String Data Type and Operations

Hui Chen

Department of Computer & Information Science

Brooklyn College

Objectives

- To introduce objects and instance methods (**§**4.4).

- To represent strings using the **String** objects (§4.4).

- To return the string length using the **length()** method (§4.4.1).

- To return a character in the string using the **charAt(i)** method (§4.4.2).

- To use the **+** operator to concatenate strings (§4.4.3).

- To read strings from the console (§4.4.4).

- To read a character from the console (§4.4.5).

- To compare strings using the **equals** method and the **compareTo** methods (§4.4.6).

- To obtain substrings (§4.4.7).

- To find a character or a substring in a string using the **indexOf** method (§4.4.8).

Outline

- Discussed
  - The Math class and its methods and constants
  - The char data type and The Character class
- The String data type
  - String class, objects, and invoking the instance methods
  - Simple (important) methods and operations of String objects
    - Getting the length, getting a character
    - Concatenating strings
    - Reading strings, characters from the console
    - Comparing strings using the equals and compareTo methods
    - Obtaining substrings
    - Finding a character or a substring

The String Data Type

- The char type only represents one character. To represent a string of characters, use the data type called String.

- Example

  - String message = "Welcome to Java";

The String Class

- String is actually a predefined class in the Java library just like the System class and Scanner class.

- The String type is not a primitive type. It is known as a *reference type*.

- Any Java class can be used as a reference type for a variable.

- For the time being, you just need to know
    - how to declare a String variable,
    - how to assign a string to the variable,
    - how to concatenate strings, and
    - how to perform simple operations for strings.

Simple Methods for **String** Objects

Method	Description
length()	Returns the number of characters in this string.
charAt(index)	Returns the character at the specified index from this string.
concat(s1)	Returns a new string that concatenates this string with string s1.
toUpperCase()	Returns a new string with all letters in uppercase.
toLowerCase()	Returns a new string with all letters in lowercase.
trim()	Returns a new string with whitespace characters trimmed on both sides.

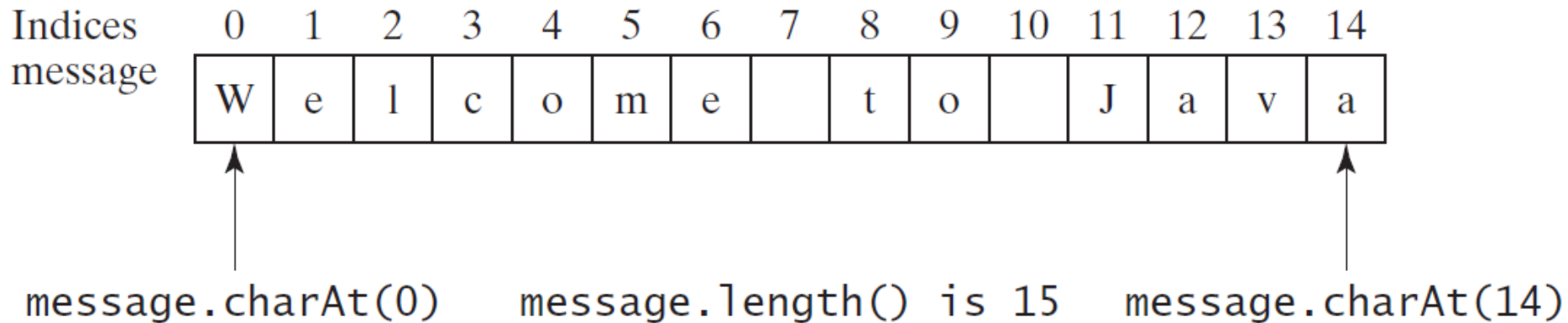# Remark: Simple Methods for **String** Objects

- Strings are **objects** in Java.

- The methods in the preceding table can only be invoked from a specific string object/instance.

- For this reason, these methods are called *instance methods*.

- A non-instance method is called a *static method*.

- A static method can be invoked without using an object.

- All the methods defined in the **Math** class are static methods.

