String

GLTi Institutional Presentation

HSBC Technology and Services

HSBC
The world's local bank

Strings

- To process strings using the String class, the StringBuffer class, and the StringTokenizer class.
- To use the String class to process fixed strings.
- To use static methods in the Character class.
- To use the StringBuffer class to process flexible strings.
- To use the StringTokenizer class to extract tokens from a string.
- To use the command-line arguments.

The `String` Class

- Constructing a String:
  - `String message = "Welcome to Java!"`
  - `String message = new String("Welcome to Java!");`
  - `String s = new String();`
- Obtaining String length and Retrieving Individual Characters in a string String
- String Concatenation (concat)
- Substrings (substring(index), substring(start, end))
- Comparisons (equals, compareTo)
- String Conversions
- Finding a Character or a Substring in a String
- Conversions between Strings and Arrays
- Converting Characters and Numeric Values to Strings

}

String

+String()
+String(value: String)
+String(value: char[])
+charAt(index: int): char
+compareTo(anotherString: String): int
+compareToIgnoreCase(anotherString: String): int
+concat(anotherString: String): String
+endsWithSuffixe(suffix: String): boolean
+equals(anotherString: String): boolean
+equalsIgnoreCase(anotherString: String): boolean
+indexOf(ch: int): int
+indexOf(ch: int, fromIndex: int): int
+indexOf(str: String): int
+indexOf(str: String, fromIndex: int): int
+intern(): String
+regionMatches(toffset: int, other: String, offset: int, len: int): boolean
+length(): int
+replace(oldChar: char, newChar: char): String
+startsWith(prefix: String): boolean
+subString(beginIndex: int): String
+subString(beginIndex: int, endIndex: int): String
+toCharArray(): char[]
+toLowerCase(): String
+toString(): String
+toUpperCase(): String
+trim(): String
+copyValueOf(data: char[]): String
+valueOf(c: char): String
+valueOf(data: char[]): String
+valueOf(d: double): String
+valueOf(f: float): String
+valueOf(i: int): String
+valueOf(l: long): String

Constructing Strings

Strings newString = new String(stringLiteral);

String message = new String("Welcome to Java!");

Since strings are used frequently, Java provides a shorthand notation for creating a string:

String message = "Welcome to Java!";

Strings Are Immutable

NOTE: A String object is immutable, whose contents cannot be changed. To improve efficiency and save memory, Java Virtual Machine stores two String objects into the same object, if the two String objects are created with the same string literal using the shorthand notation. Therefore, the shorthand notation is preferred to create strings.

Strings Are Immutable, cont.

NOTE: A string that is created using the shorthand notation is known as a *canonical string*. You can use the String's intern method to return a canonical string, which is the same string that is created using the shorthand notation.

Examples

```
String s = "Welcome to Java!";
String s1 = new String("Welcome to Java!");
String s2 = s1.intern();
System.out.println("s1 == s is " + (s1 == s));
System.out.println("s2 == s is " + (s2 == s));
System.out.println("s1 == s2 is " + (s1 == s2));
```

display

s1 == s is false

s2 == s is true

s1 == s2 false

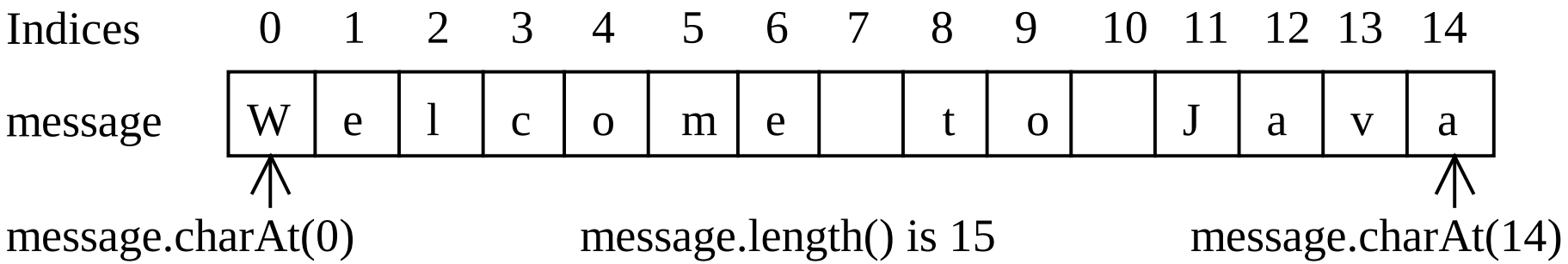
Finding String Length

Finding string length using the `length()` method:

```
message = "Welcome";
message.length() (returns 7)
```

