

20. Strings

Java

Fall 2009

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- String Class
- Character Class
- StringBuffer Class
- Command-Line Arguments
- References



Introduction



Introduction

- Strings are used often in programming.
- A string is a sequence of characters.
- In many languages, strings are treated as arrays of characters, but in Java a string is an object.
- Java provides the `String`, `StringBuilder`, and `StringBuffer` classes for storing and processing strings.

Introduction

- In most cases, you use the `String` class to create strings.
- The `String` class is efficient for storing and processing strings, but strings created with the `String` class cannot be modified.
 - `String` object is immutable
- The `StringBuilder` and `StringBuffer` classes enable you to create flexible strings that can be modified.



String Class



String Class

- The `java.lang.String` class models a sequence of characters as a string.
- The `String` class has eleven constructors and more than forty methods

Constructing a String

- To create a string from a string literal:

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

- Java treats a string literal as a `String` object. So the following statement is valid:

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

- To create a `String` object with no value:

```
String s = new String(); // same as String s = "";
```


Constructing a String

- A **String** variable holds a reference to a **String** object that stores a string value.
- These terms are different:
 - **String** variable
 - **String** object
 - string value
- For simplicity, the term string will often be used to refer to **String** variable, **String** object, and string value.

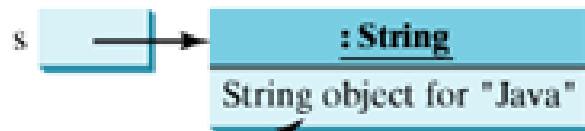
Strings

Strings Are Immutable

- A `String` object is immutable; its contents cannot be changed.
- The following code does not change the contents of the string.

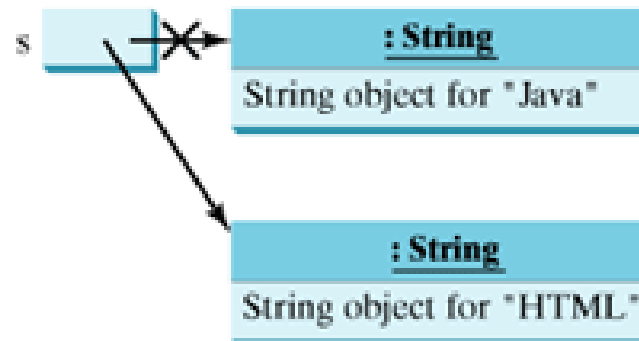
```
String s = "Java";  
s = "HTML";
```

After executing `String s = "Java";`



Contents cannot be changed

After executing `s = "HTML";`



This string object is now unreferenced

String Comparisons

- If you attempt to use the `==` operator, as follows:

```
if (string1 == string2)
```

```
    System.out.println("string1 and string2 are the same object");
```

```
else
```

```
    System.out.println("string1 and string2 are different objects");
```

- What does it is compare?
- The `==` operator only checks whether `string1` and `string2` refer to the same object
- It does not tell you whether `string1` and `string2` contain the same contents.

String Comparisons

- you should use the `equals` method for an equality comparison of the contents of objects.

```
if (string1.equals(string2))
    System.out.println("string1 and string2 have the same contents");
else
    System.out.println("string1 and string2 are not equal");
```

- The `equals` method returns `true` if two strings are equal, and `false` if they are not equal.