- They are not tied to a specific object instance. The syntax to invoke an instance method is

  - **referenceVariable.methodName(arguments)**

Getting String Length

String message = **"Welcome to Java"**;

System.out.println(**"The length of "** + message + **" is "** + message.length());

Getting Characters from a String

Indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
message	W	e	l	c	o	m	e		t	o		J	a	v	a

```
message.charAt(0)     message.length() is 15     message.charAt(14)
```

String message = **"Welcome to Java"**;

System.out.println(**"The first character in message is "**

  + message.charAt(0));

Converting Strings

- Examples

  "Welcome".toLowerCase() returns a new string, welcome.

  "Welcome".toUpperCase() returns a new string, WELCOME.

  " Welcome ".trim() returns a new string, Welcome.

String Concatenation

- Examples

    String s3 = s1.concat(s2); or String s3 = s1 + s2;

    // Three strings are concatenated

    String message = "Welcome " + "to " + "Java";

    // String Chapter is concatenated with number 2

    String s = "Chapter" + 2; // s becomes Chapter2

    // String Supplement is concatenated with character B

    String s1 = "Supplement" + 'B'; // s1 becomes SupplementB

Reading Strings from the Console

- Examples

    Scanner input = new Scanner(System.in);

    System.out.print("Enter three words separated by spaces: ");

    String s1 = input.next();

    String s2 = input.next();

    String s3 = input.next();

    System.out.println("s1 is " + s1);

    System.out.println("s2 is " + s2);

    System.out.println("s3 is " + s3);

Reading a Character from the Console

- Examples

    Scanner input = new Scanner(System.in);

    System.out.print("Enter a character: ");

    String s = input.nextLine();

    char ch = s.charAt(0);

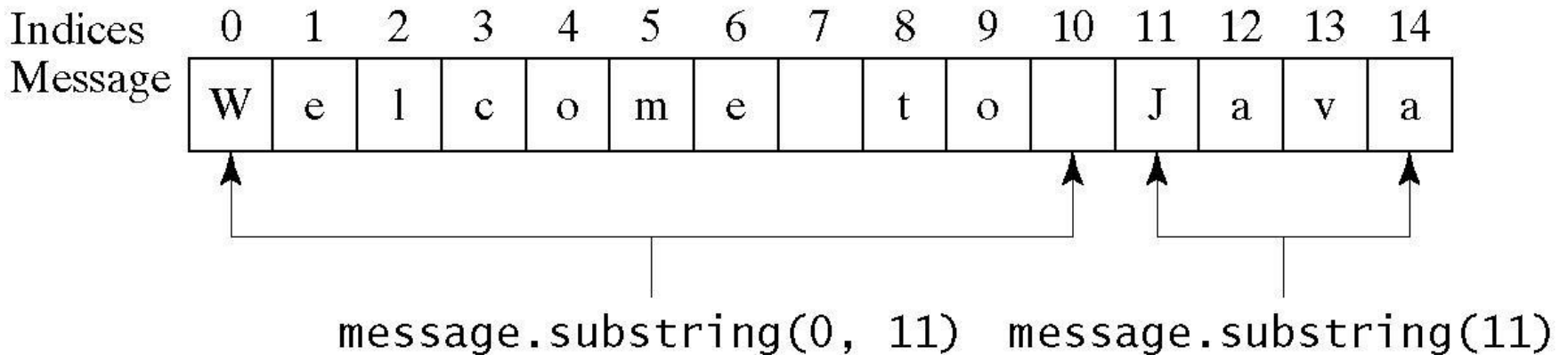   System.out.println("The character entered is " + ch);

Compare Strings

Method	Description
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	Returns an integer greater than `0`, equal to `0`, or less than `0` to indicate whether this string is greater than, equal to, or less than `s1`.
`compareToIgnoreCase(s1)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.

