

7

Manipulating and Formatting the Data in Your Program

ORACLE

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

1

Objectives

After completing this lesson, you should be able to:

- Describe the `String` class and use some of the methods of the `String` class
- Use the JDK documentation to search for and learn how to use a class
- Describe the `StringBuilder` class
- Explain what a constant is and how to use it
- Explain the difference between promoting and casting of variables



ORACLE

7 - 2

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

2

Topics

- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

String Class

```
String hisName = "Fred Smith"; — Standard syntax
```

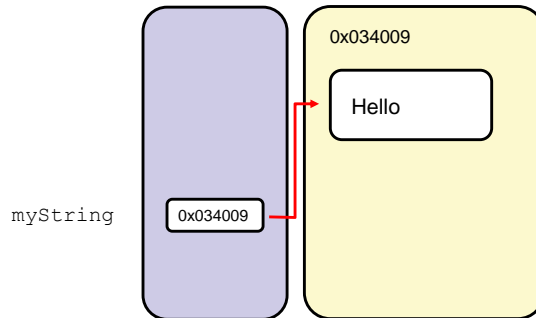
The `new` keyword can be used,
but it is *not* best practice:

```
String herName = new String("Anne Smith");
```

- A `String` object is immutable; its value cannot be changed.
- A `String` object can be used with the string concatenation operator symbol (+) for concatenation.

Concatenating Strings

```
String myString = "Hello";
```



ORACLE

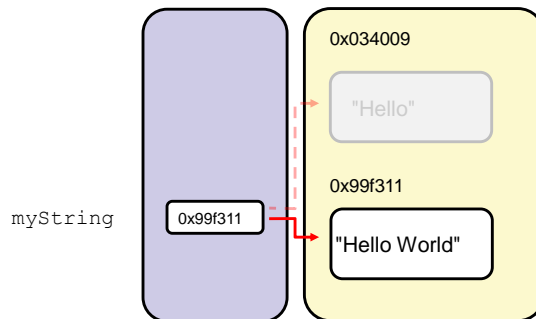
7 - 5

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

5

Concatenating Strings

```
String myString = "Hello";  
myString = myString.concat(" World");
```



ORACLE

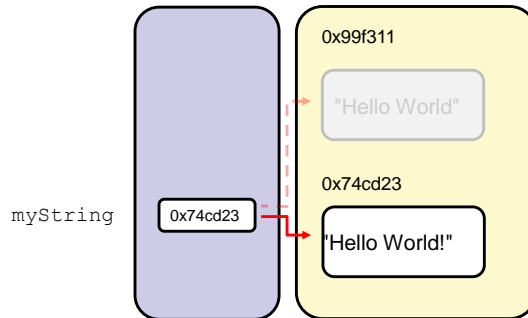
7 - 6

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

6

Concatenating Strings

```
String myString = "Hello";  
myString = myString.concat(" World");  
myString = myString + "!"
```



ORACLE

7 - 7

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

7

String Method Calls with Primitive Return Values

A method call can return a single value of any type.

- An example of a method of primitive type `int`:

```
String hello = "Hello World";  
int stringLength = hello.length();
```

ORACLE

7 - 8

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

8

String Method Calls with Object Return Values

Method calls returning objects:

```
String greet = " HOW ".trim();  
String lc = greet + "DY".toLowerCase();
```

Or

```
String lc = (greet + "DY").toLowerCase();
```

Topics

- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

Java API Documentation

Consists of a set of webpages;

- Lists all the classes in the API
 - Descriptions of what the class does
 - List of constructors, methods, and fields for the class
- Highly hyperlinked to show the interconnections between classes and to facilitate lookup
- Available on the Oracle website at:
<http://download.oracle.com/javase/8/docs/api/index.html>

ORACLE

7 - 11

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

11

Java Platform SE 8 Documentation

Select All Classes
or a particular
package.

The classes for the
selected package(s)
are listed here.

Details about the
class selected

The screenshot shows the Java Platform SE 8 Documentation website. On the left, there is a sidebar with a search bar and a list of packages and classes. The 'String' class is selected. The main content area displays the details for the 'String' class, including its inheritance, implemented interfaces, and source code. The 'String' class is a final class that extends 'Object' and implements 'Serializable', 'ComparableString', and 'CharSequence'. It is used to represent character strings in Java programs.

