



Java Foundations

3-3

Textual Data



ORACLE ACADEMY

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Objectives

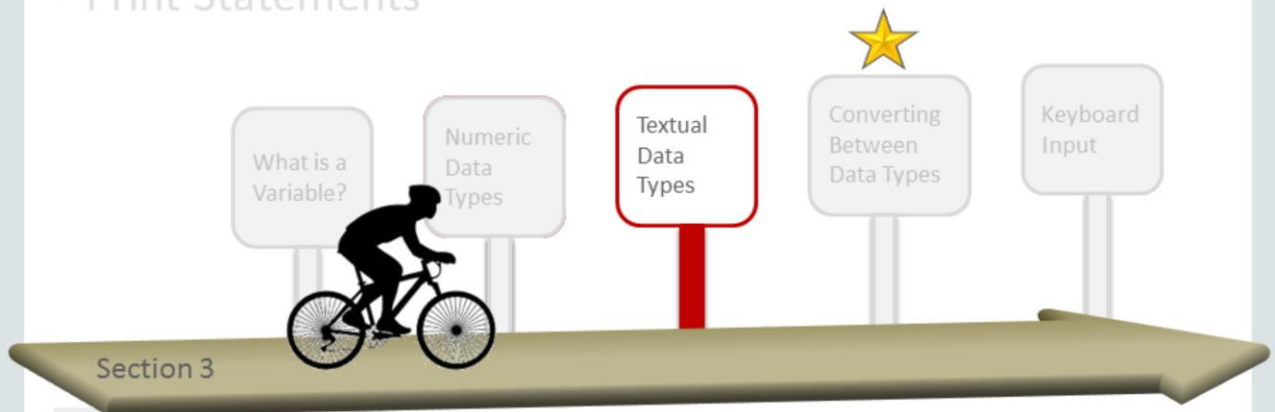
This lesson covers the following objectives:

- Use the `char` data type
- Use Strings
- Concatenate Strings
- Understand escape sequences
- Understand print statements better



Topics

- Characters and Strings
- String Concatenation
- Mixing Strings and Numbers
- Print Statements



Textual Primitive Type

- The only primitive textual data type is `char`.
- It's used for a single character (16 bits).

Example:

```
- char shirtSize = 'M';
```

Single quotes must be used with char literal values.

Another data type is used for storing and manipulating data as a single character. The `char` primitive type is 16 bits in size. When you assign a literal value to a `char` variable, you must use single quotation marks around the character, as shown in the code example.

Stringing Characters Together

You can string characters together to create sentences.

- Here's an inefficient way to do it.
- One line of code is required for every letter in a sentence.

```
char letter1 = 'H';  
char letter2 = 'e';  
char letter3 = 'l';  
char letter4 = 'l';  
char letter5 = 'o';  
//Long sentences would be painful to code  
System.out.println(letter1 +letter2 +letter3  
    +letter4 +letter5);
```

Stringing Characters Together Efficiently

Here's a better way

- Only one line is required for the entire sentence:

```
String greeting = "Hello World!";  
//Notice the double quotes  
System.out.println(greeting);
```

Characters vs. Strings

- `char`s are for a single character.
 - Use single quotation marks.



```
char shirt1Size = 'S';  
char shirt2Size = 'M';  
char shirt3Size = 'L';
```

- `char`s can't handle multiple characters.



```
char shirt4Size = 'XL';  
char shirt5Size = "XXL";
```


Characters vs. Strings

- A String can handle multiple characters.
 - Use double quotation marks.



```
String shirt6Size = "XXXL";
```

Primitives

Type	Length	Data
<code>boolean</code>	1 bit	<code>true</code> / <code>false</code>
<code>byte</code>	8 bits	Integers
<code>short</code>	16 bits	Integers
<code>int</code>	32 bits	Integers
<code>long</code>	64 bits	Integers
<code>float</code>	32 bits	Floating point numbers
<code>double</code>	64 bits	Floating point numbers
<code>char</code>	16 bits	Single characters

Where are Strings?

