2^7, or… −128	2^7-1, or… 127
short	16 bits	2^{16}, or… 65,535	-2^{15}, or… −32,768	$2^{15}-1$, or… 32,767
int	32 bits	2^{32},or… 4,294,967,296	-2^{31}, or… −2,147,483,648	$2^{31}-1$, or… 2,147,483,647
long	64 bits	2^{64}, or… 18,446,744,073,709,551 ,616	-2^{63}, or… −9,223,372,036, 854,775,808L	$2^{63}-1$, or… 9,223,372,036, 854,775,807L

Note the L

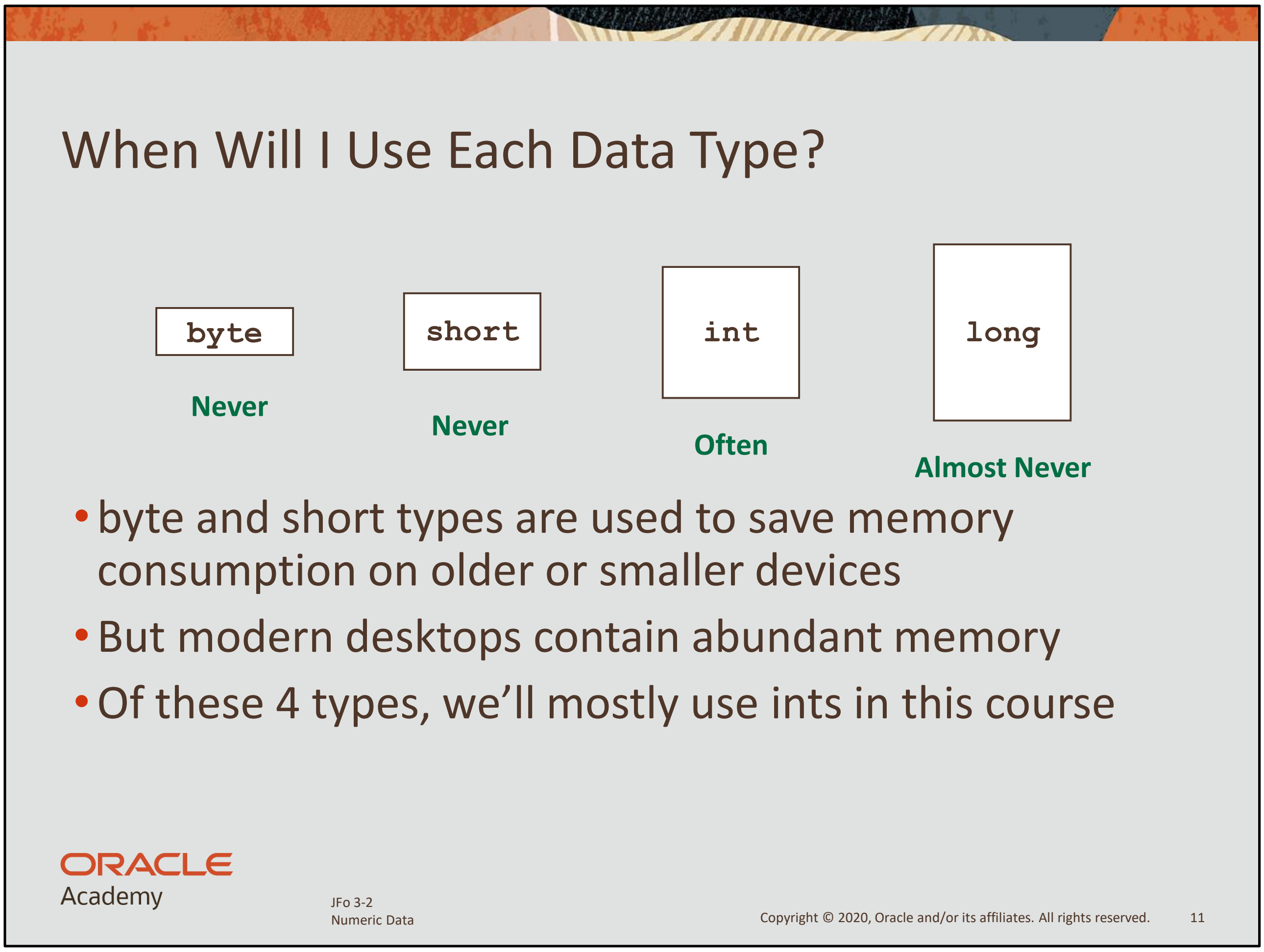
ORACLE Academy

JFo 3-2
Numeric Data

10

There are four integral (whole number) primitive types in the Java programming language. You have already been using the int data type, so let's focus on the other three. Integral types are used to store numbers that don't have decimal portions. They're shown here in order of size.

byte: If you need to store people's ages, a variable of type byte would work because byte types can accept values in that range.
short: A short will hold 16 bits of data.
long: When you specify a literal value for a long type, put a capital L to the right of the value to explicitly state that it's a long type. Integer literals are assumed by the compiler to be of type int unless you specify otherwise by using an L indicating long type.
You can express any of the integral types as binary (0s and 1s). For instance, a binary expression of the number 2 is shown as an allowed value of the byte integral type. The binary value is 0b10. Notice that this value starts with 0b (that is, zero followed by either a lowercase or uppercase letter *B*). The letter indicates to the compiler that a binary value follows.



When Will I Use Each Data Type?

byte	short	int	long
Never	**Never**	**Often**	**Almost Never**

- byte and short types are used to save memory consumption on older or smaller devices
- But modern desktops contain abundant memory
- Of these 4 types, we'll mostly use ints in this course

ORACLE Academy

JFo 3-2
Numeric Data

11

Examples of allowed literal values:

byte = 2, -114, 0b10 (binary number)
short = 2, -32699
int (default type for integral literals) = 2, 147334778, 123_456_678
long = 2, –2036854775808L, 1

Find x

```
int x = 20;
x = 25;
x = 5 + 3;

System.out.println();
```

- x always equals 20 …
  - Until you assign x a different value
- x could be assigned a calculated value