String Length

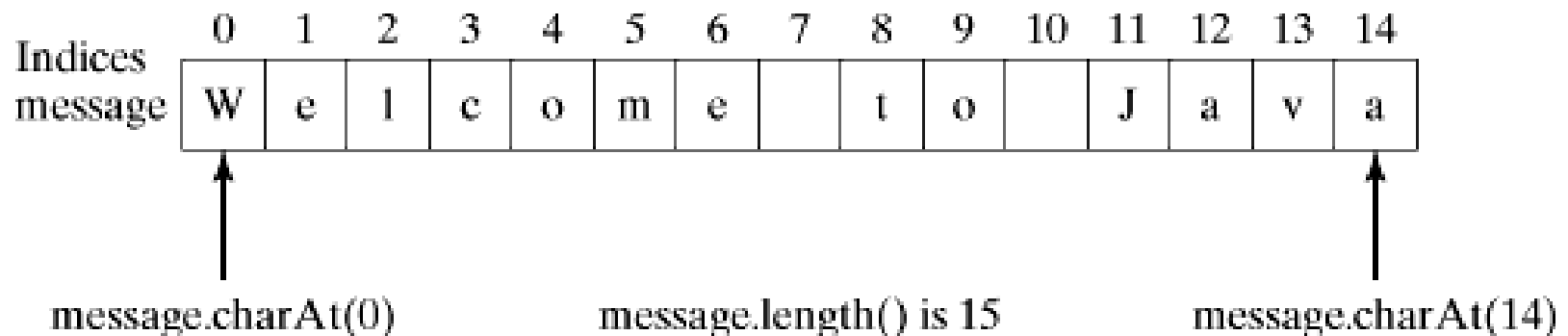
Finding string length using the `length()` method:

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

Strings

Retrieving Individual Characters in a String

- `message.charAt(index)` can be used to retrieve a specific character in a string `message`.
- The index is between 0 and `s.length() - 1`.



String Concatenation

- You can use the `concat` method to concatenate two strings.

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

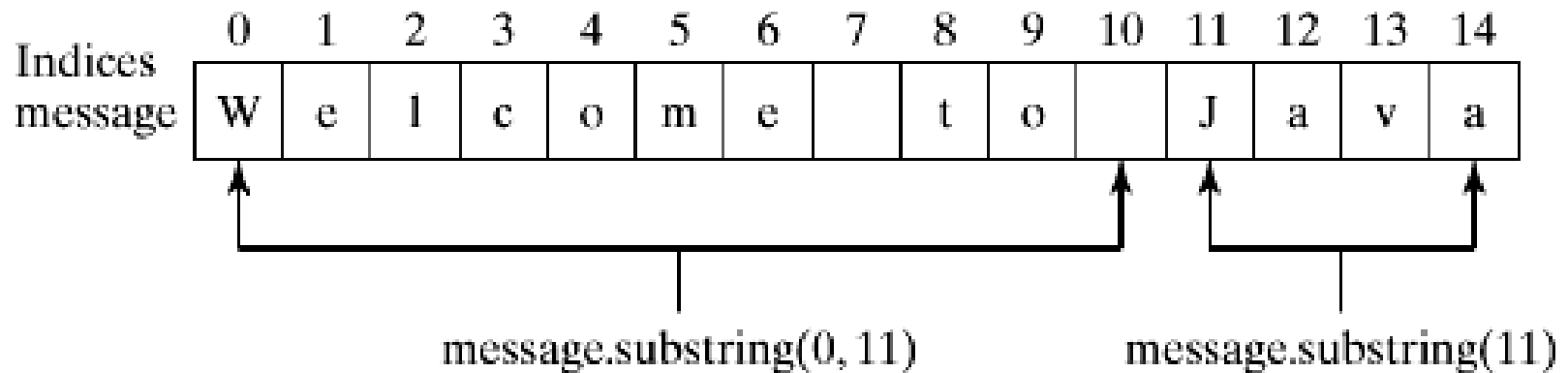
```
String s3 = s1 + s2;
```

Extracting Substrings

- You can obtain a substring from a string using the substring method in the `String` class.

```
String s1 = "Welcome to Java";
```

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



- The `s2` now becomes "Welcome to HTML".

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` and `toUpperCase` methods
 - return a new string by converting all the characters in the string to lowercase or uppercase.
- `trim` method
 - returns a new string by eliminating blank characters from both ends of the string.
- `replace(oldChars, newChars)` method
 - can be used to replace all occurrences of a character in the string with a new character.

Finding a Character or a Substring in a String

- You can use the `indexOf` and `lastIndexOf` methods to find a character or a substring in a string.

