

# 4

## Data in a Cart

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the purpose of a variable in the Java language
- List and describe four data types
- Declare and initialize `String` variables
- Concatenate `String` variables with the '+' operator
- Make variable assignments
- Declare and initialize `int` and `double` variables
- Modify variable values by using numeric operators
- Override default operator precedence using ( )



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Variables

- A variable refers to something that can change.
  - Variables can be initiated with a value.
  - The value can be changed.
  - A variable holds a specific type of data.

Diagram illustrating the components of a variable declaration in Java:

```
String firstName = "Mary";
```

The diagram shows the code snippet above with three labels and lines pointing to specific parts:

- The type of data** points to `String`.
- Variable name** points to `firstName`.
- The value of the variable** points to `"Mary"`.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A variable is simply a storage location in memory that holds a specific value. That value can be changed by copying (or “assigning”) a different value to that variable.

# Variable Types

- Some of the types of values a variable can hold:
  - `String` (example: "Hello")
  - `int` (examples: -10, 0, 2, 10000)
  - `double` (examples: 2.00, 99.99, -2042.09)
  - `boolean` (true or false)
- If uninitialized, variables have a default value:
  - `String`: null
  - `int`: 0
  - `double`: 0.0
  - `boolean`: false

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Variables are declared to hold a specific type of data. Some of the more common types are:

- `String`: This is text data, such as "Hello".
- `int`: This is integer data—positive or negative whole numbers.
- `double`: These are positive or negative *real* numbers containing a decimal portion.
- `boolean`: This data type has a value of either true or false.

Most variables that have not been initialized are given a default value. The default values for `String`, `int`, `double`, and `boolean` are shown above. (Local variables are the exception. You will learn about local variables in the lesson titled "Creating and Using Methods.")

Notice that `String` begins with an uppercase letter, but the other types do not. You will learn the reason for this later, when you also learn about some other data types.

# Naming a Variable

## Guidelines:

- Begin each variable with a lowercase letter. Subsequent words should be capitalized:
  - `myVariable`
- Names are case-sensitive.
- Names cannot include white space.
- Choose names that are mnemonic and that indicate to the casual observer the intent of the variable.
  - `outOfStock` (a boolean)
  - `itemDescription` (a String)

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Uses of Variables

- Holding data used within a method:

```
String name = "Sam" ;  
double price = 12.35;  
boolean outOfStock = true;
```

- Assigning the value of one variable to another:

```
String name = name1;
```

- Representing values within a mathematical expression:

```
total = quantity * price ;
```

- Printing the values to the screen:

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Variables are used extensively in the Java programming language for tasks such as:

- Holding data used within a method, such as the `main` method
- Assigning the value of one variable to another. In the first example above, the `name` variable is initialized with the value, "Sam", and in the second example, its value is changed to the value of `name1` (unknown here).
- Representing values within a mathematical expression (\* is the symbol for multiplication)
- Printing the values to the screen. For example, the same `System.out.println` method that you used in the last exercise to print out the text literal, "Welcome to the Shopping Cart", can also be used to print out the value stored in the `name` variable.

# Topics

- Introducing variables
- **Working with String variables**
- Working with numbers
- Manipulating numeric data

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.



# Variable Declaration and Initialization

- Syntax :

```
type identifier [= value];
```

- Examples:

The diagram shows four examples of variable declarations and initializations, each enclosed in a blue box. Handwritten blue annotations with arrows point to specific parts of the code:

- Example 1: `String customer;` - Annotation: "Variable declared" (points to `String`).
- Example 2: `String name, city;` - Annotation: "Two variables declared" (points to `name, city`).
- Example 3: `String address = "123 Oak St";` - Annotation: "Variable declared and initialized" (points to `address`).
- Example 4: `String country = "USA", state = "CO";` - Annotation: "Two variables declared and initialized" (points to `country = "USA", state = "CO"`).

