Chapter 8 Strings and Text I/O

1.

```
s1 == s2  => true
s2 == s3  => false
s1.equals(s2) => true
s2.equals(s3) => true
s1.compareTo(s2) => 0
s2.compareTo(s3) => 0
s1 == s4 => true
s1.charAt(0) => W
s1.indexOf('j') => -1
s1.indexOf("to") => 8
s1.lastIndexOf('a') => 14
s1.lastIndexOf("o", 15) => 9
s1.length() => 16
s1.substring(5) => me to Java!
s1.substring(5, 11) => me to
s1.startsWith("Wel") => true
s1.endsWith("Java") => true
s1.toLowerCase() => welcome to java!
s1.toUpperCase()=> WELCOME TO JAVA!
"  Welcome ".trim() => Welcome
s1.replace('o', 'T') => WelcTme tT Java!
s1.replaceAll("o", "T") => WelcTme tT Java!
s1.replaceFirst("o", "T") => WelcTme tT Java!
s1.toCharArray() returns an array of characters consisting of W, e, l, c, o, m, e,   , t, o,  ,  J, a, v, a

(Note that none of the operation cause the contents of a string to change)
```

2. `String s = new String("new string");`
   Answer: Correct

   String s3 = s1 + s2;
   Answer: Correct

   String s3 = s1 - s2;
   Answer: Incorrect

   s1 == s2
   Answer: Correct

   s1 >= s2
   Answer: Incorrect

   s1.compareTo(s2);
   Answer: Correct

   int i = s1.length();
   Answer: Correct

   char c = s1(0);
   Answer: Incorrect

char c = s1.charAt(s1.length());

Answer: Incorrect : it's out of bounds, even if the preceding problem is fixed.

3. Use the method `equalsIgnoreCase`.
4. Use the methods `toUpperCase` and `toLowerCase`. The conversion methods (toLowerCase, toUpperCase, trim, replace) don't change the contents of the string that invokes these methods. These methods create new strings.
5. 0.
6. Use the overloaded static valueOf method in the String class.
7. The text is declared in Line 2 as a data field, but redeclared in Line 5 as a local variable. The local variable is assigned with the string passed to the constructor, but the data field is still null. In Line 10, test.text is null, which causes NullPointerException when invoking the toLowerCase() method.
8. The constructor is declared incorrectly. It should not have void.

9. A lowercase letter is between 'a' and 'z'. You can use the static isLowerCase(char) method in the Character class to test if a character is in lowercase. An uppercase letter is between 'A' and 'Z'. You can use the static isUpperCase(char) method in the Character class to test if a character is in uppercase.
10. An alphanumeric character is between '0' and '9', or 'A' and 'Z', or 'a' and 'z'. You can use the static isLetterOrDigit(char ch) method in the Character class to test if a character is a digit or a letter.
11. The StringBuilder class, introduced in JDK 1.5, is similar to StringBuffer except that the update methods in StringBuffer are synchronized.
12. Use the StringBuffer's constructor to create a string buffer for a string, and use the toString method in StringBuffer class to return a string from a StringBuffer.
13. StringBuffer sb = new StringBuffer(s);
    sb.reverse();
    s = sb.toString();
14. StringBuffer sb = new StringBuffer(s);
    sb.delete(4, 10);
    s = sb.toString();
15. Both string and string buffer use arrays to hold characters. The array in a string is fixed once a string is created. The array in a string buffer may change if the buffer capacity is changed. To accommodate the change, a new array is created.
16.

    (1) Java is fun

    (2) JavaHTML

    (3) Jais funva

(4) JHTMLava

(5) v

(6) 4

(7) Jav

(8) Ja

(9) avaJ

(10) JComputera

(11) av

(12) va

17.

The output is

Java
Java and HTML

NOTE:

Inside the method, the statement s = s + " and HTML" creates a new String object s, which is different from the original String object passed to the change(s, buffer) method. The original String object has not been changed. Therefore, the printout from the original string is Java.

Inside the method, the content of the StringBuffer object is changed to Java and HTML. Therefore, the printout from buffer is Java and HTML.

18.

```
public static void main(String[] args)
```

can be replaced by

```
public static void main(String args[])

public static void main(String[] x)

public static void main(String x[])
```

but not

```
static void main(String x[])
```

because it is not public.