ORACLE

7 - 12

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

12

Java Platform SE 8: Method Summary

```
public int charAt(String str)
```

The return type of
the method

The name of
the method

The type of the parameter that must be
passed into the method

| Method Summary | |
|-------------------|---|
| Methods | |
| Modifier and Type | Method and Description |
| char | charAt (int index) Returns the char value at the specified index. |
| int | codePointAt (int index) Returns the character (Unicode code point) at the specified index. |
| int | codePointBefore (int index) Returns the character (Unicode code point) before the specified index. |
| int | codePointCount (int beginIndex, int endIndex) Returns the number of Unicode code points in the specified text range of this String. |
| int | compareTo (String anotherString) Compares two strings lexicographically. |
| int | compareToIgnoreCase (String str) Compares two strings lexicographically, ignoring case differences. |
| String | concat (String str) Concatenates the specified string to the end of this string. |

7 - 13

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

ORACLE

13

Java Platform SE 8: Method Detail

Click here to get the detailed
description of the method.

| | |
|-----|---|
| int | indexOf (String str) Returns the index within this string of the first occurrence of the specified substring. |
| int | indexOf (String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |

Detailed description for the
indexOf() method

Further details about
parameters and return value are
shown in the method list.

| indexOf |
|--|
| <pre>public int indexOf(String str)</pre> <p>Returns the index within this string of the first occurrence of the specified substring. The returned index is the smallest value <i>k</i> for which:</p> <pre>this.startsWith(str, k)</pre> <p>If no such value of <i>k</i> exists, then -1 is returned.</p> <p>Parameters:</p> <p>str - the substring to search for.</p> <p>Returns:</p> <p>the index of the first occurrence of the specified substring, or -1 if there is no such occurrence.</p> |

7 - 14

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

ORACLE

14

indexOf Method Example

```
1 String phoneNum = "404-543-2345";
2 int idx1 = phoneNum.indexOf('-');
3 System.out.println("index of first dash: " + idx1);
4
5
6 int idx2 = phoneNum.indexOf('-', idx1+1);
7 System.out.println("second dash idx: " + idx2);
```

The 1-arg version

The 2-arg version

Topics

- Using the `String` class
- Using the Java API docs
- **Using the `StringBuilder` class**
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

StringBuilder Class

`StringBuilder` provides a mutable alternative to `String`.
`StringBuilder`:

- Is instantiated using the `new` keyword
- Has many methods for manipulating its value
- Provides better performance because it is mutable
- Can be created with an initial capacity

`String` is still needed because:

- It may be safer to use an immutable object
- A method in the API may require a string
- It has many more methods not available on `StringBuilder`

ORACLE

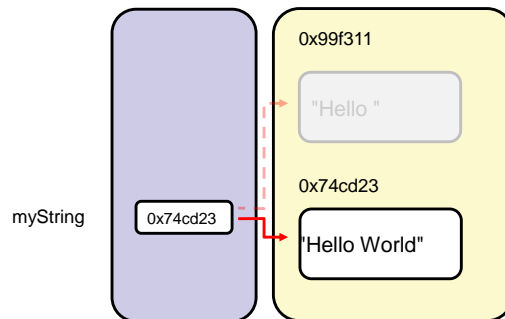
7 - 17

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

17

StringBuilder Advantages over String for Concatenation (or Appending)

```
String myString = "Hello";  
myString = myString + " World";
```



ORACLE

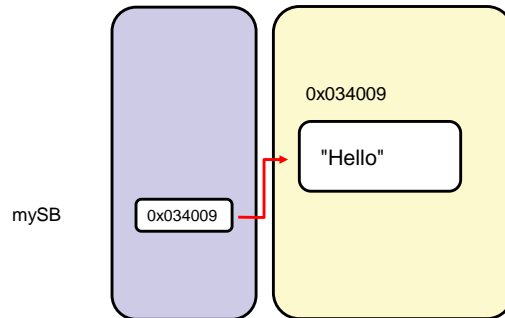
7 - 18

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

18

StringBuilder: Declare and Instantiate

```
StringBuilder mySB = new StringBuilder("Hello");
```



ORACLE

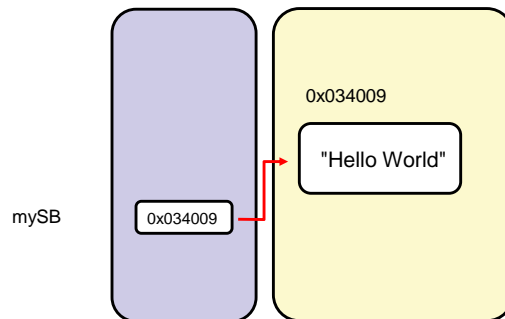
7 - 19

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

19

StringBuilder Append

```
StringBuilder mySB = new StringBuilder("Hello");  
mySB.append(" World");
```



ORACLE

7 - 20

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

20

Quiz

Which of the following statements are true? (Choose all that apply.)