Values for `x`: ~~20~~ ~~25~~ 8

ORACLE Academy

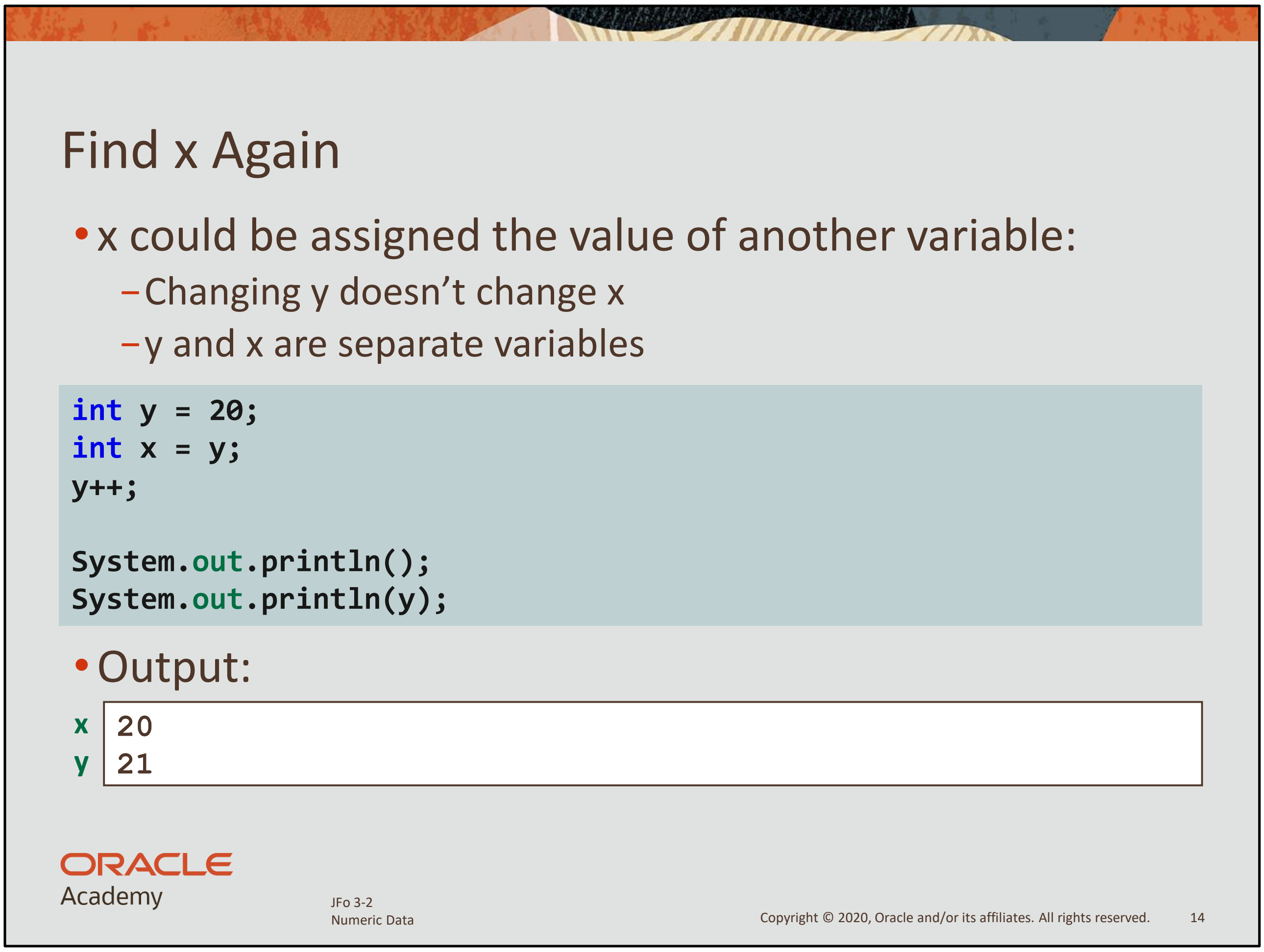
Find x

```
int x = 20;
x = 25;
x = 5 + 3;
x = x + 1;
x += 1;
x++;
System.out.println();
```

- x could be assigned a new value based on its current value:
  - Java provides the shorthand += operator to do this
  - Adding 1 to a variable is so common that Java provides the shorthand ++ operator

Values for x: ~~20~~ ~~25~~ ~~8~~ ~~9~~ ~~10~~ 11

ORACLE Academy



Find x Again

- x could be assigned the value of another variable:
  - Changing y doesn’t change x
  - y and x are separate variables

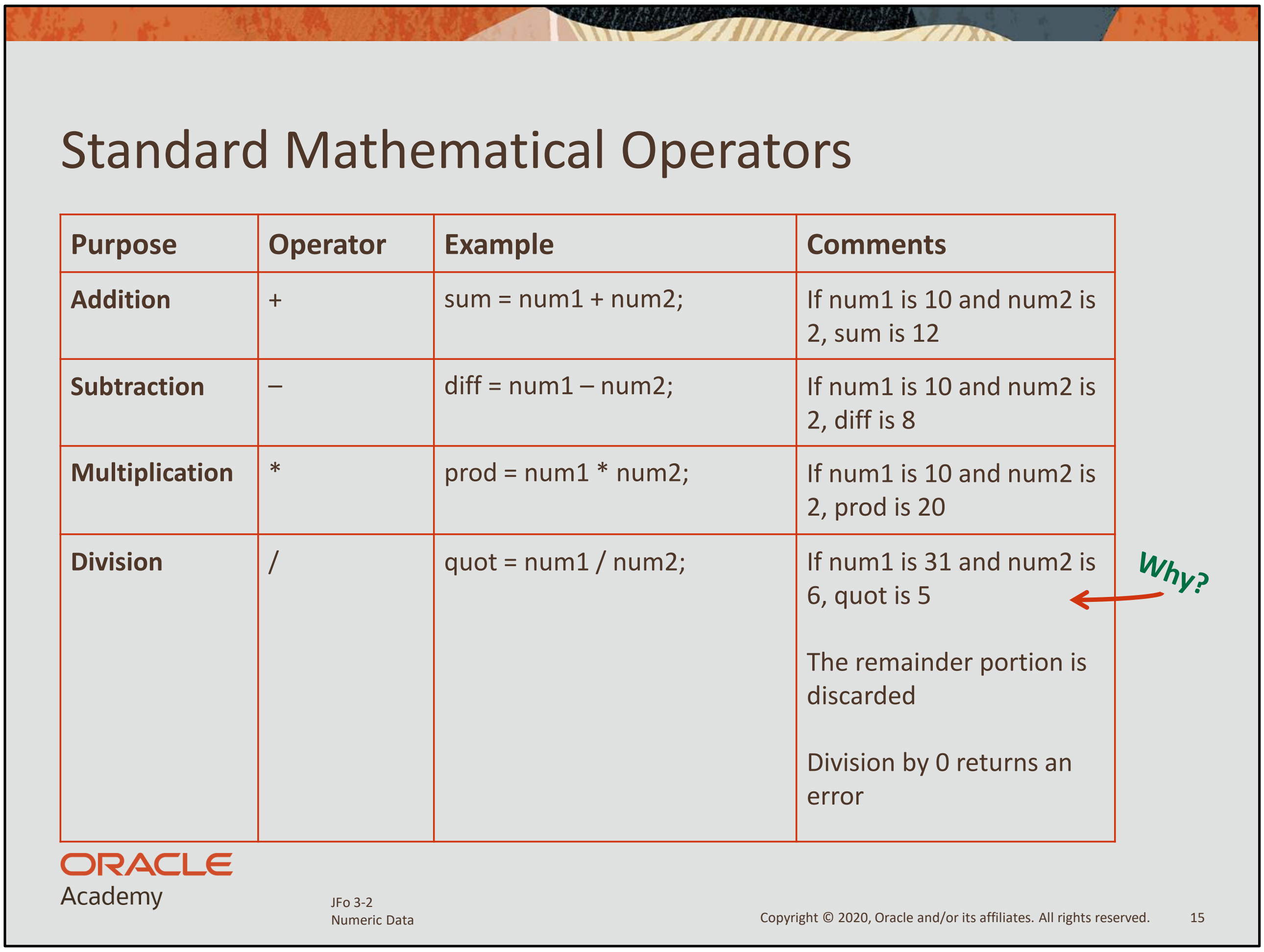
```
int y = 20;
int x = y;
y++;

System.out.println();
System.out.println(y);
```