Obtaining Substrings

Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`, as shown in Figure 9.6. Note that the character at `endIndex` is not part of the substring.


```
Indices   0   1   2   3   4   5   6   7   8   9   10  11  12  13  14
Message   W   e   l   c   o   m   e       t   o       J   a   v   a
```

message.substring(0, 11)   message.substring(11)

# Finding a Character or a Substring

Method	Description
indexOf(ch)	Returns the index of the first occurrence of ch in the string. Returns −1 if not matched.
indexOf(ch, fromIndex)	Returns the index of the first occurrence of ch after fromIndex in the string. Returns −1 if not matched.
indexOf(s)	Returns the index of the first occurrence of string s in this string. Returns −1 if not matched.
indexOf(s, fromIndex)	Returns the index of the first occurrence of string s in this string after fromIndex. Returns −1 if not matched.
lastIndexOf(ch)	Returns the index of the last occurrence of ch in the string. Returns −1 if not matched.
lastIndexOf(ch, fromIndex)	Returns the index of the last occurrence of ch before fromIndex in this string. Returns −1 if not matched.
lastIndexOf(s)	Returns the index of the last occurrence of string s. Returns −1 if not matched.
lastIndexOf(s, fromIndex)	Returns the index of the last occurrence of string s before fromIndex. Returns −1 if not matched.

Examples of Finding Characters and Substrings

int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
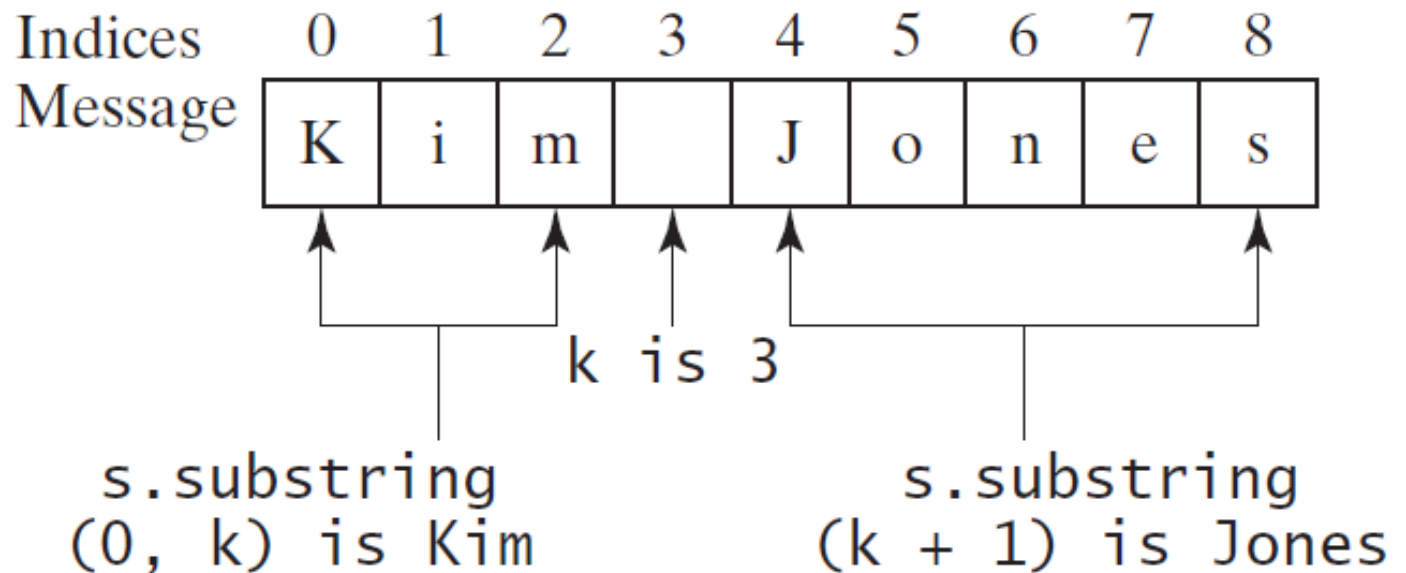
Conversion between Strings and Numbers

- Examples

  int intValue = Integer.parseInt(intString);

  double doubleValue = Double.parseDouble(doubleString);

  String s = number + "";

Questions?