- a. The dot (.) operator creates a new object instance.
- b. The `String` class provides you with the ability to store a sequence of characters.
- c. The Java API specification contains documentation for all of the classes in a Java technology product.
- d. `String` objects cannot be modified.

ORACLE

7 - 21

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

21

Exercise 7-1: Use `indexOf` and `substring` Methods

In this exercise, you use `indexOf` and `substring` methods to get just the customer's first name and display it.



ORACLE

7 - 22

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

22

Exercise 7-2: Instantiate the `StringBuilder` object

In this exercise, you instantiate a `StringBuilder` object, initializing it to `firstName` using the `StringBuilder` constructor.



ORACLE

7 - 23

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

23

Topics

- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

ORACLE

7 - 24

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

24

Primitive Data Types

- Integral types (`byte`, `short`, `int`, and `long`)
- Floating point types (`float` and `double`)
- Textual type (`char`)
- Logical type (`boolean`)

ORACLE

7 - 25

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

25

Some New Integral Primitive Types

| Type | Length | Range |
|--------------------|---------|---|
| <code>byte</code> | 8 bits | -2^7 to $2^7 - 1$ (-128 to 127, or 256 possible values) |
| <code>short</code> | 16 bits | -2^{15} to $2^{15} - 1$ (-32,768 to 32,767, or 65,535 possible values) |
| <code>int</code> | 32 bits | -2^{31} to $2^{31} - 1$ (-2,147,483,648 to 2,147,483,647, or 4,294,967,296 possible values) |
| <code>long</code> | 64 bits | -2^{63} to $2^{63} - 1$ (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807, or 18,446,744,073,709,551,616 possible values) |

ORACLE

7 - 26

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

26

Floating Point Primitive Types

| Type | Float Length |
|---|--------------|
| <code>float</code> | 32 bits |
| <code>double</code> (default type for floating point literals) | 64 bits |

Example:

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

ORACLE

7 - 28

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

28

Textual Primitive Type

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

```
- public char colorCode = 'U';
```

Single quotes must be used with char literal values.

ORACLE

7 - 29

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

29

Java Language Trivia: Unicode

- Unicode is a standard character encoding system.
 - It uses a 16-bit character set.
 - It can store all the necessary characters from most languages.
 - Programs can be written so they display the correct language for most countries.

| Character | UTF-16 | UTF-8 | UCS-2 |
|-----------|-----------|-------------|-------|
| A | 0041 | 41 | 0041 |
| c | 0063 | 63 | 0063 |
| Ö | 00F6 | C3 B6 | 00F6 |
| 世 | 4E9C | E4 BA 9C | 4E9C |
| € | D834 DD1E | F0 9D 84 9E | N/A |

ORACLE

7 - 30

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

30

Constants

- Variable (can change):
 - `double salesTax = 6.25;`
- Constant (cannot change):
 - `final int NUMBER_OF_MONTHS = 12;`

The `final` keyword causes a variable to be read only.

ORACLE

7 - 31

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

31

Quiz

The variable declaration `public int myInteger=10;` adheres to the variable declaration and initialization syntax.

