The String Class

- Constructing a String

  String message1 = new String("Welcome to Java");
  String s1 = new String();

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

  String message1 = "Welcome to Java";
  String s2 = "";

Strings Are Immutable

- A String object is immutable; its contents cannot be changed

- Does the following code change the contents of the string?

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

  Returns true if this string is equal to string s1.
  Returns true if this string is equal to string s1 case-insensitive.
  Returns true if the specified substring of this string exactly matches the specified substring in string s1.
  Returns true if this string ends with the specified suffix.
String Comparisons

```
equals
String s1 = new String("Welcome");
String s2 = new String("welcome");
if (s1.equals(s2)) {
    // s1 and s2 have the same contents
} else if (s1 == s2) {
    // s1 and s2 have the same reference
}
```

String Comparisons, cont.

```
compareTo
String s1 = new String("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 contents
} else {
    // s1 is less than s2
}
```

Finding String Length

```
Finding string length using the length() method:
message = "Welcome"
message.length()
```

Retrieving Individual Characters in a String

```
Use message.charAt(index)
Index starts from 0
```

String Length, Characters, and Combining Strings

```
java.lang.String

- length(): int
  Returns the number of characters in this string.
- charAt(index: int): char
  Returns the character at the specified index from this string.
- concate(s1: String): String
  Returns a new string that concatenate this string with string s1.
```

String Concatenation

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

```java
s1 + s2 + s3 + s4 + s5
((s1.concat(s2)).concat(s3)).concat(s4)).concat(s5);
```
### Extracting Substrings

```java
java.lang.String

substring(beginIndex: int): String
substring(beginIndex: int, endIndex: int): String
```

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 8.6. Returns this string's substring that begins at the specified beginIndex and extends to the character at index endIndex - 1, as shown in Figure 8.6. Note that the character at endIndex is not part of the substring.

### Extracting Substrings

#### Using the `substring` method in the `String` class

<table>
<thead>
<tr>
<th>String s1 = &quot;Welcome to Java&quot;;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indices</td>
</tr>
<tr>
<td>message</td>
</tr>
</tbody>
</table>

```
message.substring(0, 11)  // From index 0 to 10
message.substring(11)     // From index 11 to the end
```

### Converting, Replacing, and Splitting Strings

```java
java.lang.String

toLowerCase(): String
toUpperCase(): String
trim(): String
replace(oldChar: char, newChar: char): String
replaceFirst(oldString: String, newString: String): String
replaceAll(oldString: String, newString: String): String
split(delimiter: String): String
```

Returns a new string with all characters converted to lowercase.

Returns a new string with all characters converted to uppercase.

Returns a new string with blank characters trimmed on both sides.

Returns a new string that replaces all matching character in this string with the new character.

Returns a new string that replaces the first matching substring in this string with the new substring.

Returns a new string that replace all matching substrings in this string with the new substring.

Returns an array of strings consisting of the substrings split by the delimiter.

### Examples

```
"Welcome".toLowerCase()  // "welcome"
"Welcome".toUpperCase()    // "WELCOME"
"Welcome".replace(‘e’, ‘A’) // "WalcomA"
```

### Splitting a String

```
String str = "Cat-Dog-Mouse";
String[] tokens = str.split("-");
System.out.println(tokens[2]);  // "Mouse"
System.out.println("eats");    // False
System.out.println(tokens[0]);  // "Cat"
```

### Finding a Character or a Substring in a String

```java
java.lang.String

indexOf(ch: char): int
indexOf(ch: char, fromIndex: int): int
indexOf(s: String): int
indexOf(s: String, fromIndex: int): int
lastIndexOf(ch: char): int
lastIndexOf(ch: char, fromIndex: int): int
lastIndexOf(s: String): int
lastIndexOf(s: String, fromIndex: int): int
```

Returns the index of the first occurrence of ch in the string. Returns -1 if not matched.

Returns the index of the first occurrence of ch after fromIndex in this string. Returns -1 if not matched.

Returns the index of the last occurrence of ch in the string. Returns -1 if not matched.

Returns the index of the last occurrence of ch before fromIndex in this string. Returns -1 if not matched.
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.
"Welcome to Java".lastIndexOf('a') returns 14.

Problem: 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.