- Output:

```
x  20
y  21
```

ORACLE Academy



Standard Mathematical Operators

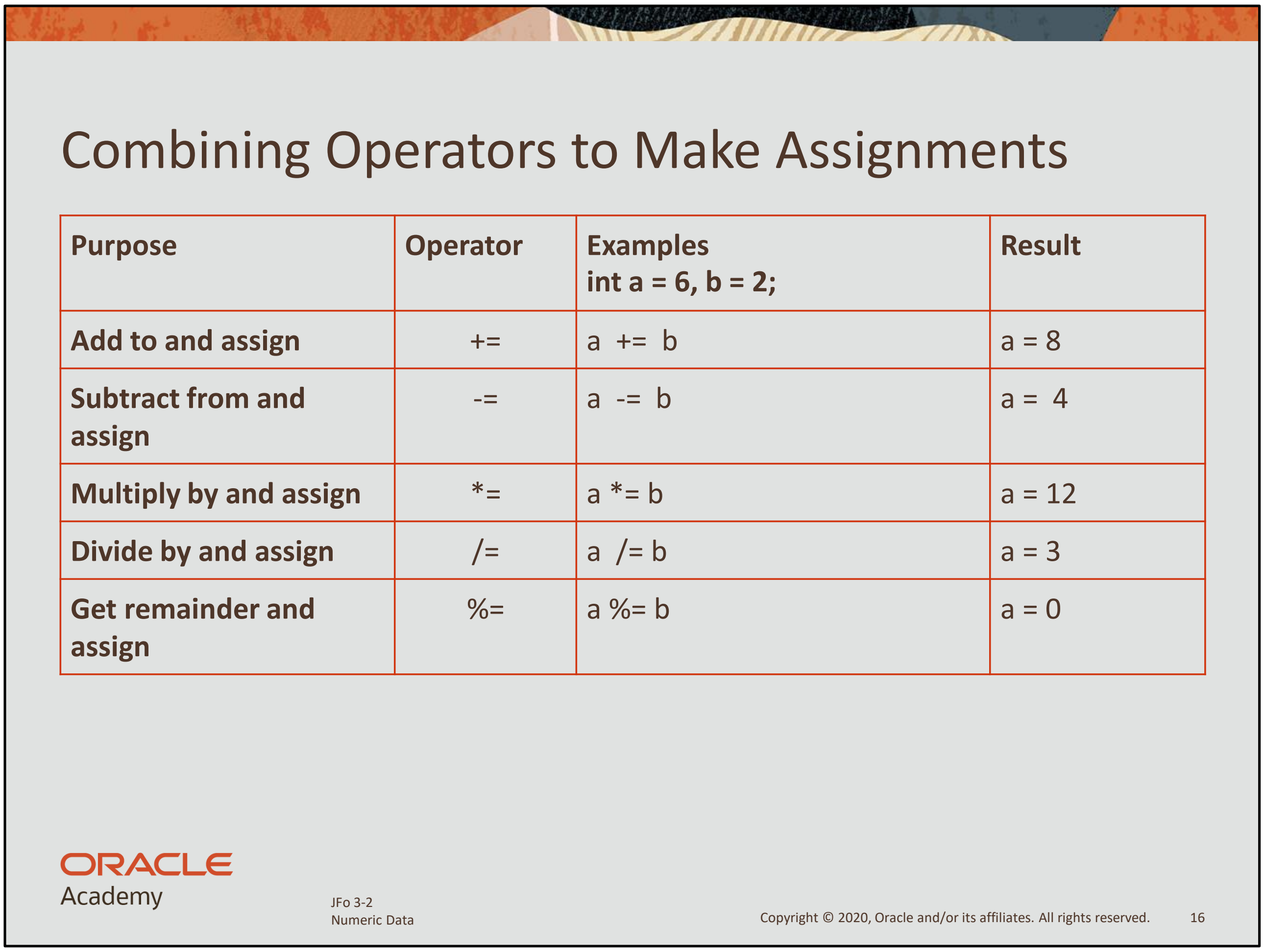
Purpose	Operator	Example	Comments
Addition	+	sum = num1 + num2;	If num1 is 10 and num2 is 2, sum is 12
Subtraction	–	diff = num1 – num2;	If num1 is 10 and num2 is 2, diff is 8
Multiplication	*	prod = num1 * num2;	If num1 is 10 and num2 is 2, prod is 20
Division	/	quot = num1 / num2;	If num1 is 31 and num2 is 6, quot is 5 The remainder portion is discarded Division by 0 returns an error

Why?

ORACLE
Academy

The table in the slide assumes that all operands and result variables are integers (int). Mixing doubles and ints can alter the results. For instance, in the division example, if the quotient and dividend (or if all three) are doubles, the quotient would show the decimal remainder portion:

```
double quot, num1;
num1 = 31;
int num2 = 5;
quot = num1 / num2;
Answer: quot = 6.2
```



Combining Operators to Make Assignments

Purpose	Operator	Examples int a = 6, b = 2;	Result
Add to and assign	+=	a += b	a = 8
Subtract from and assign	-=	a -= b	a = 4
Multiply by and assign	*=	a *= b	a = 12
Divide by and assign	/=	a /= b	a = 3
Get remainder and assign	%=	a %= b	a = 0