Let's Investigate

- Can we spot other differences between a `char` and `String`?

```
char shirt3Size = 'L';  
String shirt6Size = "XXXL";
```

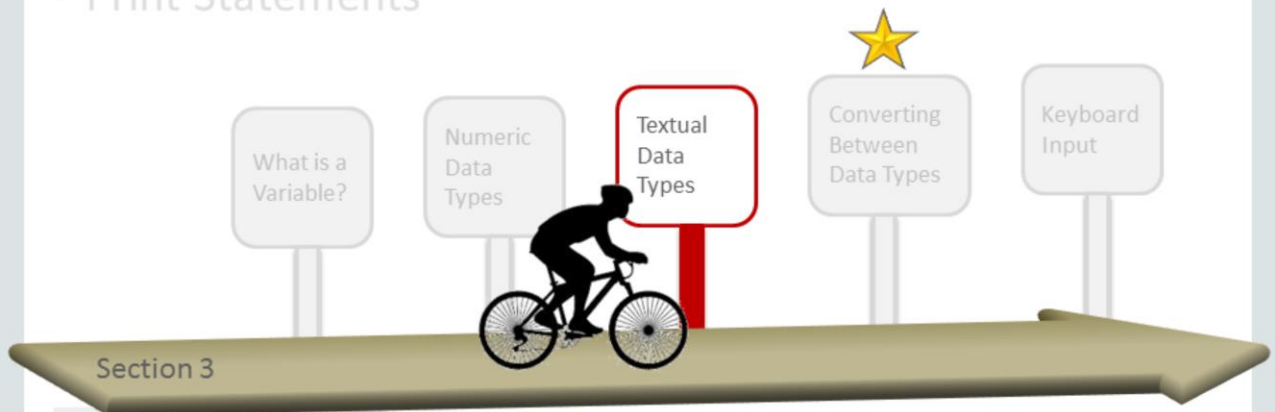
1. `char` turns blue.
 - `char` is a keyword for a primitive data type.
 - Keywords turn blue in NetBeans.
2. `String` is capitalized.
 - Strings are an object, not a primitive.
 - Object types are capitalized by convention.

Strings Are Objects

- Java comes with a String class which details.
 - String properties
 - String behaviors
- Strings are special objects.
 - Strings are handled a little differently than most objects.
- More on these points in future sections:
 - Objects may have primitives as properties.
 - Objects may have objects as properties, such as Strings.
 - Objects are stored differently from primitives in memory.

Topics

- Characters and Strings
- String Concatenation
- Mixing Strings and Numbers
- Print Statements



String Declaration and Initialization

Declare and assign `String` values like they're any other primitive.

```
//One variable declared and initialized
int intVar = 300;
String stringVar = "Three Hundred";

//Many variables declared
int x, y, z;
String xString, yString, zString;

//A declared variable is initialized later
x = 1;
xString = "One";
```

String Variable vs. String Literal

```
String stringVariable = "This is a String literal.";
```

Variable *Literal*

- A String can be created by combining String literals:

```
String combinedLiterals = "I want to" + " buy a shirt.";
```

- A String can be created by combining variables:

```
String var1 = "This shirt has";  
String var2 = " too many buttons."  
String combinedVariables = var1 + var2;
```

String Concatenation

- Combining multiple Strings is called concatenation.
- Strings can be combined by using the + operator.
 - stringVariable1 + stringVariable2
 - stringVariable1 + "String literal"
 - stringVariable1 + "String literal" + stringVariable2

```
String greet1 = "Hello";  
String greet2 = "World";  
String message1 = greet1 + " " +greet2 + "!";  
String message2 = greet1 + " " +greet2 + " " +2016 + "!";
```

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.

You can concatenate any number of String variables and String literals to achieve your goal.

String Concatenation Output

- Concatenation example:

```
String greet1 = "Hello";  
String greet2 = "World";  
String message1 = greet1 + " " +greet2 + "!";  
String message2 = greet1 + " " +greet2 + " " +2016 + "!";
```

- You can concatenate Strings within a print statement:

```
System.out.println(message2) ;  
System.out.println(greet1 + " " +greet2 + " " +2016 + "!") ;
```

Output:

Hello World 2016!

Hello World 2016!



JFo 3-3
Textual Data

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

17

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

- In the first example, the message2 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 is 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 1 Scenario

Think back to the Duke's Choice clothing catalog:

- The scenario included a ShoppingCart class.
- A few ShoppingCart properties and behaviors are loosely examined in this exercise.
- ShoppingCart properties: *Represented as Strings in this exercise*
 - Who owns it
 - The items it contains
 - A message/description of the cart
- ShoppingCart behaviors:
 - Prints its message



The Duke's Choice catalog scenario was discussed in Section 2, Lesson 3.



Exercise 1, Part 1

- Import and edit the ShoppingCart01 project.
- Declare and initialize the String variable `custName`.
- Declare and initialize the String variable `itemDesc`.
- Declare a String variable `message`.