CountEachLetter.java

Problem: Finding Palindromes

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

CheckPalindrome.java

Review

- String class – an immutable sequence of characters
- Character class – a character

Review questions

- Suppose s is a string with the value "java". What will be assigned to x if you execute the following code?
  ```java
  char x = s.charAt(4);
  ```
  - A. 'a'
  - B. 'v'
  - C. Nothing will be assigned to x, because the execution causes the runtime error StringIndexOutOfBoundsException.

The Character Class

```
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 equals to another
- Returns true if the specified character is a digit
- Returns true if the specified character is a letter
- Returns true if the specified character is a letter
- 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
Review questions

Suppose s1 and s2 are two strings. Which of the following statements or expressions is incorrect?

A. String s3 = s1 - s2;
B. boolean b = s1.compareTo(s2);
C. char c = s1[0];
D. char c = s1.charAt(s1.length());

---

StringBuilder/StringBuffer

An alternative to the String class — a mutable string
In general, a StringBuilder/StringBuffer can be used wherever a string is used
StringBuilder/StringBuffer is more flexible than String
You can add, insert, or append new contents into a string buffer
A String object is fixed once the string is created.

---

Review questions

Analyze the following code.

```java
class Test {
    public static void main(String[] args) {
        String s;
        System.out.println("s is "+ s);
    }
}
```

A. The program has a compilation error because s is not initialized, but it is referenced in the println statement.
B. The program has a runtime error because s is not initialized, but it is referenced in the println statement.
C. The program has a runtime error because s is null in the println statement.
D. The program compiles and runs fine.

---

Outline

- StringBuilder/StringBuffer class
- File class and File I/O

---

StringBuilder Constructors

```java
java.lang.StringBuilder

StringBuilder() Constructs an empty string builder with capacity 16.
StringBuilder(capacity: int) Constructs a string builder with the specified capacity.
StringBuilder(s: String) Constructs a string builder with the specified string.
```

---

Modifying Strings in the Builder