Retrieving Individual Characters in a String

- Do not use `message[0]`
- Use `message.charAt(index)`
- Index starts from `0`



String Concatenation

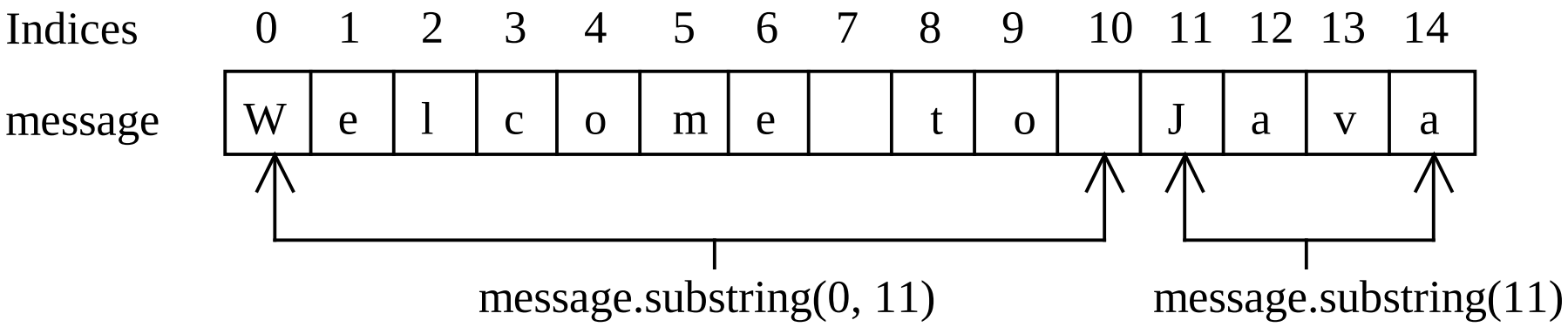
```
String s3 = s1.concat(s2);
```

String s3 = s1 + s2;

Extracting Substrings

`String` is an immutable class; its values cannot be changed individually.

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0, 11) + "HTML";
```



String Comparisons

- equals

```
String s1 = "Welcome";
String s2 = "welcome";

if (s1.equals(s2)){
  // s1 and s2 have the same contents
}

if (s1 == s2) {
  // s1 and s2 have the same reference
}
```

String Comparisons, cont.

} compareTo(Object object)

```
String s1 = "Welcome";
String s2 = "welcome";