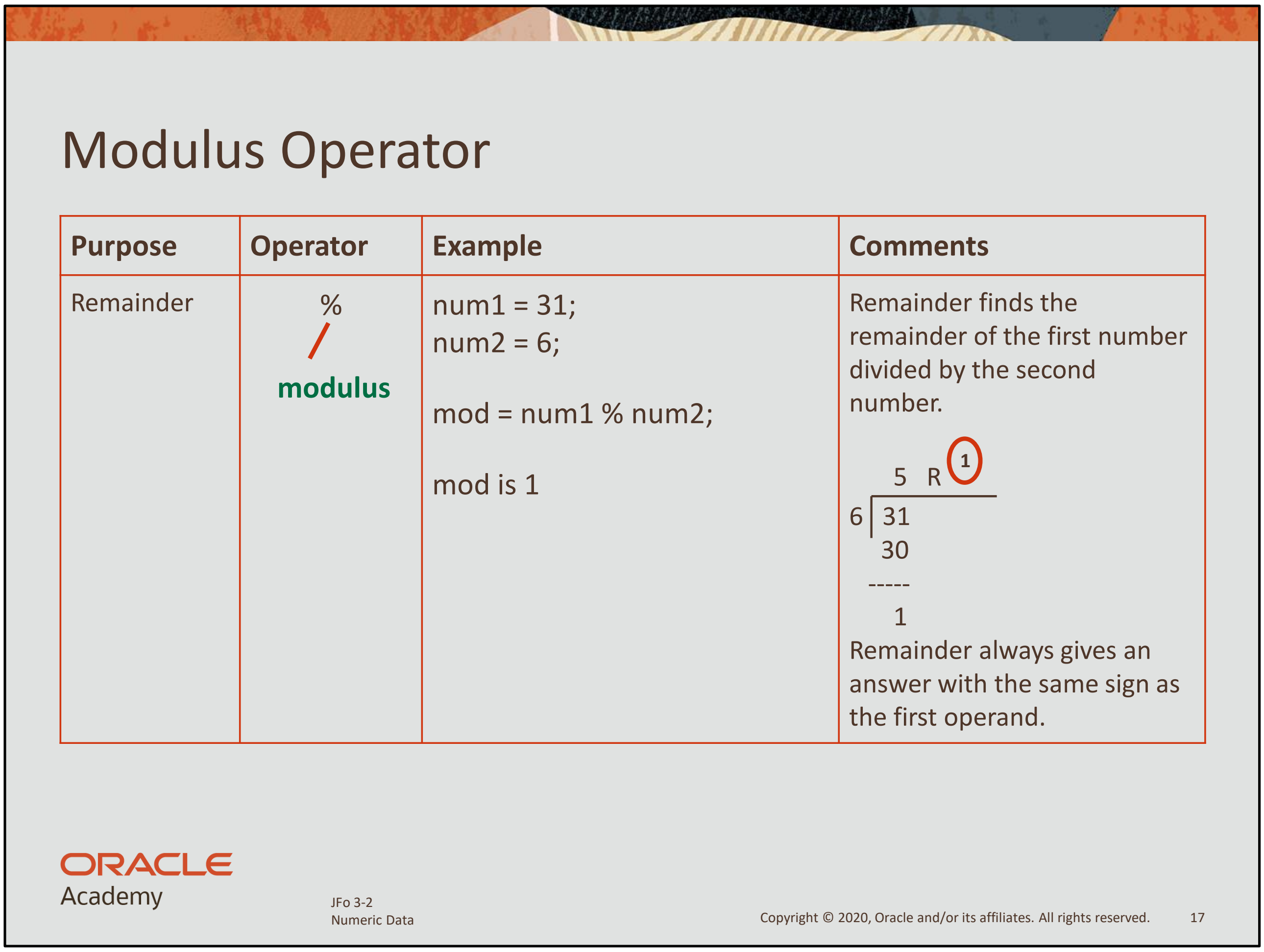
ORACLE
Academy

Several very useful shortcuts are shown in the table. You can combine any operator with the equal sign to abbreviate your code. For example:

a = a + b;

can be expressed as:

a += b;



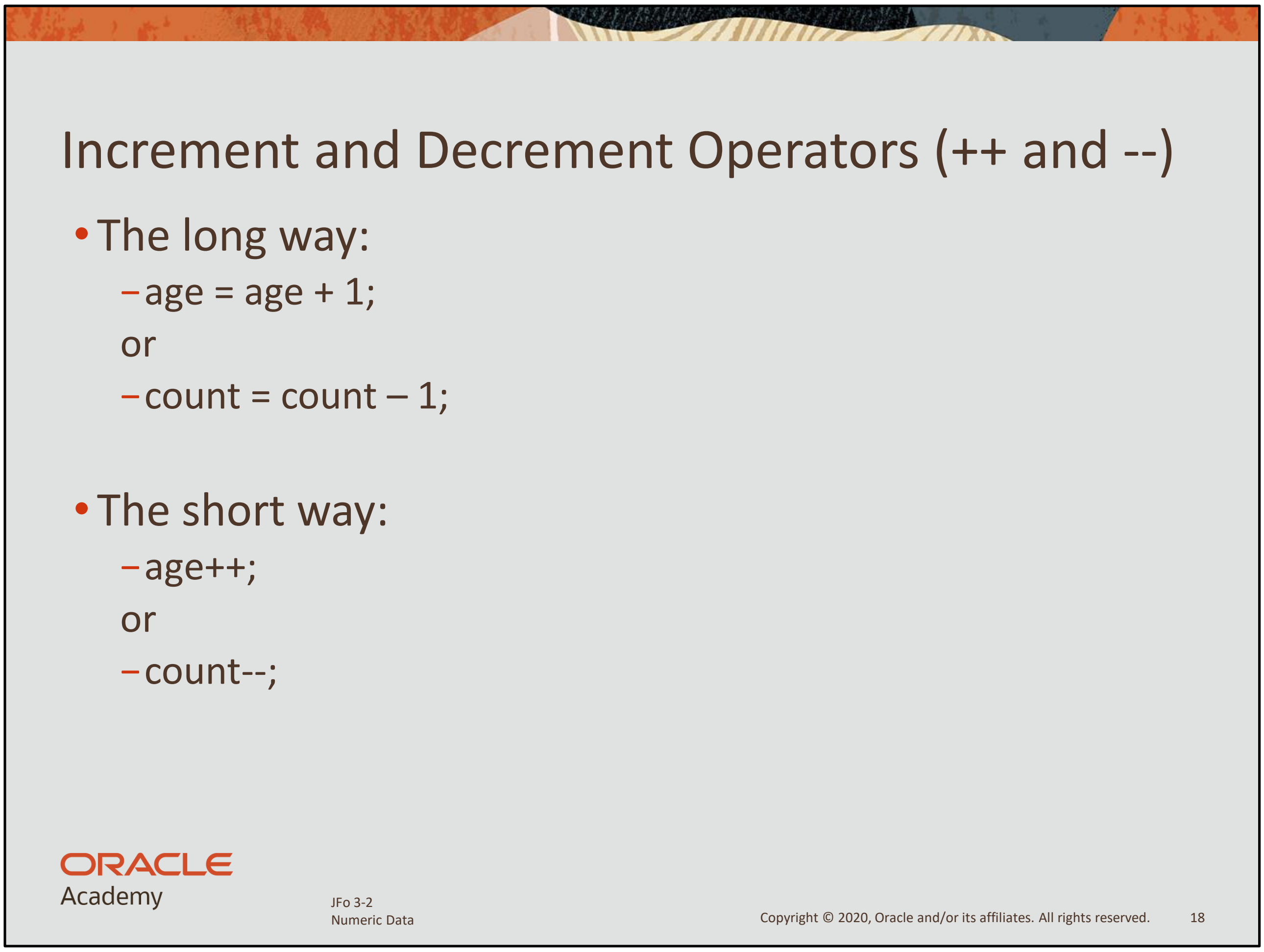
Modulus Operator

Purpose	Operator	Example	Comments

Remainder	%
	modulus
		num1 = 31;
		num2 = 6;
		
		mod = num1 % num2;
		
		mod is 1
			Remainder finds the remainder of the first number divided by the second number.
			
			5 R 1
			6 | 31
			30
			----- 	
			1
			Remainder always gives an answer with the same sign as the first operand.

ORACLE Academy

Programs do a lot of mathematical calculating, from the simple to the complex. Arithmetic operators let you specify how the numerical values within variables should be evaluated or combined. The standard mathematical operators (often called *binary operators*) used in the Java programming language are shown in the tables in this section.
Note: The % is known as a modulus.



Increment and Decrement Operators (++ and --)

- The long way:
  - age = age + 1;

  or
  - count = count – 1;
- The short way:
  - age++;

  or
  - count--;