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

`"Welcome to Java".indexOf('o')` returns 4.

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

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

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

`"Welcome to Java".lastIndexOf('o')` returns 9.

`"Welcome to Java".lastIndexOf("come")` returns 3.

`"Welcome to Java".lastIndexOf("java")` returns -1.

Strings

Converting Characters and Numeric Values 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'.

Strings

Converting Strings to Characters and Numeric Values

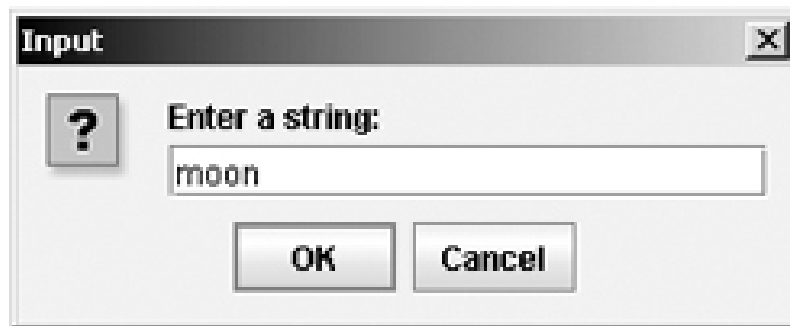
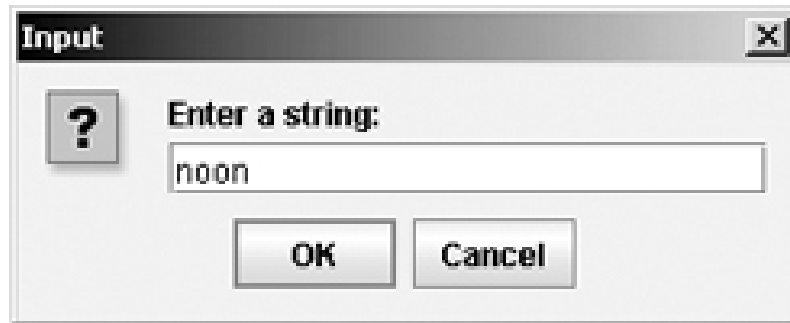
- To convert a string to a **double** value use:
`Double.parseDouble(str)`
- To convert a string to an **int** value use:
`Integer.parseInt(str)`

CheckPalindrome.java

- This example writes a program that prompts the user to enter a string and reports whether the string is a palindrome.
- A string is a palindrome if it reads the same forward and backward.
- The words "mom," "dad," and "noon," for instance, are all palindromes.
- Program: [CheckPalindrome.java](#)

CheckPalindrome.java

- The program checks whether a string is a palindrome





Character Class



Character Class

- Java provides a class for every primitive data type.
- These classes are `Character`, `Boolean`, `Byte`, `Short`, `Integer`, `Long`, `Float`, and `Double`
- These classes are in the `java.lang` package.
- They enable the primitive data values to be treated as objects.
- They also contain useful methods for processing primitive values.
- This section introduces the `Character` class.

Character Class

- The `Character` class has a constructor and several methods.

`java.lang.Character`

```
+Character(value: char)
+charValue(): char
+compareTo(anotherCharacter: Character): int
+equals(anotherCharacter: Character): boolean
+isDigit(ch: char): boolean
+isLetter(ch: char): boolean
+isLetterOrDigit(ch: char): boolean
+isLowerCase(ch: char): boolean
+isUpperCase(ch: char): boolean
+toLowerCase(ch: char): char
+toUpperCase(ch: char): char
```

Constructs a character object with char value.

Returns the char value from this object.

Compares this character with another.

Returns true if this character is equal to another.

Returns true if the specified character is a digit.

Returns true if the specified character is a letter.

Returns true if the character is a letter or a digit.

Returns true if the character is a lowercase letter.

Returns true if the character is an uppercase letter.

Returns the lowercase of the specified character.

