ORACLE® ACADEMY

Java Foundations

3-3
Textual Data

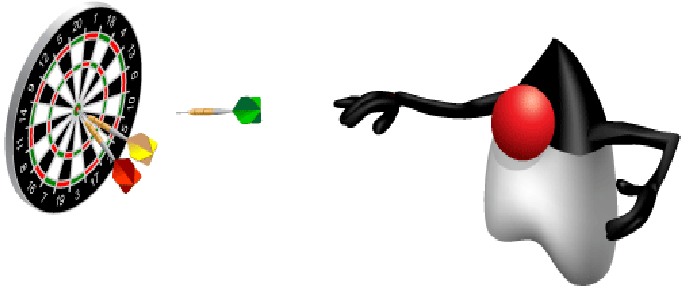
ORACLE ACADEMY

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

Objectives

This lesson covers the following objectives:

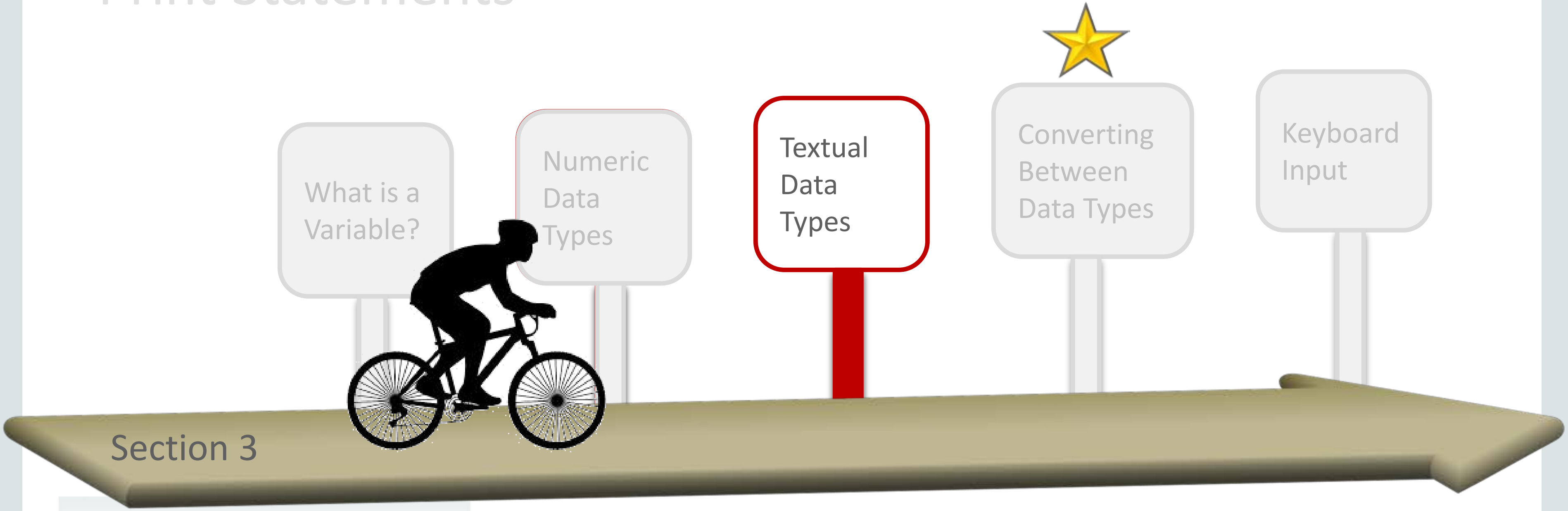
- Use the `char` data type
- Use Strings
- Concatenate Strings
- Understand escape sequences
- Understand print statements better



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

Topics

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



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

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.

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

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);
```

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

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);
```

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

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";
```

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

Characters vs. Strings

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

✓

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

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

Primitives

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

Where are Strings?

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

Let’s Investigate

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

```java
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.