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The syntax for declaring and initializing a variable is:

```
type identifier [= value];
```

where:

- **type** represents the type of information or data held by the variable. In the examples above, you see only `String` variable types declared.
- **identifier** is the variable name. In the first example above, the variable name is `customer`.
- The second example shows how you can declare multiple variables of the same type on a single line without initializing them. Notice that they are separated by a comma.
- You can either declare a variable without assigning an initial value (as you saw in the first example) or you can initialize the variable at the same time you declare it (as shown in the example that declares and initializes the `address` variable).
- The final example shows how to declare and initialize multiple variables of the same type. Once again, each initialization is separated by a comma. Some of the multiple variables declared on a single line can be initialized, whereas others are just declared.

# String Concatenation

- String variables can be combined using the '+' operator.
  - `stringVariable1 + stringVariable2`
  - `stringVariable1 + "String literal"`
  - `stringVariable1 + "String literal" + stringVariable2`

- **Example:**

```
String greet1 = "Hello";  
String greet2 = "World";  
String message = greet1 + " " + greet2 + "!";  
String message = greet1 + " " + greet2 + " " + 2014 + "!";
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Combining multiple Strings is called “concatenation.” You can concatenate a `String` variable to another `String` variable. You can also concatenate a `String` literal to a `String` variable. As you can see in the example above, you can concatenate any number of `String` variables and `String` literals to achieve your goal.

You may find the last example surprising. You can also concatenate a number into a `String` variable. The compiler converts the numeric value to its equivalent `String` value. If we were to print the `message` variable after the last example, the output would be “Hello World 2014!”

## String Concatenation Output

You can concatenate `String` variables within a method call:

```
System.out.println(message);  
System.out.println(greet1 + " " + greet2 + "!");
```

### Output:

```
Hello World!  
Hello World!
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the examples above, you see two variations of printing out `String` data by using the `System.out.println` method.

- In the first example, the `message` variable that you saw in the previous slide will be printed.
- In the second example, the expression containing the concatenation of variables plus `String` literals can be used within the method parentheses. The concatenation will be completed by the runtime engine before the `println` method is executed.
- As you can see, the output of both method invocations is the same.

## Exercise 4-1: Using String Variables

In this exercise, you declare, initialize, and concatenate String variables and literals.

```
Exercise: 04-Variables, Exercise1 | Index | Solution

ShoppingCart.java

1 package ex04_1_exercise;
2
3 public class ShoppingCart {
4
5     public static void main(String[] args) {
6         // Declare and initialize String variables. Do not initialize message yet.
7
8
9
10
11
12
13
14         // Assign the message variable
15
16
17         // Print and run the code
18
19
20     }
21 }
22
23
```

Exercise 4-1:

1. Declare and initialize 2 String variables: `custName` and `itemDesc`.
2. Declare a String variable called `message`. Do not initialize it.
3. Assign the message variable with a concatenation of the `custName` and `itemDesc`. Include a String literal that results in a complete sentence. (example: "Mary Smith wants to purchase a Shirt")
4. Print the message to the System output.
5. Run the code.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- Open the Java Code Console and access 04-Variables > Exercise1.
- Follow the instructions below the code editor to declare and initialize variables, and also to concatenate a string message. Print the message by using `System.out.println`.

**Note:** If you need help, the solution for this exercise can be found by clicking the Solution link.

## Quiz

Which of the following variable declarations and/or initializations are correct?

- a. `int count = 5; quantity = 2;`
- b. `string name, label;`
- c. `boolean complete = "false";`
- d. `boolean complete = true;`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

### Answer: d

- a is incorrect because each `int` declaration and assignment must be separated by a comma and not a semicolon.
- b is incorrect because `String` is not capitalized.
- c is incorrect because a `boolean` type variable does not hold `String` values. It holds only `true` and `false`.
- d is correct.

# Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- Manipulating numeric data

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

## int and double Values

- `int` variables hold whole number values between:
  - `-2,147,483,648`
  - `2,147,483,647`
  - Examples: `2`, `1343387`, `1_343_387`
- `double` variables hold larger values containing decimal portions.
  - Use when greater accuracy is needed.
  - Examples: `987640059602230.7645` , `-1111`, `2.1E12`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- The `int` data type stores 32 bits of data. This means that you can store whole numbers within the range: `-2,145,483,648` and `2,147,483,647`. You cannot use commas to make the number more readable when you assign a value to an `int` variable. However, you can use underscores (`_`) to make your code more readable, as shown in one of above `int` examples. The compiler ignores these underscores. If you print the number to system output, the underscores will not appear. The only benefit of this is readability in your code.
- The `double` data type stores 64 bits of data. This means that you can store extremely large values—either negative or positive. The examples above show:
  - An extremely large number with four decimal points of precision
  - A negative whole number
  - A decimal number using exponential notation

## Initializing and Assigning Numeric Values

- **int variables:**

- `int quantity = 10;` ✓

- `int quantity = 5.5;` ✗

Compilation fails!

- **double variables:**

- `double price = 25.99;` ✓

- `double price = 75;` ✓

Run time will interpret as 75.0.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.



# Topics

- Introducing variables
- Working with `String` variables
- Working with numbers
- **Manipulating numeric data**

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Standard Mathematical Operators

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

## Increment and Decrement Operators (++ and --)

The long way:

```
age = age + 1;
```

or

```
count = count - 1;
```

The short way:

```
age++;
```

or

```
count--;
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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;
```

# Operator Precedence

Here is an example of the need for rules of precedence.

Is the answer to the following problem 34 or 9?

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

# Operator Precedence

Rules of precedence:

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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. Operators are processed in the following order:

1. Operators within a pair of parentheses
2. Increment and decrement operators
3. Multiplication and division operators, evaluated from left to right
4. Addition and subtraction operators, evaluated from left to right

If standard mathematical operators of the same precedence appear successively in a statement, the operators are evaluated from left to right.

# Using Parentheses

## Examples:

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Your expression will be automatically evaluated with the rules of precedence. However, you should use parentheses to provide the structure you intend:

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

## Exercise 4-2: Using and Manipulating Numbers

In this exercise, you declare, initialize, and concatenate `String` variables and literals.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- Open the Java Code Console and access Lessons > 04-Variables > Exercise2
- Follow the instructions below the code editor to:
  - Declare and initialize two `double` variables to hold the price and the tax
  - Declare an `int` to hold the number of items the shopper wishes to purchase
  - Declare another `double` variable, `total`, but do not initialize it yet
  - Change the existing message variable to include the number of items desired
  - Calculate the total cost and then print it with a suitable descriptive `String`
  - Run the project

**Note:** If you need help, the solution for this exercise can be found by clicking the Solution link.

## Quiz

Which of the following statements are correct Java code?

- a. `int count = 11.4;`
- b. `double amount = 11.05;`
- c. `int cost = 133_452_667;`
- d. `double total = 1.05 * amount;`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

**Answer: b, c, d**

- a is incorrect because the assignment of a decimal value to an `int` is a possible loss of precision and therefore will not compile.
- b is correct.
- c is correct because underscores can be used to make a number more readable.
- d is correct.



## Quiz

Given:

```
String name = "Bob";  
String msg;  
int num = 3;
```

Which of the following statements correctly assigns the value "Bob wrote 3 Java programs." to the msg variable?

- a. `msg = name + " wrote " + num " Java programs.";`
- b. `msg = name + " wrote " + 3 + " Java programs.";`
- c. `msg = "Bob wrote " + (2+1) + " Java programs.";`
- d. `msg = name + " wrote " + 2+1 + " Java programs.";`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

**Answer: b, c**

- a is incorrect because it is missing a + sign between the num variable and the final String literal.
- b is correct because the compiler converts the int of value 3 to a String.
- c is correct because, due to the use of parentheses, the addition operation is performed first, before the concatenation.
- d is incorrect because it would result in "Bob wrote 21 Java programs." The compiler converts each number to a String separately and concatenates them together.

# Summary

In this lesson, you should have learned how to:

- Describe the purpose of a variable in the Java language
- List and describe four data types
- Declare and initialize `String` variables
- Concatenate `String` variables with the '+' operator
- Make variable assignments
- Declare and initialize `int` and `double` variables
- Modify numeric values by using operators
- Override default operator precedence using ( )



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.