Returns the uppercase of the specified character.

Character Class

```
Character charObject = new Character('b');
```

```
charObject.equals(new Character('b')) returns true
```

```
charObject.equals(new Character('d')) returns false
```

CountEachLetter.java

- This example presents a program that prompts the user to enter a string and counts the number of occurrences of each letter in the string regardless of case
- Program: [CountEachLetter.java](#)



StringBuffer Class



StringBuffer Class

- `StringBuffer` class is an alternative to the `String` class.
- `StringBuffer` is more flexible than `String`.
- You can add, insert, or append new contents into a string buffer, whereas the value of a `String` object is fixed once the string is created.
- You may replace `StringBuffer` by `StringBuilder`. The program can compile and run without any other changes.
- The `StringBuffer` class has three constructors and more than thirty methods

StringBuffer Constructors

- **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");
```

Inserting Contents into a String Buffer

```
strBuf.insert(11, "HTML and ");
```

- This code inserts "HTML and " at position 11 in strBuf (just before J). The new strBuf is "Welcome to HTML and Java".

StringBuffer Class

- `strBuf.delete(8, 11)` changes the buffer to Welcome Java.
- `strBuf.deleteCharAt(8)` changes the buffer to Welcome o Java.
- `strBuf.reverse()` changes the buffer to avaJ ot emocleW.
- `strBuf.replace(11, 15, "HTML")` changes the buffer to Welcome to HTML.
- `strBuf.setCharAt(0, 'w')` sets the buffer to welcome to Java.

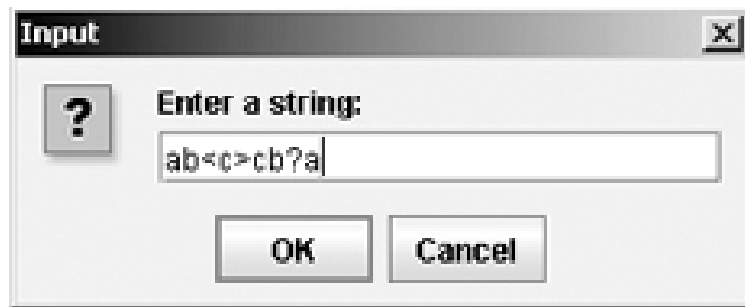
StringBuffer Class

- `toString()` method returns the string from the string buffer.