Exercise 1, Part 2

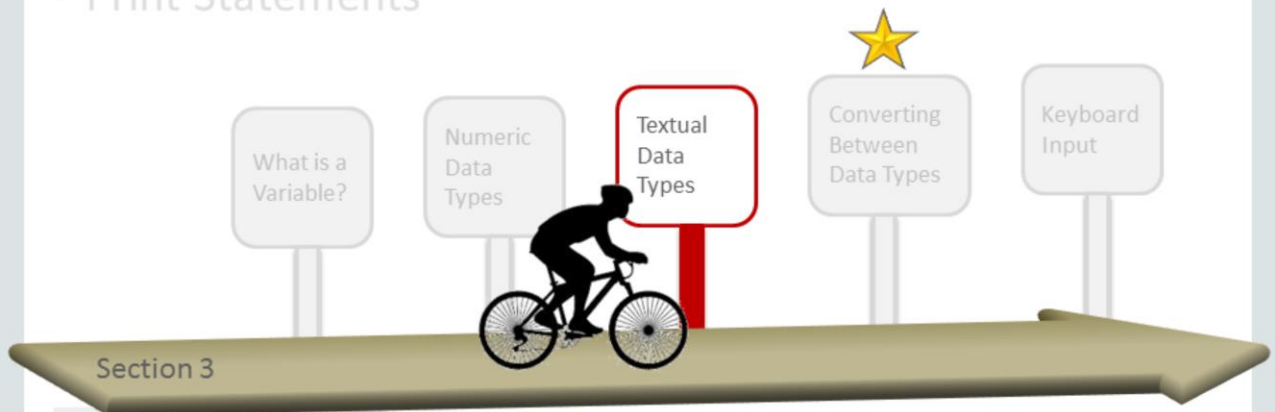
- Assign `message` a concatenated value that includes `custName`, `itemDesc`, and a String literal, which results in a complete sentence:
 - (example: “Alex wants to purchase a Shirt”)
- Print the message.

Your program should produce similar output:

```
Alex wants to purchase a Shirt
```

Topics

- Characters and Strings
- String Concatenation
- Mixing Strings and Numbers
- Print Statements



Mixing Strings and Numbers

- Strings may contain numbers:

```
String totalPrice = "Total: $" +3;  
System.out.println(totalPrice);      //Total: $3
```

- But be careful when trying to do math:

```
String totalPrice = "Total: $" +3 +2 +1;  
System.out.println(totalPrice);      //Total: $321
```

- Use parentheses for numbers:

```
String totalPrice = "Total: $" +(3 +2 +1);  
System.out.println(totalPrice);      //Total: $6
```

Exercise 2 Scenario

- Question: As customers fill their cart, how much will they pay?
- We need to represent the cart's items with a little more detail to answer this.



Exercise 2 Scenario

- A ShoppingCart may need to know the following **properties**:
 - Item price
 - Sales tax rate
 - Item quantity
 - Calculated total price of all items in the cart
- A ShoppingCart may need the following **behaviors**:
 - Print a message with its total





Exercise 2, Part 1

- Import and edit the ShoppingCart02 project.
- Declare and initialize numeric fields:
 - `price (double)`
 - `tax (double)`
 - `quantity (int)`
- Declare a double `totalPrice`:
 - Assign a value, calculated from `price`, `tax`, and `quantity`.



Exercise 2, Part 2

- Change message to include quantity:
 - (example: “Alex wants to purchase 2 Shirts.”)
- Print another message showing the total cost.

Your program should produce a similar output:

```
Alex wants to purchase 2 Shirts  
Total cost with tax is: $25.78
```