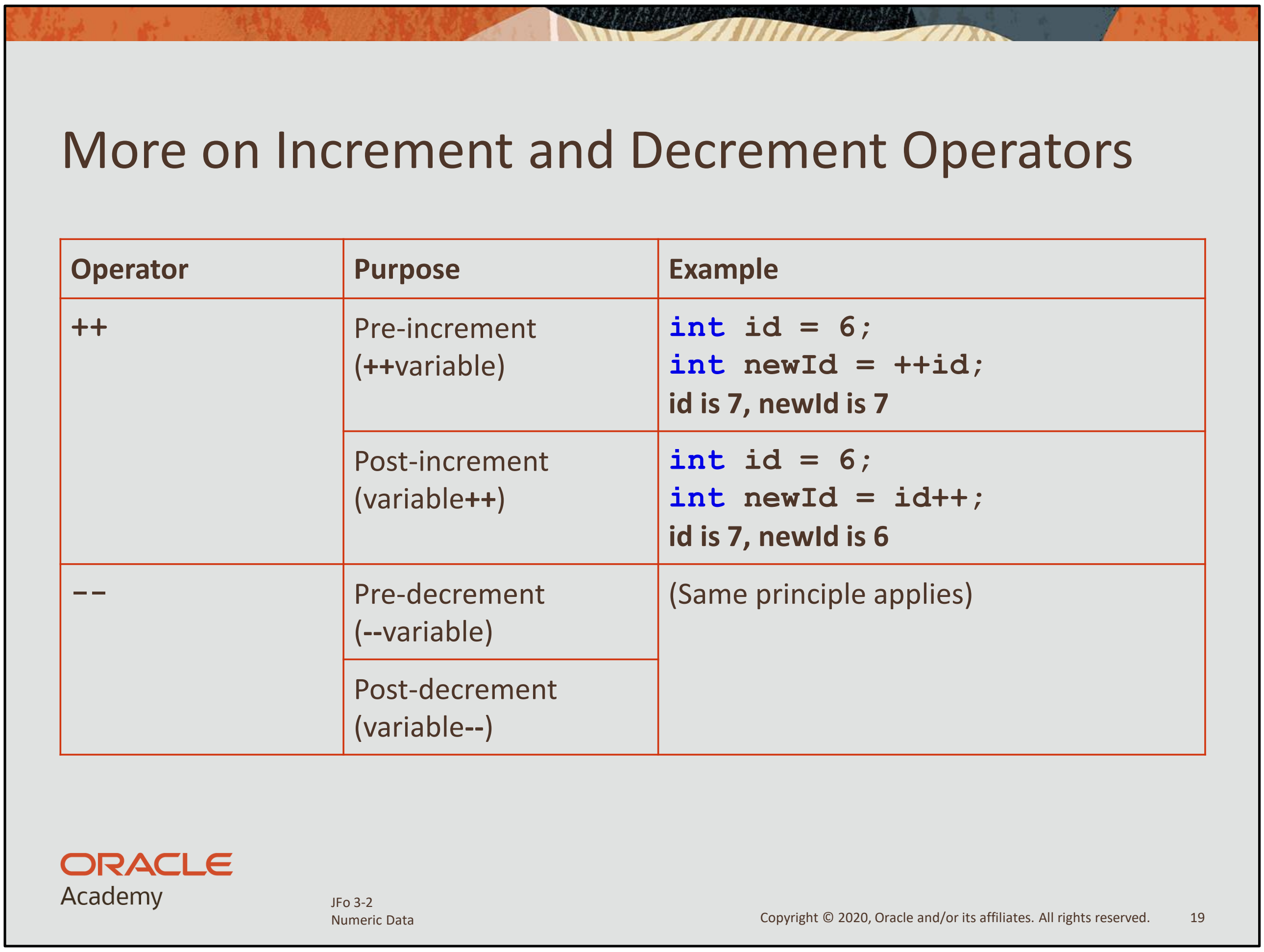
ORACLE Academy

JFo 3-2
Numeric Data

18

A common requirement in programs is to add or subtract 1 from the value of a variable. You can do this by using the + operator as follows:

age = age + 1;



More on Increment and Decrement Operators

Operator	Purpose	Example

`++`	Pre-increment (++variable)	`int id = 6;`
		`int newId = ++id;`
		id is 7, newId is 7

	Post-increment (variable++)	`int id = 6;`
		`int newId = id++;`
		id is 7, newId is 6

`--`	Pre-decrement (--variable)	(Same principle applies)
	Post-decrement (variable--)	

ORACLE Academy

JFo 3-2
Numeric Data

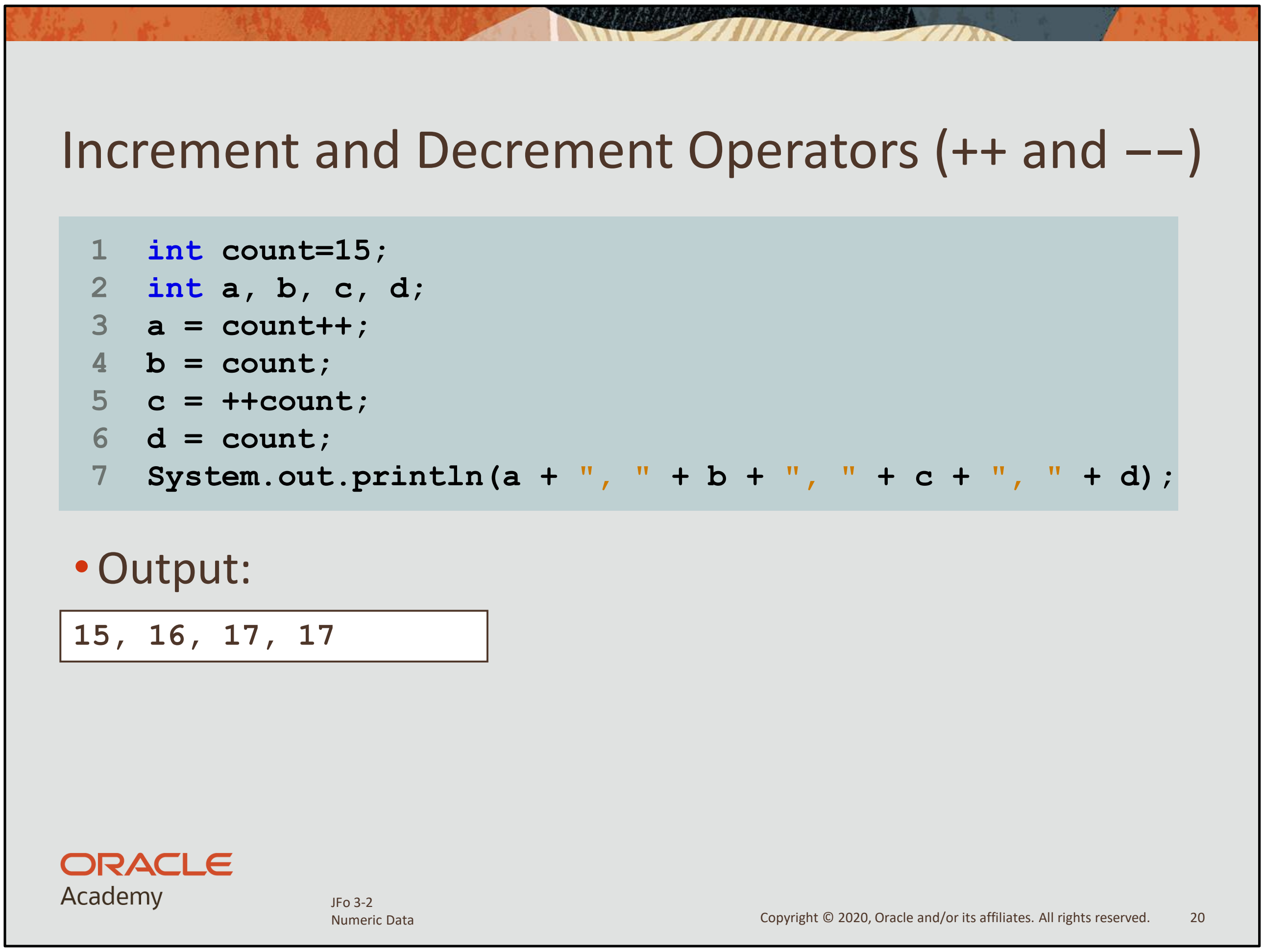
19

You have used increment and decrement operators before, placing them after the variable that you want to affect. But did you know that these operators can come before (pre-increment and pre-decrement) or after (post-increment and post-decrement) a variable.
When you put the ++ or –- operator before a variable, the value is changed immediately. When you put the operator after the variable, it isn't changed until after that expression is evaluated.

In the first code example in the slide, id is initialized to 6. In the next line, you see newId = ++id. Because the operator precedes id, this increment is immediately evaluated and the value assigned to newId is 7.