- a. True
- b. False

Topics

- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

Modulus Operator

| Purpose | Operator | Example | Comments |
|-----------|-----------------------------|---|--|
| Remainder | $\%$ modulus | <pre>num1 = 31; num2 = 6; mod = num1 % num2; mod is 1</pre> | <p>Remainder finds the remainder of the first number divided by the second number.</p> <p>5 R 1</p> <p>6 $\overline{)31}$ 30 ----- 1</p> <p>Remainder always gives an answer with the same sign as the first operand.</p> |

ORACLE

7 - 34

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

34

Combining Operators to Make Assignments

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

ORACLE

7 - 35

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

35

More on Increment and Decrement Operators

| Operator | Purpose | Example |
|----------|-------------------------------|---|
| ++ | Preincrement (++variable) | int id = 6; int newId = ++id; id is 7, newId is 7 |
| | Postincrement (variable++) | int id = 6; int newId = id++; id is 7, newId is 6 |
| -- | Predecrement (--variable) | (same principle applies) |
| | Postdecrement (variable--) | |

Increment and Decrement Operators (++ and --)

Examples:

```
1  int count=15;
2  int a, b, c, d;
3  a = count++;
4  b = count;
5  c = ++count;
6  d = count;
7  System.out.println(a + ", " + b + ", " + c + ", " + d);
```

Output:

15, 16, 17, 17

Topics

- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

ORACLE

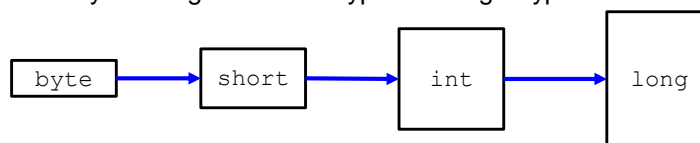
7 - 38

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

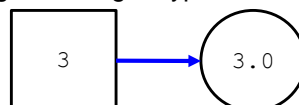
38

Promotion

- Automatic promotions:
 - If you assign a smaller type to a larger type



- If you assign an integral type to a floating point type



- Examples of automatic promotions:
 - `long intToLong = 6;`
 - `double intToDouble = 3;`

ORACLE

7 - 39

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

39

Caution with Promotion

Equation:

$$55555 * 66666 = 3703629630$$

Example of potential issue:

```
1 int num1 = 55555;
2 int num2 = 66666;
3 long num3;
4 num3 = num1 * num2;           //num3 is -591337666
```

Example of potential solution:

```
1 int num1 = 55555;
2 long num2 = 66666; ————— Changed from int to long
3 long num3;
4 num3 = num1 * num2;           //num3 is 3703629630
```

ORACLE

7 - 40

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

40

Caution with Promotion

Equation:

$$7 / 2 = 3.5$$

Example of potential issue:

```
1 int num1 = 7;
2 int num2 = 2;
3 double num3;
4 num3 = num1 / num2;           //num3 is 3.0
```

Example of potential solution:

```
1 int num1 = 7;
2 double num2 = 2; ————— Changed from int to double
3 double num3;
4 num3 = num1 / num2;           //num3 is 3.5
```

ORACLE

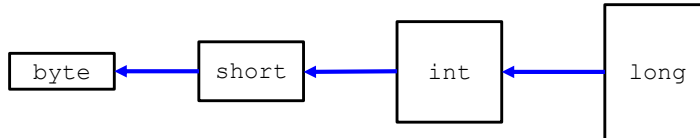
7 - 41

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

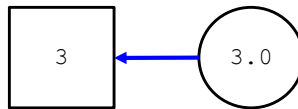
41

Type Casting

- When to cast:
 - If you assign a larger type to a smaller type



- If you assign a floating point type to an integral type



- Examples of casting:
 - `int longToInt = (int)20L;`
 - `short doubleToShort = (short)3.0;`

ORACLE

7 - 42

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

42

Caution with Type Casting

Example of potential issue:

```
1 int myInt;  
2 long myLong = 123987654321L;  
3 myInt = (int) (myLong); // Number is "chopped"  
4                        // myInt is -566397263
```

Safer example of casting:

```
1 int myInt;  
2 long myLong = 99L;  
3 myInt = (int) (myLong); // No data loss, only zeroes.  
4                        // myInt is 99
```

ORACLE

7 - 43

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

43

Caution with Type Casting

- Be aware of the possibility of lost precision.