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

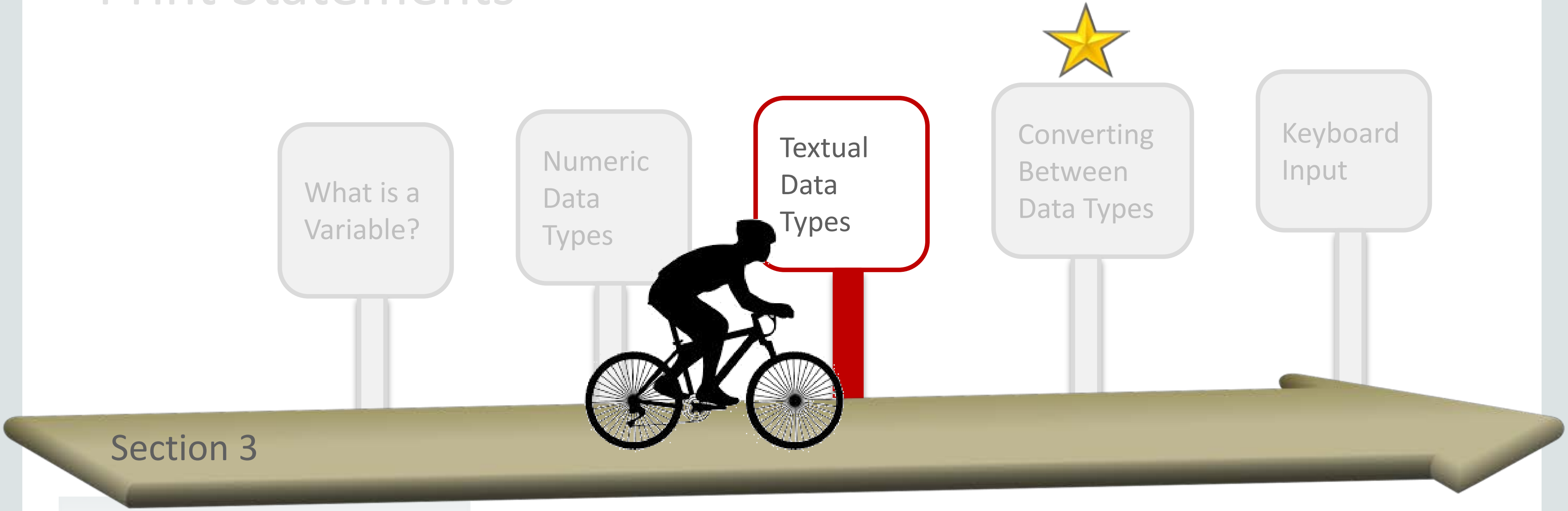
Strings Are Objects

- 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.

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

Topics

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



What is a Variable?

Numeric Data Types

Textual Data Types

Converting Between Data Types

Keyboard Input

Section 3

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

String Declaration and Initialization

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

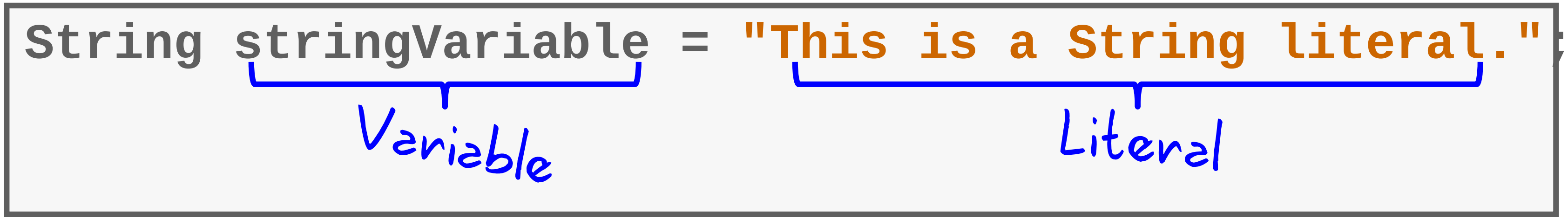
```
//One variable declared and initialized
int intVar = 300;
String stringVar = "Three Hundred";

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

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

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

String Variable vs. String Literal



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

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

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 +"!";
```

