7 Manipulating and Formatting the Data in Your Program

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

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

3

`String` Class

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

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

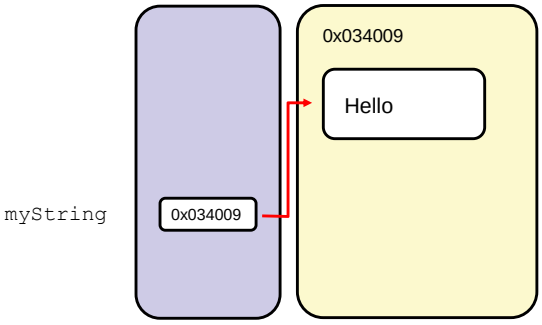
```
String herName = new String("Anne Smith");
```

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

4

Concatenating Strings

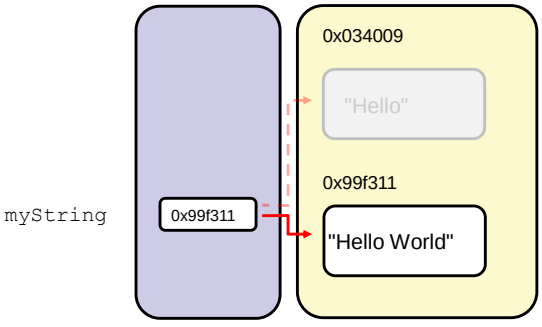
```
String myString = "Hello";
```



5

Concatenating Strings

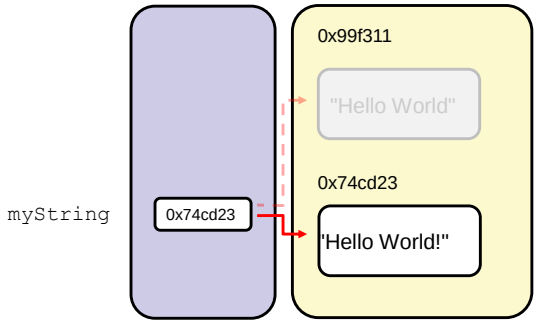
```
String myString = "Hello";
myString = myString.concat(" World");
```