In the second code example, the ++ operator follows id. id was incremented after the assignment occurred. Therefore, newId is 6.

These same behaviors apply to a decrement (--) operator for its placement before or after the variable.



Increment and Decrement Operators (++ and ––)

```
int count=15;
int a, b, c, d;
a = count++;
b = count;
c = ++count;
d = count;
System.out.println(a + ", " + b + ", " + c + ", " + d);
```

- Output:

```
15, 16, 17, 17
```

ORACLE Academy

JFo 3-2
Numeric Data

The slide example shows basic use of the increment and decrement operators:

```
int count=15;
int a, b, c, d;
a = count++;
b = count;
c = ++count;
d = count;
System.out.println(a + ", " + b + ", " + c + ", " + d);
```

The result of this code fragment is:

```
15, 16, 17, 17
```

What is the result of the following code?

```
int i = 16;
System.out.println(++i + " " + i++ + " " + i);
```

Exercise 1, Part 1

- Import and edit the `Chickens01` project
- Read this story and calculate/print the totalEggs collected between Monday and Wednesday:
  - Farmer Brown's chickens always lay eggsPerChicken eggs precisely at noon, which he collects that day
  - On Monday, Farmer Brown has chickenCount chickens
  - On Tuesday morning, Farmer Brown gains 1 chicken
  - On Wednesday morning, a wild beast eats half the chickens!
  - How many eggs did Farmer Brown collect if he starts with …
    - eggsPerChicken = 5, chickenCount = 3
    - eggsPerChicken = 4, chickenCount = 8

ORACLE
Academy

Exercise 1, Part 2

- Your program should produce the following output:

```
45    First scenario
84    Second scenario
```

Integer Division Deception

- The wild beast ate half the chickens
- When we divide 9 chickens in half, Java thinks 9/2 = 4
  - But 9/2 = 4.5
  - Shouldn't Java round up to 5?
  - What's going on here?

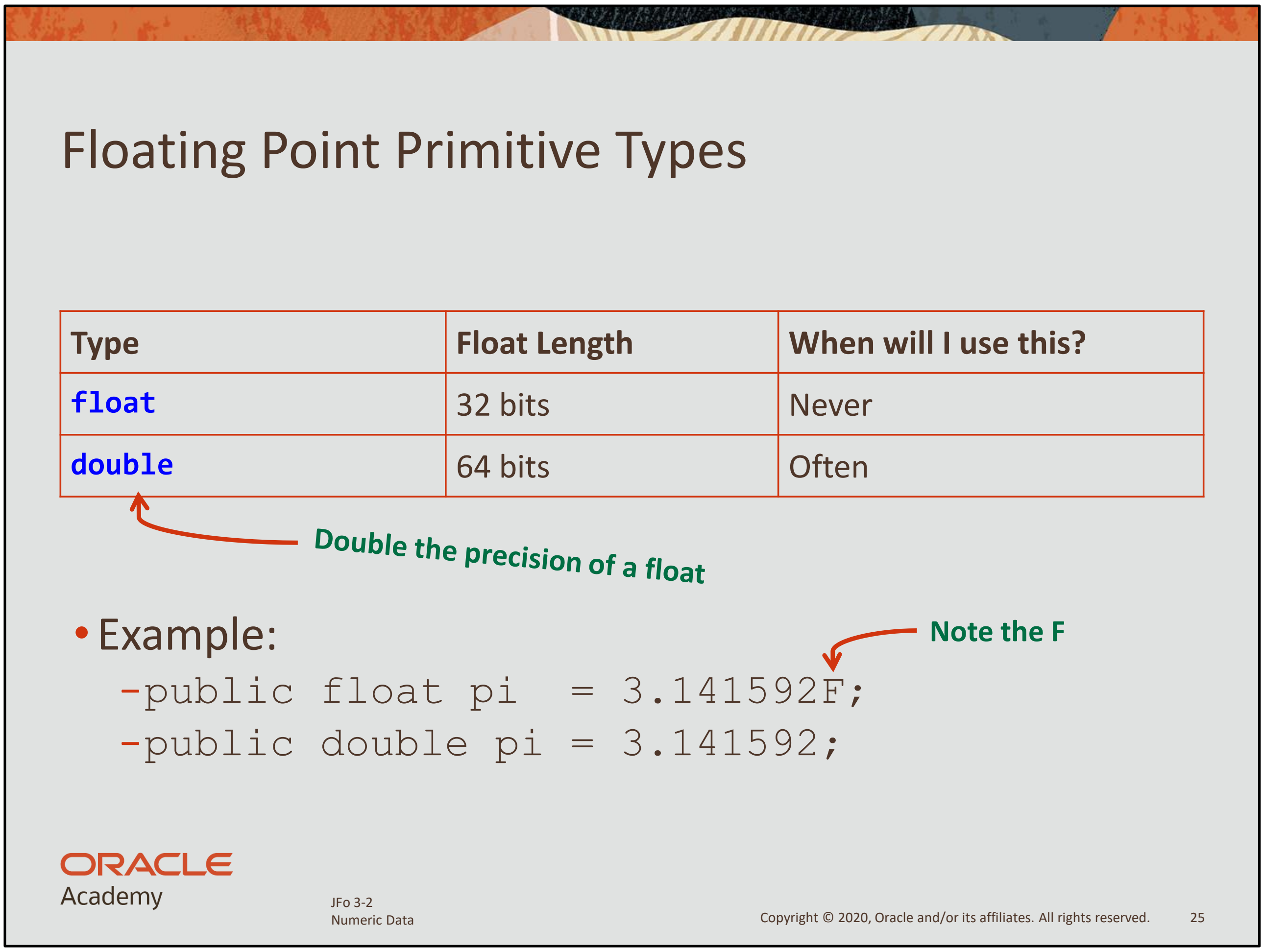
Academy

Java Division

- Java integers aren’t rounded
- Java integers are truncated, meaning any numbers after the decimal point are removed

```
int x = 9/2;
System.out.println(x);  //prints 4
```

- We need other data types if we have scenarios that require floating point precision!



Floating Point Primitive Types

Type	Float Length	When will I use this?
`float`	32 bits	Never
`double`	64 bits	Often

Double the precision of a float

- Example:

Note the F

```
-public float pi  = 3.141592F;
-public double pi = 3.141592;
```

ORACLE Academy

JFo 3-2
Numeric Data

25

There are two types for floating point numbers: float and double. Again, we will focus on the new data type here, the float. Floating point types are used to store numbers with values to the right of the decimal point, such as 12.24 or 3.14159.

float is used to store smaller floating point numbers. A float variable can hold 32 bits. Floating point values are assumed to be of type double unless you explicitly state that it is a float type, not a double type. You do that by putting a capital F (float) to the right of the value.

Here are examples of allowed literal values:

float = 99F, -327456,99.01F, 4.2E6F (engineering notation for 4.2 * 106)

double = -1111, 2.1E12, 99970132745699.999

Note: Use the double type when a greater range or higher accuracy is needed.

