STRINGS

zyBook 2.18, zyBook 2.19, zyBook 3.16, zyBook 3.17, zyBook 3.19,
Java String API, Java Object API: equals method, Oracle - Java Tutorial: Strings

@ Dr. Jessica Young Schmidt and NCSU Computer Science Faculty

STRING CLASS

- The String class represents character strings.
- String objects are immutable and cannot change.
- String objects are created and assigned like primitive values:
  - String <name > = "<text >";
  - String <name > = <expression >;
- For example:

```
String department = “CSC”;
int courseNum = 116;
String course = department + courseNum;
```

STRING INDEXING

- String objects consist of a list of characters (char)
- Characters are numbered internally with an index
- Index starts at 0
- For example,

```
String lastName = "Bai";
```

0	1	2
B	a	i

HOW MANY CHARACTERS ARE IN THE STRING ?

public int length()

- Returns: the length of the sequence of characters represented by this object.
- For example,

```
String lastName = “Bai";
int numLetters = lastName.length(); // 3 characters
```

0	1	2
B	a	i

WHAT IS THE N-TH CHARACTER OF THE STRING ?

public char charAt(int index)

- Parameters: index – the index of the char value.
- Returns: the char value at the specified index of this string. The first char value is at index 0.
- Throws: IndexOutOfBoundsException - if the index argument is negative or not less than the length of this string.
- For example,

```
String lastName = “Bai";
lastName.charAt(0);                          // ‘B’
lastName.charAt(lastName.length());          // IndexOutOfBoundsException
lastName.charAt(lastName.length() – 1);      // ‘i’
```

0	1	2
B	a	i

WHERE IS SUBSTRING IN THE STRING ?

public int indexOf(String str)

- Parameters: str – the substring to search for.
- Returns: the index of the first occurrence of the specified substring, or -1 if there is no such occurrence.
- For example,

```
String lastName = “Bai";
lastName.indexOf(“i”);        // 2
lastName.indexOf(“ai”);       // 1
lastName.indexOf(“hi”);       // -1
```

0	1	2
B	a	i

SUBSTRINGS

public String substring(int beginIndex)

- Parameters: beginIndex – the beginning index, inclusive.
- Returns: the specified substring.
- Throws: IndexOutOfBoundsException – if beginIndex is negative or larger than the length of this String object.
- For example,

```
String lastName = “Bai";
lastName.substring(1);          // “ai”
```

0	1	2
B	a	i

SUBSTRINGS

public String substring(int beginIndex, int endIndex)

- Parameters:

  beginIndex – the beginning index, inclusive. endIndex – the ending index, exclusive.

- Returns: the specified substring.
- Throws: IndexOutOfBoundsException – if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.
- For example,

```
String lastName = “Bai";
lastName.substring(0, 2);        // “Ba”
```

0	1	2
B	a	i

UPPERCASE AND LOWERCASE

public String toLowerCase()

- Returns: the String, converted to lowercase.

public String toUpperCase()

- Returns: the String, converted to uppercase.

For example,

```
String lastName = “Bai";
String lower = lastName.toLowerCase();          // “bai”
String upper = lastName.toUpperCase();          // “BAI”
```

HOW TO DETERMINE IF TWO STRING OBJECTS ARE EQUAL?

public boolean equals(Object anObject)

- Overrides: equals in class Object
- Parameters: anObject – The object to compare this String against
- Returns: true if the given object represents a String equivalent to this string, false otherwise
- For example,

```
String lastName = “Bai";
lastName.equals(“Bai”);          // true
lastName.equals(“Gina”);         // false
```

HOW TO COMPARE TWO STRING OBJECTS?

public int compareTo(String anotherString)

- Parameters: anotherString – the String to be compared.
- Returns: the value 0 if the argument string is equal to this string; a value less than 0 if this string is lexicographically less than the string argument; and a value greater than 0 if this string is lexicographically greater than the string argument.
- For example,

```
String lastName = “Bai";
lastName.compareTo(“Bai”);          // 0
lastName.compareTo(“BAI”);          // -32
lastName.compareTo(“bai”);          // 32
```

ESCAPE SEQUENCES

Escape sequences have special meaning to the compiler and begin with

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\n	Insert a newline in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.