6

Concatenating Strings

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



`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();
```

`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

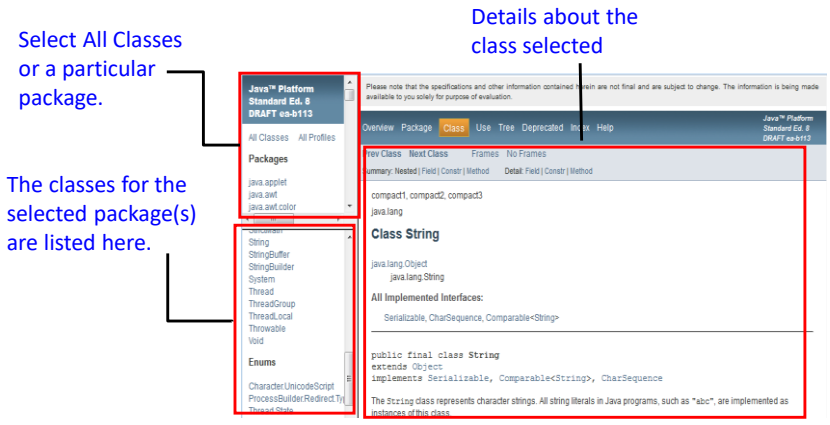
Java API Documentation

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

11

Java Platform SE 8 Documentation



12

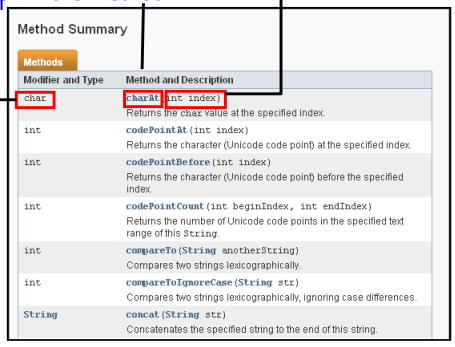
Java Platform SE 8: Method Summary

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

The return type of the method

The name of the method

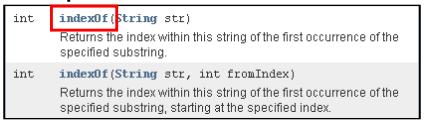
The type of the parameter that must be passed into the method



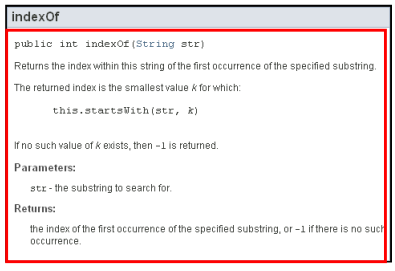
13

Java Platform SE 8: Method Detail

Click here to get the detailed description of the method.



Detailed description for the `indexOf()` method



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

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

15

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

16

`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

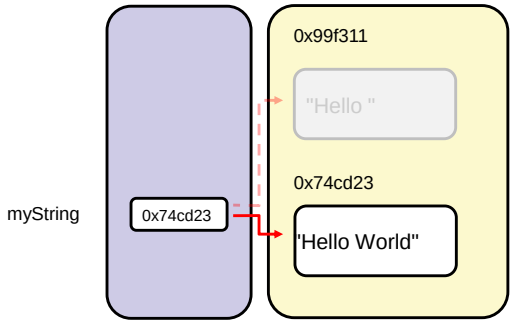
`String` is still needed because:

- 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`

17

`StringBuilder` Advantages over String for Concatenation (or Appending)

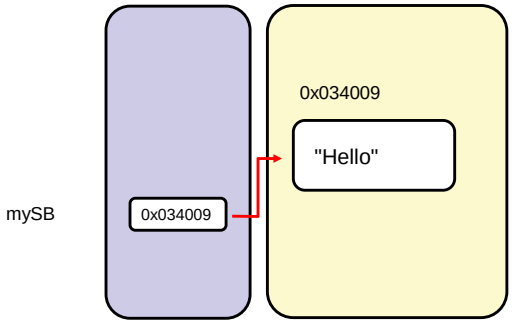
```
String myString = "Hello";
myString = myString + " World";
```



18

StringBuilder: Declare and Instantiate

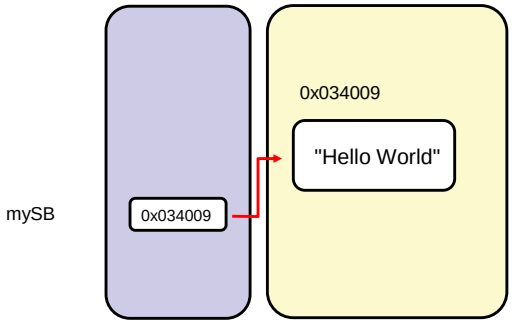
```
StringBuilder mySB = new StringBuilder("Hello");
```



19

StringBuilder Append

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



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.

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.

22

Exercise 7-2: Instantiate the `StringBuilder` object

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

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

24

Primitive Data Types

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

Some New Integral Primitive Types

Type	Length	Range
`byte`	8 bits	-2^7 to $2^7 - 1$ (–128 to 127, or 256 possible values)
`short`	16 bits	-2^{15} to $2^{15} - 1$ (–32,768 to 32,767, or 65,535 possible values)
`int`	32 bits	-2^{31} to $2^{31} - 1$ (–2,147,483,648 to 2,147,483,647, or 4,294,967,296 possible values)
`long`	64 bits	-2^{63} to $2^{63} - 1$ (–9,223,372,036854,775,808 to 9,223,372,036854,775,807, or 18,446,744,073,709,551,616 possible values)

Floating Point Primitive Types

Type	Float Length
`float`	32 bits
`double` (default type for floating point literals)	64 bits

Example:

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


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.

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
亜	4E9C	E4 BA 9C	4E9C
𝄞	D834 DD1E	F0 9D 84 9E	N/A

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.

31

Quiz

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

a. True
b. False

32

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

33

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.

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

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--*)	

36

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
```

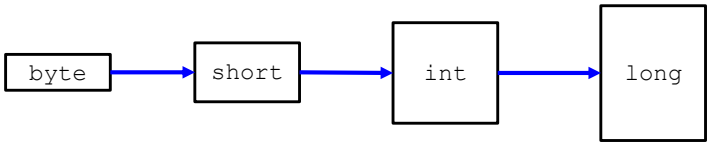

37

Topics

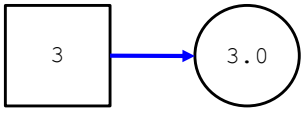
- Using the `String` class
- Using the Java API docs
- Using the `StringBuilder` class
- Doing more with primitive data types
- Using the remaining numeric operators
- Promoting and casting variables

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;`

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
```


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
```

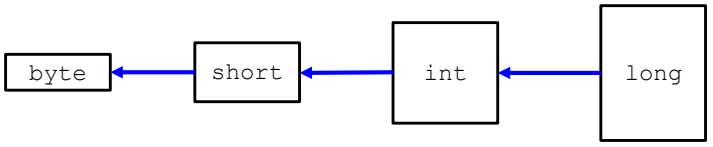
Example of potential solution:

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

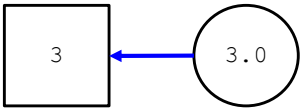

41

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;`

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
```


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
```


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
```


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`.

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);
```

Using a `long`

```
1 public class Person {
2
3   public int ageYears = 32;
4
4   public void calculateAge() {
5
6     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.

48

Using Floating Points

Example of potential problem:

Expressions are automatically promoted to floating points.

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


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

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

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?

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

55