Exercise Notes

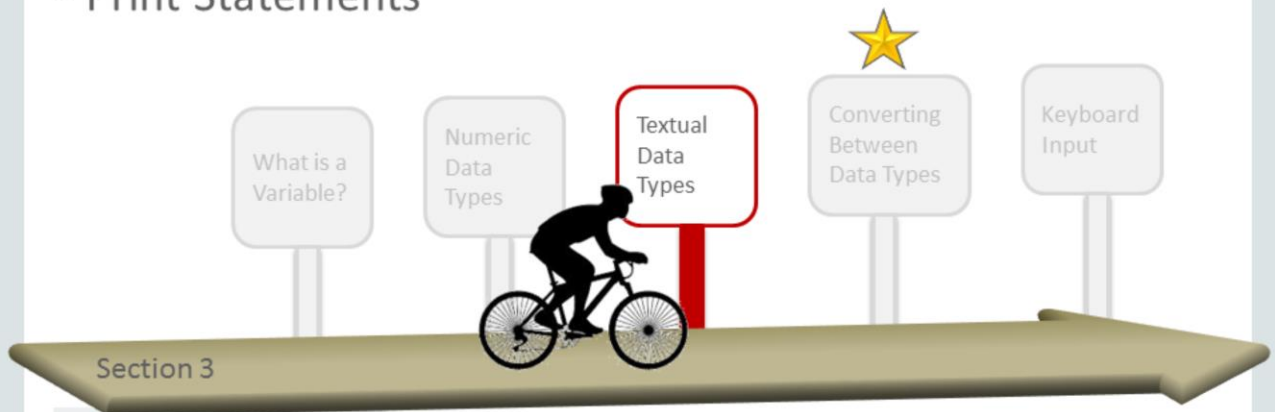
- It isn't best practice to represent properties and behaviors of objects entirely within the main method.
- We break this rule in this section so we can focus on manipulating data.
- We'll try to do a better job following the rules in the next section.

Aah! Why don't you follow the rules!?



Topics

- Characters and Strings
- String Concatenation
- Mixing Strings and Numbers
- Print Statements



- Remember when we printed the cat?
- The double backslash didn't actually print:
 - Only a single backslash printed.
 - Why?

ORACLE ACADEMY

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

29

Escape Sequence

- A character preceded by a backslash is called an **escape sequence** and has special meaning to the compiler.
- The table in the next slide shows Java escape sequences.

Escape Sequence

Escape Sequence	Description
<code>\t</code>	Insert a new tab.
<code>\b</code>	Insert a backspace.
<code>\n</code>	Insert a new line.
<code>\r</code>	Insert a carriage return.
<code>\f</code>	Insert a formfeed.
<code>\'</code>	Insert a single quote character.
<code>\"</code>	Insert a double quote character.
<code>\\</code>	Insert a backslash character.

Escape Sequence: Example

If you want to put quotation marks within quotation marks, you must use the escape sequence, `\`, on the interior quotation marks.

```
The cat said "Meow!" to me.
```

```
System.out.println("The cat said \"Meow!\" to me.");
```


Print Statements

- Writing text on a new line might not print to a new line:


```
System.out.println("This is the first line."  
    + "This is NOT the second line.");
```

Output:

This is the first line.This is NOT the second line.

- Escape sequences can help format your output:

```
System.out.println("This is the first line. \n"  
    + "This is the second line.");
```



Output:

This is the first line.

This is the second line.

More Print Statements

- There are two important methods for printing:

```
System.out.println("Print Line method");  
System.out.print("Print method");
```

- `println` works as if you're automatically putting `\n` at the end of the statement.
- The following two statements produce equivalent results:

```
System.out.println("Printing ");  
System.out.print("Printing \n");
```

`println()` vs. `print()`

- `println()` automatically creates a line:

```
System.out.println("This is the first line.");  
System.out.println("This is the second line.");
```

Output:

This is the first line.
This is the second line.

- `print()` won't automatically create a line:

```
System.out.print("This is the first line.");  
System.out.print("This is NOT the second line.");
```

Output:

This is the first line.This is NOT the second line.

NetBeans Shortcut

Print Method	How Often Will I Use this?
<code>System.out.println()</code>	Often
<code>System.out.print()</code>	Not so often

- `System.out.println()` is used very often.
- But requires a lot of typing to set up.
- Netbeans offers a shortcut:

1. Type `sout`.

```
sout
```

2. Press Tab.

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

Printing Lots of Text, Option 1

- Depending on what you're trying to print, you may find it beneficial to either:
 - Break a single print statement over many lines in NetBeans:

```
System.out.println("This is the first line."
    + "This is the still the first line."
    + "It's just that the line is very long "
    + "and I can't see it all in NetBeans."
    + "\n" + "This is the second line."
    + "\n" + "This is the third line.");
```

– OR...

Printing Lots of Text, Option 2

– Use many print statements:

```
System.out.println("This is the first line.");  
System.out.println("This is the second line.");  
System.out.println("This is the third line.");  
System.out.println("This is the fourth line.");
```

Summary

In this lesson, you should have learned how to:

- Use the `char` data type
- Use Strings
- Concatenate Strings
- Understand escape sequences
- Understand print statements better