19.

(1)
Number of strings is 4
I
have
a
dream

(2)
Number of strings is 1
1 2 3

(3)
Number of strings is 0

(4)
Number of strings is 1
*
(5)
Number of strings is (the number of files and directory from where the command is executed)
Displays all files and directory names in the directory where the command is executed.

20. See Section 8.6.2 on Regular Expression Syntax

21.

```
System.out.println("abc".matches("a[\\w]c")); => true
System.out.println("12Java".matches("[\\d]{2}[\\w]{4}"));  => true
System.out.println("12Java".matches("[\\w]*")); => true
System.out.println("12Java".matches("[\\w]*")); => true
System.out.println("12Java".matches("[\\d]*"));  => false
System.out.println("12Java".matches("[\\d]{2}[\\w]")); => false
System.out.println("12Java".matches("[\\d]{2}[\\w]*")); => true
System.out.println("12Java".matches("[\\d]{2}[\\w]+")); => true
System.out.println("12Java".matches("[\\d]{1}[\\w]")); => false
System.out.println("12Java".matches("[\\d]{1}[\\w]*")); => true
System.out.println("12_Java".matches("[\\d]{2}[\\w]{5}"));  => true
System.out.println("12Java".matches("[\\d]{2}[\\w]{1,15}"));  => true
System.out.println("12#Java".matches("[\\d]{2}[\\w]{1,15}"));   => false
```

22.

```
System.out.println("Java".replaceAll("[av]", "RX")); => JRXRXRX
System.out.println("Java".replaceAll("av", "RX")); => JRXa
System.out.println("Java".replaceAll("[a][v]", "RX")); => JRXa
System.out.println("Java".replaceFirst("\\w", "RX")); => RXava
System.out.println("Java".replaceFirst("\\w*", "RX")); => RX
System.out.println("Java12".replaceAll("\\d", "RX")); => JavaRXRX
System.out.println("Java".replaceAll("\\d", "RX")); => Java
System.out.println("Java".replaceAll("\\W", "RX")); => Java
System.out.println("Java".replaceAll("\\D", "RX")); => RXRXRXRX
System.out.println("Java".replaceAll("\\w*", "RX")); => RXRX
System.out.println("Java".replaceAll("\\w+", "RX")); => RX
System.out.println("Java".replaceAll("\\w+?", "RX")); => RXRXRXRX
System.out.println("Java".replaceFirst("\\w+?", "RX")); => RXava
```

23.
```
"Java".split("[a]") => J, v
"Java".split("[av]")  => J
"Java#HTML#PHP".split("#") => Java, HTML, PHP
"JavaTOHTMLToPHP".split("TO|To") => Java, HTML, PHP
"JavaTOHTMLToPHP".split("TH") => JavaTOHTMLToPHP
"JavaTOHTMLToPHP".split("T|H") => Java, O, , MLRXRX
```

24. The \ is a special character. It should be written as \\ in Java using the Escape sequence.

25. Use exists() in the File class to check whether a file exists. Use delete() in the File class to delete this file. Use renameTo(File) to rename the name for this file. You cannot find the file size using the File class.

26. No. The File class can be used to obtain file properties and manipulate files, but cannot perform I/O.

27. To create a Formatter for a file, use new Formatter(new File(filename)). This statement may throw an exception. Java forces you to write the code to deal with exceptions. One way to deal with it is to declare throws Exception in the method declaration. If the close() method is not invoked, the data may not be saved properly.

28. The contents of the file temp.txt is:

```
amount is 32.320000 3.232000e+01
amount is 32.3200 3.2320e+01
 false
  Java
```

29. No. The line separator on Windows is \r\n.
30. Create a Formatter for a StringBuilder or StringBuffer using new Formatter(StringBuilder/StringBuffer) and use the format method to append data to the string builder or string buffer.
31. To create a Scanner for a file, use new Scanner(new File(filename)). This statement may throw an exception. Java forces you to write the code to deal with exceptions. One way to deal with it is to declare throws Exception in the method

declaration. If the close() method is not invoked, the problem will run fine. But it is a good practice to close the file to release the resource on the file.

32. If you attempt to create a Scanner for a nonexistent file, an exception will occur. If you attempt to create a Formatter for an existing file, the contents of the existing file will be gone.