```java
java.lang.StringBuilder

append(data: char[]): StringBuilder Appends a char array into this string builder.
append(data: char[], offset: int, len: int): StringBuilder Appends a subarray in data into this string builder.
append(v: aPrimitiveType): StringBuilder Appends a primitive type value as a string to this builder.
append(s: String): StringBuilder Appends a string to this string builder.
delete(startIndex: int, endIndex: int): StringBuilder Deletes characters from startIndex to endIndex.
deleteCharAt(index: int): StringBuilder Deletes a character at the specified index.
insert(index: int, data: char[]): StringBuilder Inserts a subarray of the data in the array to the builder at the specified index.
insert(offset: int, data: char[]): StringBuilder Inserts data into this builder at the position offset.
insert(offset: int, b: aPrimitiveType): StringBuilder Inserts a value converted to a string into this builder.
insert(offset: int, s: String): StringBuilder Inserts a string into this builder at the position offset.
replace(startIndex: int, endIndex: int, s: String): StringBuilder Replaces the characters in this builder from startIndex to endIndex with the specified string.
reverse(): StringBuilder Reverses the characters in the builder.
setCharAt(index: int, ch: char): void Sets a new character at the specified index in this builder.
```
The toString, capacity, length, setLength, and charAt Methods

<table>
<thead>
<tr>
<th>java.lang.StringBuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>toString(): String</td>
</tr>
<tr>
<td>capacity(): int</td>
</tr>
<tr>
<td>charAt(index: int): char</td>
</tr>
<tr>
<td>length(): int</td>
</tr>
<tr>
<td>substring(startIndex: int): String</td>
</tr>
<tr>
<td>substring(startIndex: int, endIndex: int): String</td>
</tr>
<tr>
<td>trimToSize(): void</td>
</tr>
<tr>
<td>length(): int</td>
</tr>
</tbody>
</table>

Problem: Checking Palindromes Using StringBuilder

- Rewrite the palindromes program that checks whether a string is a palindrome by using StringBuilder class

  CheckPalindromeStringBuilder.java

The File Class

- File class is used to obtain file properties and to delete and rename files
- The File class is intended to provide an abstraction that deals with most of the machine-dependent complexities of files and path names in a machine-independent fashion
- The File class is a wrapper class for the file name and its directory path.

  It is not for reading and writing file contents.

Problem: Explore File Properties

- Let's test the File class

  TestFileClass.java

File I/O

- A File object encapsulates the properties of a file or a path but does not contain the methods for reading/writing data from/to a file

  In order to perform I/O, we need to use I/O classes: Scanner and PrintWriter
Writing Data Using PrintWriter

<table>
<thead>
<tr>
<th>PrintWriter(filename: String)</th>
<th>Creates a PrintWriter for the specified file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(s: String): void</td>
<td>Writes a string.</td>
</tr>
<tr>
<td>print(c: char): void</td>
<td>Writes a character.</td>
</tr>
<tr>
<td>print(cArray: char[]): void</td>
<td>Writes an array of characters.</td>
</tr>
<tr>
<td>print(i: int): void</td>
<td>Writes an int value.</td>
</tr>
<tr>
<td>print(l: long): void</td>
<td>Writes a long value.</td>
</tr>
<tr>
<td>print(f: float): void</td>
<td>Writes a float value.</td>
</tr>
<tr>
<td>print(d: double): void</td>
<td>Writes a double value.</td>
</tr>
<tr>
<td>print(b: boolean): void</td>
<td>Writes a boolean value.</td>
</tr>
<tr>
<td>Also contains the overloaded println methods.</td>
<td></td>
</tr>
<tr>
<td>Also contains the overloaded printf methods.</td>
<td></td>
</tr>
</tbody>
</table>

PrinterWriter Example

- Write student scores to a file named “scores.txt”
- WriteData.java

Reading Data Using Scanner

<table>
<thead>
<tr>
<th>Scanner(source: File)</th>
<th>Creates a Scanner that produces values scanned from the specified file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanner(source: String)</td>
<td>Creates a Scanner that produces values scanned from the specified string.</td>
</tr>
<tr>
<td>close()</td>
<td>Closes the scanner.</td>
</tr>
<tr>
<td>hasNext(): boolean</td>
<td>Returns true if this scanner has another token in its input.</td>
</tr>
<tr>
<td>next(): String</td>
<td>Returns next token as a string.</td>
</tr>
<tr>
<td>nextByte(): byte</td>
<td>Returns next token as a byte.</td>
</tr>
<tr>
<td>nextShort(): short</td>
<td>Returns next token as a short.</td>
</tr>
<tr>
<td>nextInt(): int</td>
<td>Returns next token as an int.</td>
</tr>
<tr>
<td>nextLong(): long</td>
<td>Returns next token as a long.</td>
</tr>
<tr>
<td>nextFloat(): float</td>
<td>Returns next token as a float.</td>
</tr>
<tr>
<td>nextDouble(): double</td>
<td>Returns next token as a double.</td>
</tr>
<tr>
<td>useDelimiter(pattern: String): Scanner</td>
<td>Sets this scanner’s delimiting pattern.</td>
</tr>
</tbody>
</table>

Scanner example

- Reading data from the score file named “scores.txt”
- ReadData.java

Reading data using Scanner class

- Read data from keyboard
  Scanner input = new Scanner(System.in);
- Read data from a file
  Scanner input = new Scanner(new File(filename));

Problem: Replacing Text

Write a class named ReplaceText that replaces a string in a text file with a new string. The filename and strings are passed as command-line arguments as follows:

java ReplaceText sourceFile targetFile oldString newString

ReplaceText.java
Review questions

The following program displays __________.

```java
public class Test {
    public static void main(String[] args) {
        String s = "Java";
        StringBuilder buffer = new StringBuilder(s);
        change(buffer);
        System.out.println(buffer);
    }

    private static void change(StringBuilder buffer) {
        buffer.append(" and HTML");
    }
}
```

A. Java
B. Java and HTML
C. and HTML
D. nothing is displayed

Review questions

Which class contains the method for checking whether a file exists?

A. File
B. PrintWriter
C. Scanner
D. System

Which class do you use to write data into a text file?

A. File
B. PrintWriter
C. Scanner
D. System

Which class do you use to read data into a text file?

A. File
B. PrintWriter
C. Scanner
D. System

Practice problems

- String class
- StringBuilder class
- Command line arguments
- File class
- File I/O