- `capacity()` method returns the current capacity of the string buffer. The capacity is the number of characters it is able to store without having to increase its size.
- `length()` method returns the number of characters actually stored in the string buffer.
- `setLength(newLength)` method sets the length of the string buffer.

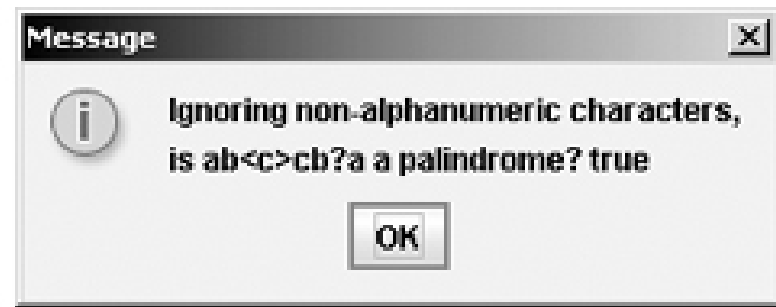
Strings

PalindromelgnoreNonAlphanumeric.java

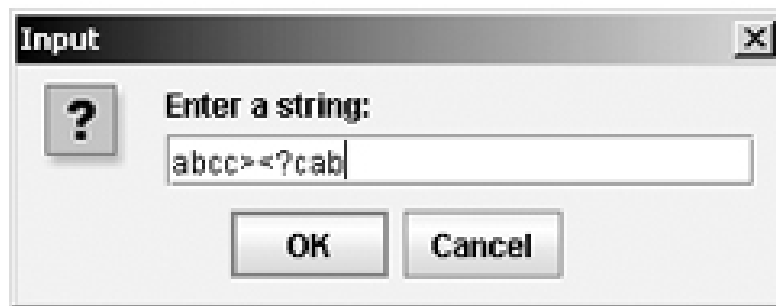
- This program ignores nonalphanumeric characters in checking whether a string is a palindrome:
 - [PalindromelgnoreNonAlphanumeric.java](#)



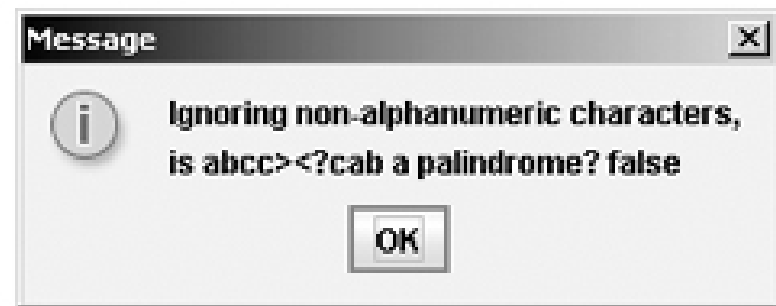
Input dialog box with a question mark icon. The text reads "Enter a string:" followed by a text input field containing "ab<c>cb?a". Below the input field are "OK" and "Cancel" buttons.



Message dialog box with an information icon. The text reads "Ignoring non-alphanumeric characters, is ab<c>cb?a a palindrome? true". Below the text is an "OK" button.



Input dialog box with a question mark icon. The text reads "Enter a string:" followed by a text input field containing "abcc><?cab". Below the input field are "OK" and "Cancel" buttons.



Message dialog box with an information icon. The text reads "Ignoring non-alphanumeric characters, is abcc><?cab a palindrome? false". Below the text is an "OK" button.

Command-Line Arguments



Command-Line Arguments

- The `main` method is just like a regular method with a parameter.
- It has parameter `args` of `String[]` type.
- You can call a regular method by passing actual parameters.
- Can you pass arguments to `main`?

Command-Line Arguments

- A main method is just a regular method.