if (s1.compareTo(s2) > 0) {
  // s1 is greater than s2
}
else if (s1.compareTo(s2 == 0) {
  // s1 and s2 have the same reference
}
else
   // s1 is less than s2
```

String Conversions

The contents of a string cannot be changed once the string is created. But you can convert a string to a new string using the following methods:

- toLowerCase
- toUpperCase
- trim
- replace(oldChar, newChar)

Finding a Character or a Substring in a String

`"Welcome to Java!".indexOf('W'))` returns 0.

`"Welcome to Java!".indexOf('x'))` returns -1.

`"Welcome to Java!".indexOf('o', 5))` returns 9.

`"Welcome to Java!".indexOf("come"))` returns 3.

`"Welcome to Java!".indexOf("Java", 5))` returns 11.

`"Welcome to Java!".indexOf("java", 5))` returns -1.

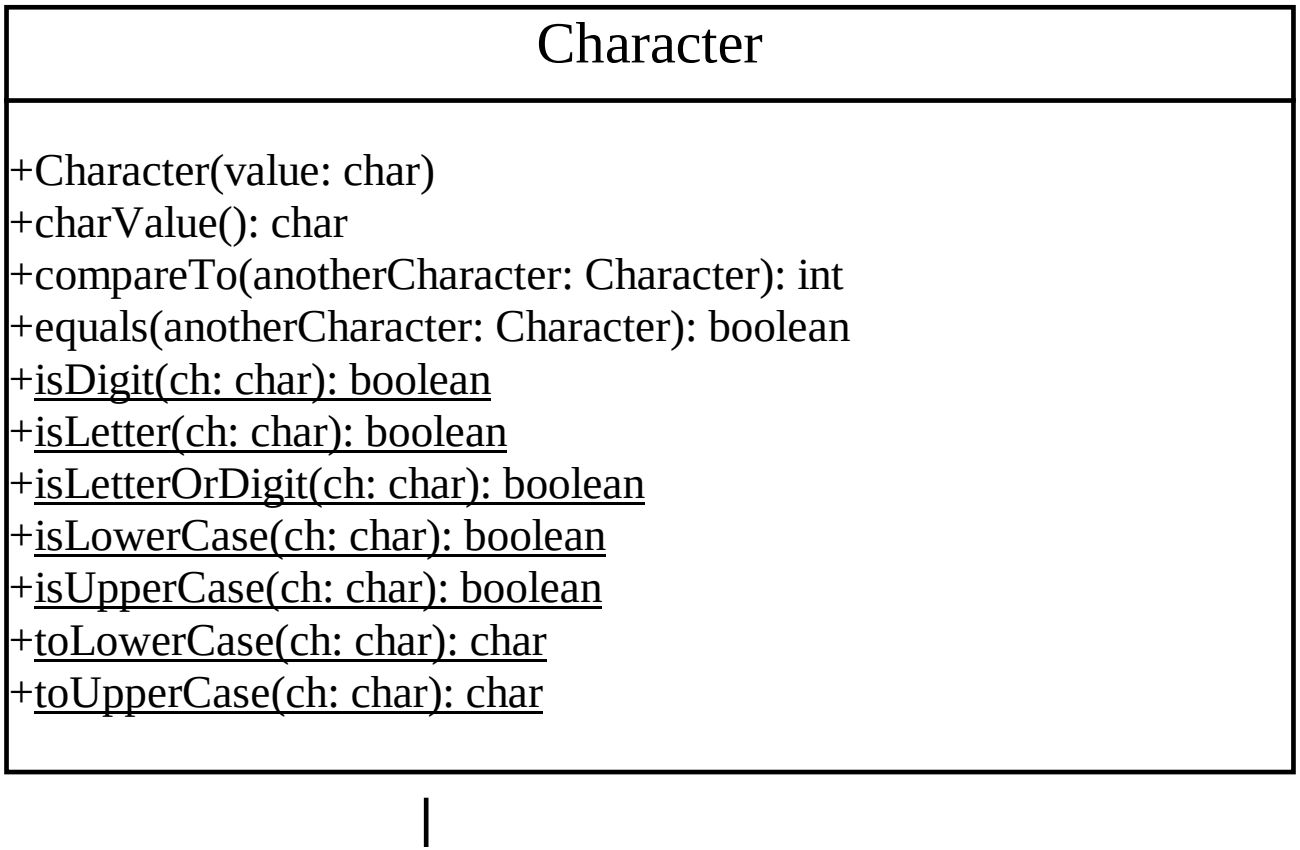
Convert Character and Numbers to Strings

The String class provides several static valueOf methods for converting a character, an array of characters, and numeric values to strings. These methods have the same name valueOf with different argument types char, char[], double, long, int, and float. For example, to convert a double value to a string, use String.valueOf(5.44). The return value is string consists of characters ‘5’, ‘.’, ‘4’, and ‘4’.

Example 7.1
Finding Palindromes

- Objective: Checking whether a string is a palindrome: a string that reads the same forward and backward.

The `Character` Class



Examples

charObject.compareTo(new Character('a')) returns 1
charObject.compareTo(new Character('b')) returns 0
charObject.compareTo(new Character('c')) returns -1
charObject.compareTo(new Character('d') returns –2
charObject.equals(new Character('b')) returns true
charObject.equals(new Character('d')) returns false

Example 7.2
Counting Each Letter in a String

This example gives a program that counts the number of occurrence of each letter in a string. Assume the letters are not case-sensitive.

The `StringBuffer` Class

The `StringBuffer` class is an alternative to the `String` class. In general, a string buffer can be used wherever a string is used.

`StringBuffer` is more flexible than `String`. You can add, insert, or append new contents into a string buffer. However, the value of a string is fixed once the string is created.

}

StringBuffer
+append(data: char[]): StringBuffer +append(data: char[], offset: int, len: int): StringBuffer +append(v: *aPrimitiveType*): StringBuffer +append(str: String): StringBuffer +capacity(): int +charAt(index: int): char +delete(startIndex: int, endIndex: int): StringBuffer +deleteCharAt(int index): StringBuffer +insert(index: int, data: char[], offset: int, len: int): StringBuffer +insert(offset: int, data: char[]): StringBuffer +insert(offset: int, b: *aPrimitiveType*): StringBuffer +insert(offset: int, str: String): StringBuffer +length(): int +replace(int startIndex, int endIndex, String str): StringBuffer +reverse(): StringBuffer +setCharAt(index: int, ch: char): void +setLength(newLength: int): void +substring(start: int): StringBuffer +substring(start: int, end: int): StringBuffer

`StringBuffer` Constructors

- `public StringBuffer()`
  No characters, initial capacity 16 characters.

- `public StringBuffer(int length)`
  No characters, initial capacity specified by the `length` argument.

- `public StringBuffer(String str)`
  Represents the same sequence of characters as the `string` argument. Initial capacity 16 plus the length of the `string` argument.

Appending New Contents into a String Buffer