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

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!
```

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

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

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

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`.

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

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.

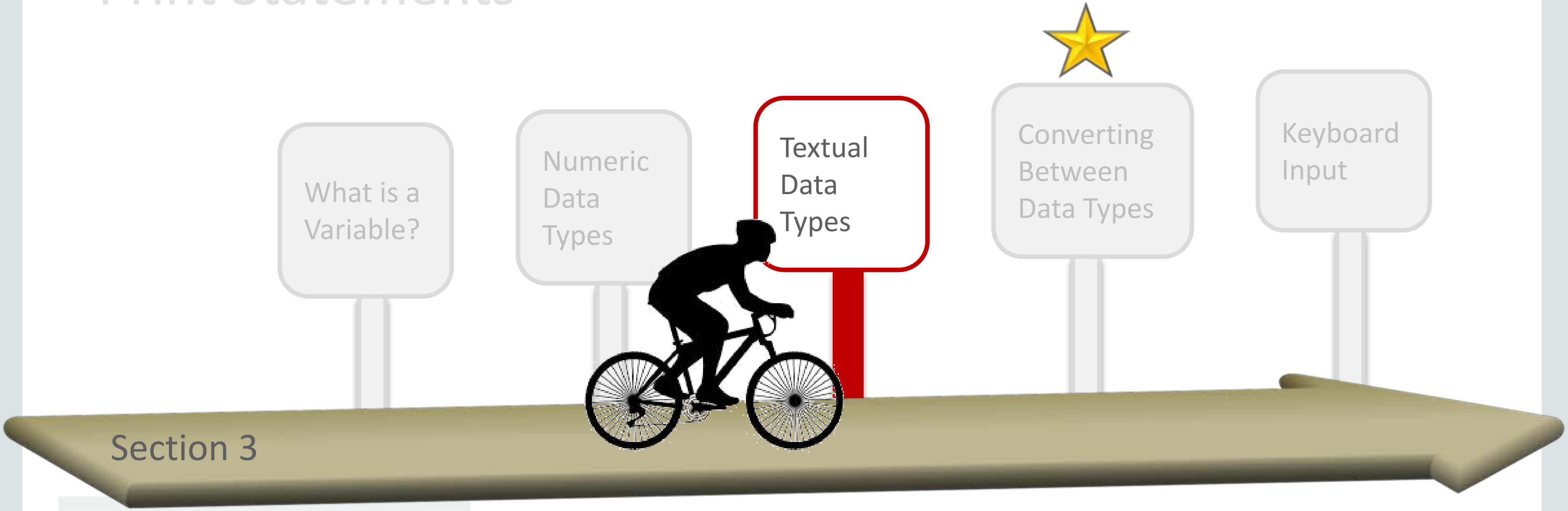
Your program should produce similar output:

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

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

Topics

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



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

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
```

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

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.