Example of potential issue:

```
1 int myInt;  
2 double myPercent = 51.9;  
3 myInt = (int) (myPercent); // Number is "chopped"  
4                               // myInt is 51
```

ORACLE

7 - 44

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

44

Using Promotion and Casting

Example of potential issue:

```
1 int num1 = 53; // 32 bits of memory to hold the value  
2 int num2 = 47; // 32 bits of memory to hold the value  
3 byte num3;      // 8 bits of memory reserved  
4 num3 = (num1 + num2); // causes compiler error
```

Solution using a larger type for num3:

```
1 int num1 = 53;  
2 int num2 = 47;  
3 int num3; ——— Changed from byte to int  
4 num3 = (num1 + num2);
```

Solution using casting:

```
1 int num1 = 53; // 32 bits of memory to hold the value  
2 int num2 = 47; // 32 bits of memory to hold the value  
3 byte num3;      // 8 bits of memory reserved  
4 num3 = (byte) (num1 + num2); // no data loss
```

ORACLE

7 - 45

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

45

Compiler Assumptions for Integral and Floating Point Data Types

- Most operations result in an `int` or `long`:
 - `byte`, `char`, and `short` values are automatically promoted to `int` prior to an operation.
 - If an expression contains a `long`, the entire expression is promoted to `long`.
- If an expression contains a floating point, the entire expression is promoted to a floating point.
- All literal floating point values are viewed as `double`.

ORACLE

7 - 46

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

46

Automatic Promotion

Example of potential problem:

```
short a, b, c;  
a = 1 ;  
b = 2 ;  
c = a + b ; //compiler error
```

} a and b are automatically promoted to integers.

Example of potential solutions:

- Declare `c` as an `int` type in the original declaration:
`int c;`
- Type cast the `(a+b)` result in the assignment line:
`c = (short) (a+b);`

ORACLE

7 - 47

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

47

Using a long

```
1 public class Person {
2
3     public int ageYears = 32;
4
5     public void calculateAge() {
6
7         int ageDays = ageYears * 365;
7         long ageSeconds = ageYears * 365 * 24L * 60 * 60;
8
9         System.out.println("You are " + ageDays + " days old.");
10        System.out.println("You are " + ageSeconds + " seconds old.");
11
12    } // end of calculateAge method
13 } // end of class
```

Using the L to indicate a long will result in the compiler recognizing the total result as a long.

ORACLE

7 - 48

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

48

Using Floating Points

Example of potential problem:

Expressions are automatically promoted to floating points.

```
int num1 = 1 + 2 + 3 + 4.0;
int num2 = (1 + 2 + 3 + 4) * 1.0;
```

//compiler error
//compiler error

Example of potential solutions:

- Declare num1 and num2 as double types:

```
double num1 = 1 + 2 + 3 + 4.0; //10.0
double num2 = (1 + 2 + 3 + 4) * 1.0; //10.0
```

- Type cast num1 and num2 as int types in the assignment line:

```
int num1 = (int) (1 + 2 + 3 + 4.0); //10
int num2 = (int) ((1 + 2 + 3 + 4) * 1.0); //10
```

ORACLE

7 - 49

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

49

Floating Point Data Types and Assignment

- Example of potential problem:

```
float float1 = 27.9; //compiler error
```

- Example of potential solutions:

- The `F` notifies the compiler that 27.9 is a float value:

```
float float1 = 27.9F;
```

- 27.9 is cast to a float type:

```
float float1 = (float) 27.9;
```

Quiz

Which statements are true?

- a. There are eight primitive types built in to the Java programming language.
- b. `byte`, `short`, `char`, and `long` are the four integral primitive data types in the Java programming language.
- c. A boolean type variable holds `true`, `false`, and `nil`.
- d. `short Long = 10;` is a valid statement that adheres to the variable declaration and initialization syntax.

Exercise 7-3: Declare a Long, Float, and Char

In this exercise, you experiment with the data types introduced in this lesson. You:

- Declare and initialize variables
- Cast one numeric type to another



ORACLE

7 - 52

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

52

Summary

In this lesson, you should have learned how to:

- Describe the `String` class and use some of the methods of the `String` class
- Use the JDK documentation to search for and learn how to use a class
- Use the `StringBuilder` class to manipulate string data
- Create a constant by using the `final` keyword in the variable declaration
- Describe how the Java compiler can use promotion or casting to interpret expressions and avoid a compiler error



ORACLE

7 - 53

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

53

Play Time!

Play **Basic Puzzle 8** before the lesson titled “Creating and Using Methods.”

Consider the following:

What happens when you rotate the blue wheel?

How else can you affect the rotation of bumpers?



ORACLE

7 - 54

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

54

Practice 7-1 Overview: Manipulating Text

This practice covers the following topics:

- Searching for a particular player and printing out the last name
- Reversing the player name so that the family name is printed first



ORACLE

7 - 55

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

55