Double Deception

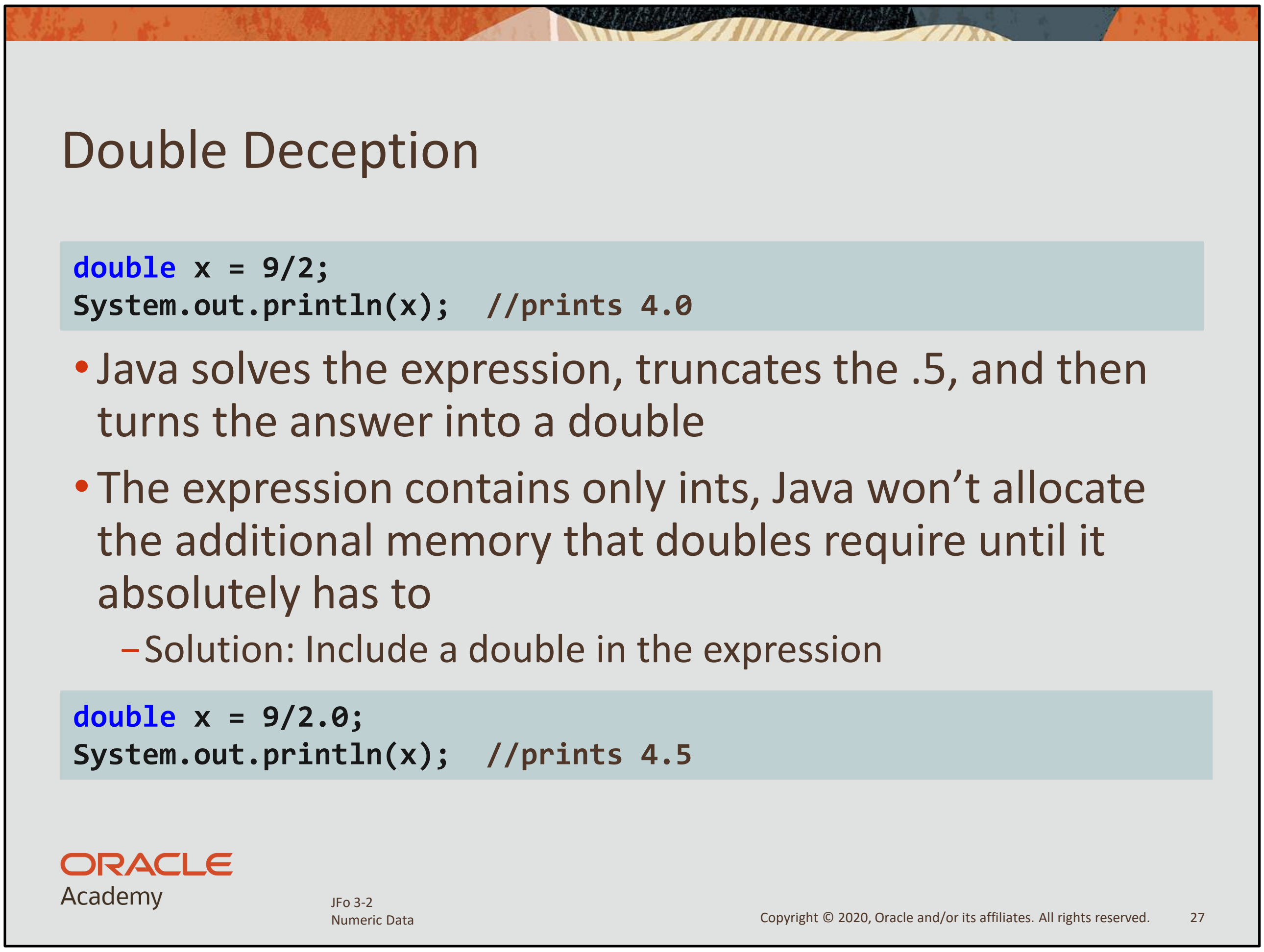
- The original problem:

```
int x = 9/2;
System.out.println(x);  //prints 4
```

- Shouldn’t a double x fix this?

```
double x = 9/2;
System.out.println(x);  //prints 4.0
```

  - No?!?!
  - Why not?



Double Deception

```java
double x = 9/2;
System.out.println(x);  //prints 4.0
```

- Java solves the expression, truncates the .5, and then turns the answer into a double
- The expression contains only ints, Java won’t allocate the additional memory that doubles require until it absolutely has to
  - Solution: Include a double in the expression

```java
double x = 9/2.0;
System.out.println(x);  //prints 4.5
```

ORACLE
Academy

One advantage of Java is that it handles memory management for you. More on this later.

One Final Note

- Declare a variable with the final keyword to make its value unchangeable (immutable)

```
final double PI = 3.141592;
PI = 3.0;        //Not Allowed
```

- Java complains if you try to change a final variable's value
- Final variable naming conventions:
  - Capitalize every letter
  - Separate words with an underscore
    - MINIMUM_AGE
    - SPEED_OF_LIGHT

ORACLE
Academy

Exercise 2, Part 1

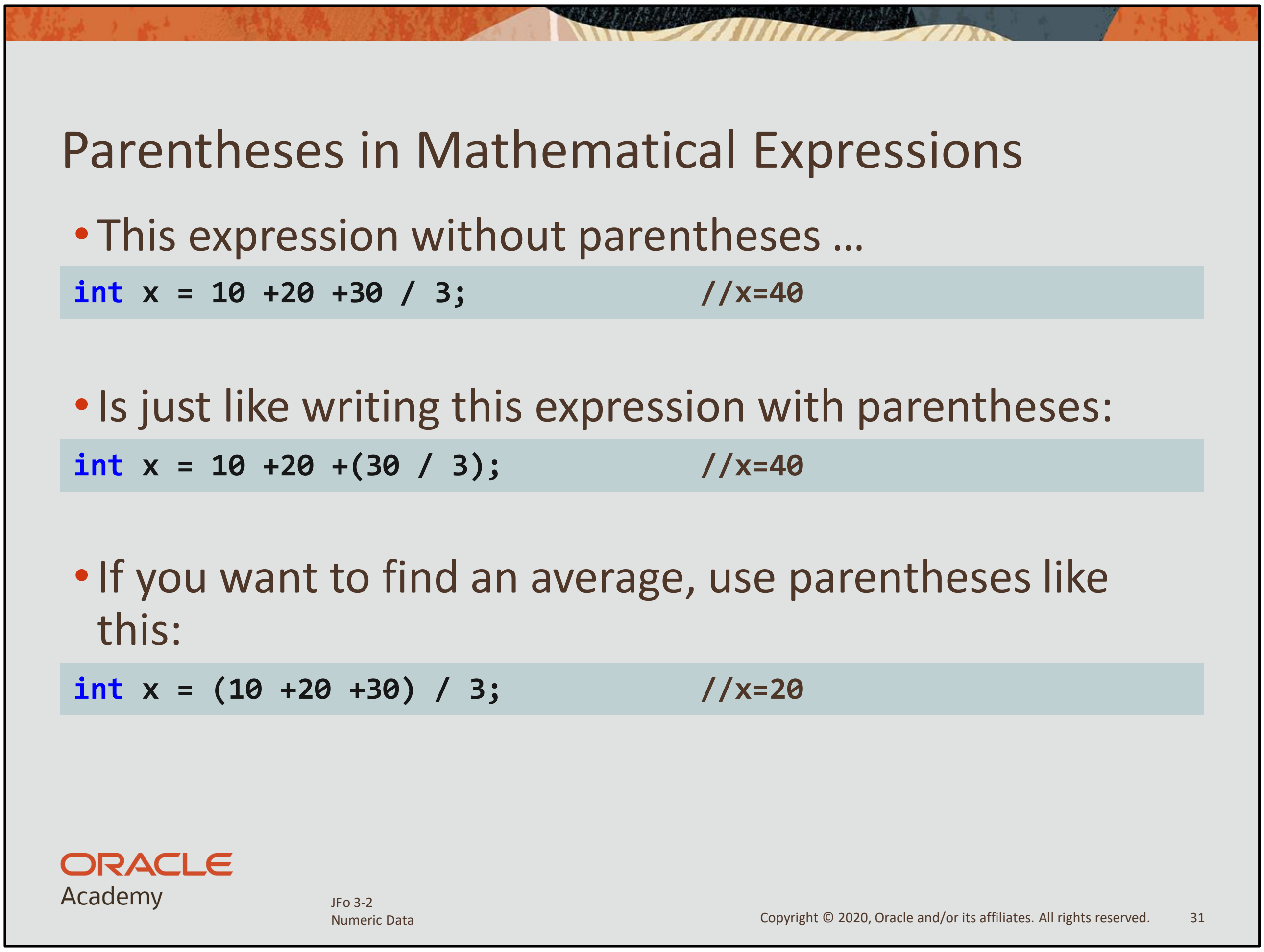
- Import and edit the `Chickens02` project
- Read this story and calculate/print the required values:
  - On Monday, Farmer Fred collects 100 eggs
  - On Tuesday, Farmer Fred collects 121 eggs
  - On Wednesday, Farmer Fred collects 117 eggs
  - What is the dailyAverage of eggs collected?
  - How many eggs could be expected in a 30-day monthlyAverage?
  - If an egg can be sold for a profit of $0.18, what is Farmer Fred’s total monthlyProfit for all eggs?