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

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

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

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.`

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

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
```

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

Exercise Notes

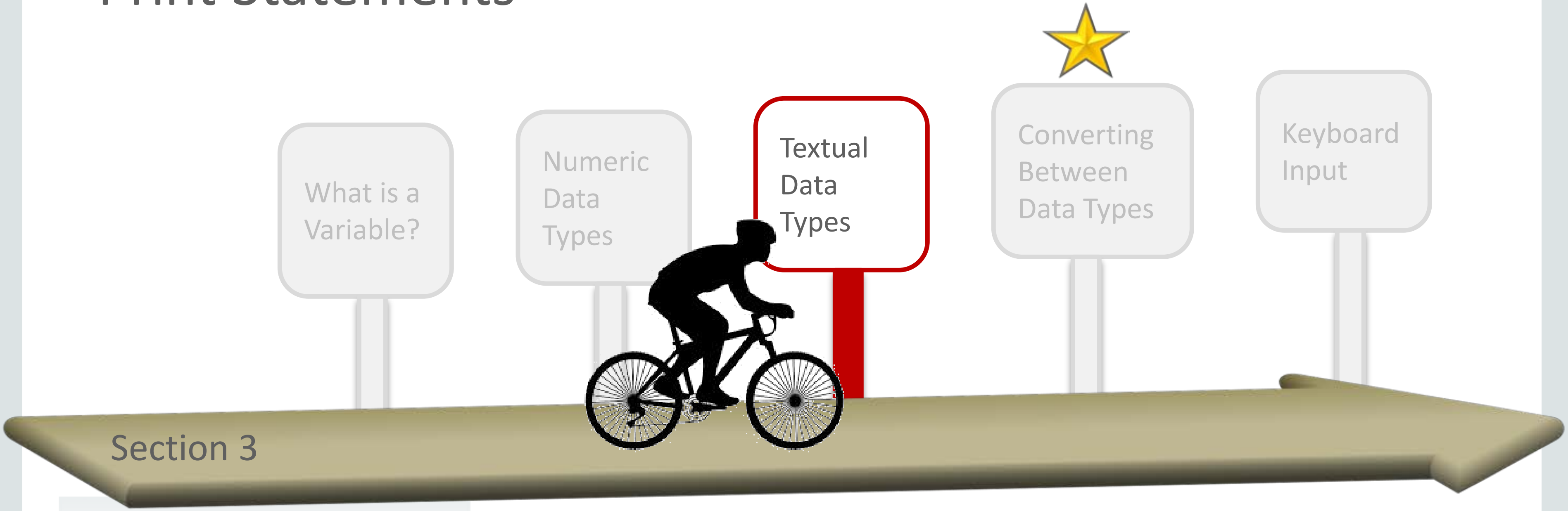
- It isn’t best practice to represent properties and behaviors of objects entirely within the main method.
- We break this rule in this section so we can focus on manipulating data.
- We’ll try to do a better job following the rules in the next section.

Aah! Why don’t you follow the rules!?

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

Topics

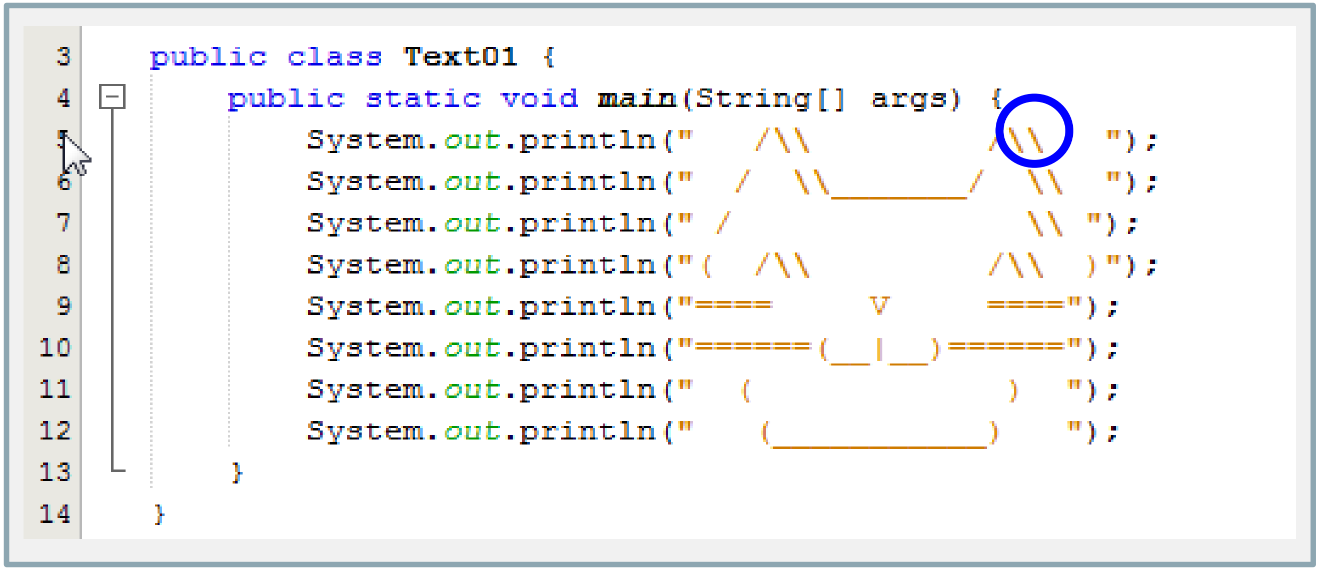
- Characters and Strings
- String Concatenation
- Mixing Strings and Numbers
- Print Statements



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

Special Characters in Strings

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



```
public class Text01 {
    public static void main(String[] args) {
        System.out.println("   /\\         /\\   ");
        System.out.println("  /  \\_______/  \\  ");
        System.out.println(" /               \\ ");
        System.out.println("(  /\\         /\\  )");
        System.out.println("====     V     ====");
        System.out.println("======(__|__)======");
        System.out.println("  (             )  ");
        System.out.println("   (___________)   ");
    }
}
```

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

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.

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

Escape Sequence

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

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

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.");
```

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

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.
```

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

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");
```

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

`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.
```

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

NetBeans Shortcut

Print Method	How Often Will I Use this?
`System.out.println()`	Often
`System.out.print()`	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("");
```

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

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…

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

Printing Lots of Text, Option 2

- Use many print statements:

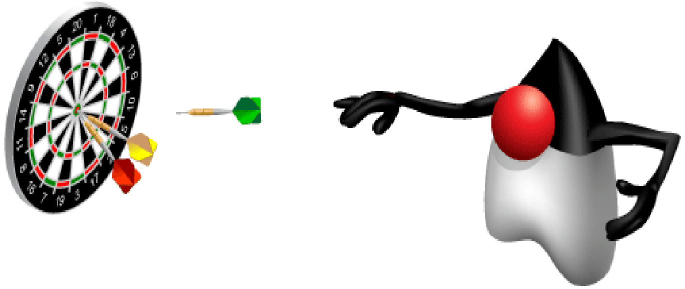
```
System.out.println("This is the first line.");
System.out.println("This is the second line.");
System.out.println("This is the third line.");
System.out.println("This is the fourth line.");
```

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

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



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

ORACLE® ACADEMY