```
StringBuffer strBuf = new StringBuffer();
strBuf.append("Welcome");
strBuf.append(' ');
strBuf.append("to");
strBuf.append(' ');
strBuf.append("Java");
```

Example 7.3
Checking Palindromes Ignoring Non-alphanumeric Characters

This example gives a program that counts the number of occurrence of each letter in a string. Assume the letters are not case-sensitive.

Example 7.4
Using StringBuffer for Output

This This example gives a program that prints the multiplication table created in Example 3.4, "Using Nested for Loops," from Chapter 3. Rather than print one number at a time, the program appends all the elements of the table into a string buffer. After the table is completely constructed in the string buffer, the program prints the entire string buffer on the console once.

TestMulTableUsingStringBuffer

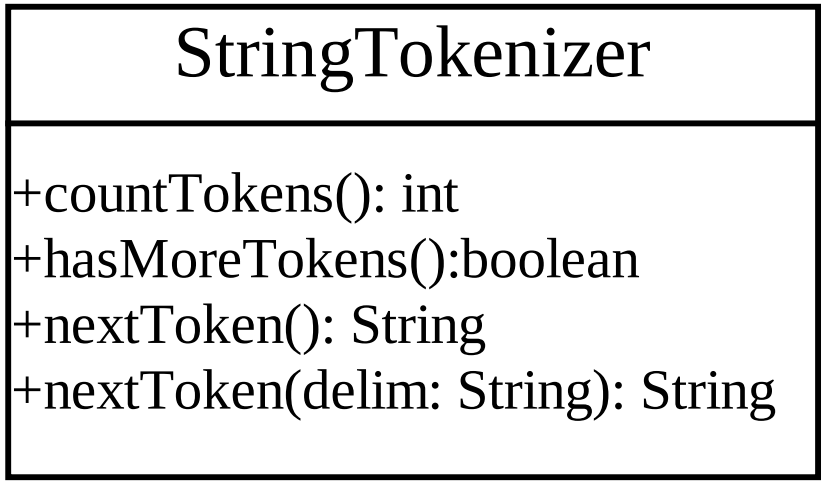
Run

The `StringTokenizer` Class Constructors

- `StringTokenizer(String s, String delim, boolean returnTokens)`
- `StringTokenizer(String s, String delim)`
- `StringTokenizer(String s)`

The `StringTokenizer` Class Methods

- `boolean hasMoreTokens()`
- `String nextToken()`
- `String nextToken(String delim)`



Example 7.5
Testing `StringTokenizer`

- Objective: Using a string tokenizer, retrieve words from a string and display them on the console.

TestStringTokenizer

Run

Command-Line Parameters

```
class TestMain {
  public static void main(String[] args) {
  ...
  }
}

java TestMain arg0 arg1 arg2 ... argn
```

Processing Command-Line Parameters

In the main method, get the arguments from `args[0], args[1], ..., args[n]`, which corresponds to `arg0, arg1, ..., argn` in the command line.

Example 7.6
Using Command-Line Parameters

- Objective: Write a program that will perform binary operations on integers. The program receives three parameters: an operator and two integers.

java Calculator + 2 3

java Calculator - 2 3

java Calculator / 2 3

java Calculator “*” 2 3

About HSBC Technology and Services

HSBC Technology and Services (HTS) is a pivotal part of the Group and seamlessly integrates technology platforms and operations with an aim to re-define customer experience and drive down unit cost of production. Its solutions connect people, devices and networks across the globe and combine domain expertise, process skills and technology to deliver unparalleled business value, thereby enabling HSBC to stay ahead of competition by addressing market changes quickly and developing profitable customer relationships.

Presenter's Contact Details:
Name: Deepak Ratnani
Role: Team Leader
Direct: + 91-20 -66423608
Email: DeepakRatnani@hsbc.co.in