Exercise 2, Part 2

- Your program should produce the following output:

```
Daily Average:   112.66666666666667
Monthly Average: 3380.0
Profit:          $608.4
```

ORACLE
Academy



Parentheses in Mathematical Expressions

- This expression without parentheses …

```
int x = 10 +20 +30 / 3;              //x=40
```

- Is just like writing this expression with parentheses:

```
int x = 10 +20 +(30 / 3);            //x=40
```

- If you want to find an average, use parentheses like this:

```
int x = (10 +20 +30) / 3;            //x=20
```

ORACLE Academy

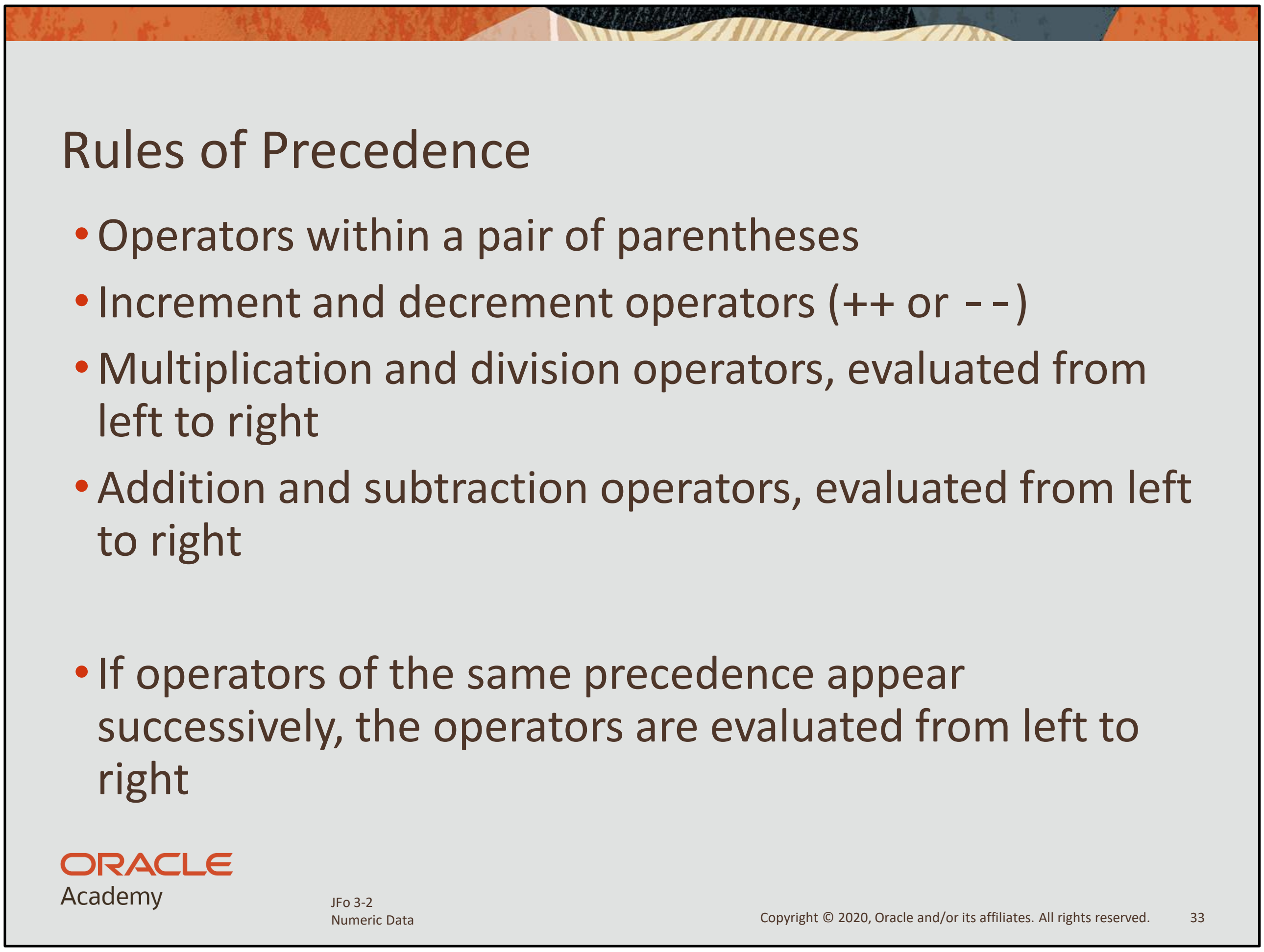
In Exercise 2, if you didn't use parentheses, you could have created an int with the total number of eggs collected over 3 days and then set dailyAverage = threeDayTotal/3.0. Or you could have divided each day's total by 3.0 individual and taken the sum.

Operator Precedence

- Here’s an example of the need for rules of precedence:

```java
int x = 25 - 5 * 4 / 2 - 10 + 4;
```

- Is the answer 34 or 9?



Rules of Precedence

- Operators within a pair of parentheses
- Increment and decrement operators (++ or - -)
- Multiplication and division operators, evaluated from left to right
- Addition and subtraction operators, evaluated from left to right

- If operators of the same precedence appear successively, the operators are evaluated from left to right

ORACLE Academy

JFo 3-2
Numeric Data

33

In a complex mathematical statement with multiple operators on the same line, how does the computer pick which operator it should use first? To make mathematical operations consistent, the Java programming language follows the standard mathematical rules for operator precedence.

Using Parentheses

- Expression are evaluated with the rules of precedence
- However, you should use parentheses to provide the intended structure
- Examples:

```
int x = (((25 - 5) * 4) / (2 - 10)) + 4;
int x = ((20 * 4) / (2 - 10)) + 4;
int x = (80 / (2 - 10)) + 4;
int x = (80 / -8) + 4;
Int x = -10 + 4;
int x = -6;
```

Summary

- In this lesson, you should have learned how to:
  - Differentiate integer data types (byte, short, int, long)
  - Differentiate floating point data types (float, double)
  - Manipulate and do math with numeric data
  - Use parentheses and order of operations

ORACLE
Academy

ORACLE
Academy