```
public class A {  
    public static void main(String[] args) {  
        String[] strings = {"New York",  
                            "Boston", "Atlanta"};  
        B.main(strings);  
    }  
}
```

```
class B {  
    public static void main(String[] args) {  
        for (int i = 0; i < args.length; i++)  
            System.out.println(args[i]);  
    }  
}
```

Command-Line Arguments

- You can pass arguments from the command line
- For example starts the program `TestMain` with three strings: `arg0`, `arg1`, and `arg2`:

```
java TestMain arg0 arg1 arg2
```
- In the main method, get the arguments from `args[0]`, `args[1]`, ..., `args[n]`, which corresponds to `arg0`, `arg1`, ..., `argn` in the command line.
- `arg0`, `arg1`, and `arg2` are strings, but they don't have to appear in double quotes on the command line. The strings are separated by a space.

Command-Line Arguments

- A string that contains a space must be enclosed in double quotes. Consider the following command line:

```
java TestMain "First num" alpha 53
```

- It starts the program with three strings: "First num" and alpha, and 53, a numeric string.
- Note that alpha and 53 are actually treated as a string. You can use "alpha" and "53".

Calculator.java

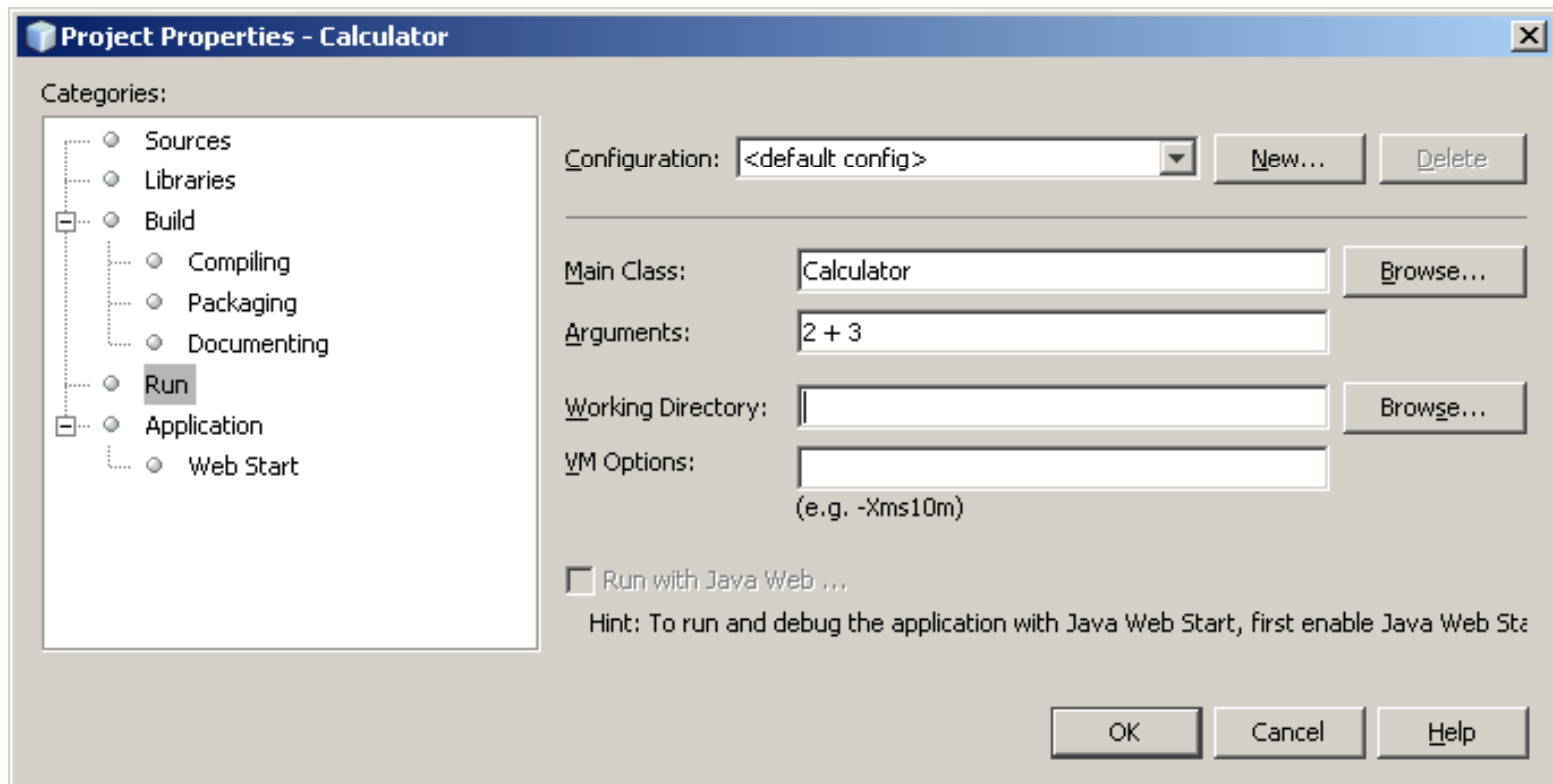
- `Calculator.java` receives three arguments: an integer followed by an operator and another integer.
- The program:
 - `Calculator.java`

Calculator.java

- We can run Calculator.java in three ways:
 - Using Windows command line:
`java Calculator 2 + 3`
 - Calling program from another class:
`TestCalculator.java`
 - Passing runtime arguments to programs by Netbeans (next slide)

Calculator.java

- We can pass runtime arguments to programs by Netbeans: Project Properties (right-click on the project name)=> run => **Arguments = 2 + 3**





References



References

- Y. Daniel Liang, **Introduction to Java Programming**, Sixth Edition, Pearson Education, 2007. (Chapter 8)



The End

