ENSF 593/594

6 - Java Strings

A Quick Overview of String and Character Processing

Introduction to java.lang Classes

- String and character processing
  - Class `java.lang.String`
  - Class `java.lang.StringBuffer`
  - Class `java.lang.Character`
  - Class `java.util.StringTokenizer`

String Class

- Java provides built-in support for strings:
  - A String object is implicitly created when a string literal (e.g. `"a string"`) is found in source code.
- String objects are fixed-length and read-only.
- Class String provides several consturtors:
  - String s1 = new String();
  - String s2 = new String(“ABC”);
  - String s3 = new String (String other);
  - String s4 = new String (charArray);
  - String s5 = new String(charArray, 5, 2); // copies subset
  - String s6 = new String(byteArray);
  - String s7 = new String (byteArray, 4, 4); // copies subset

Strings… (continued)

- The String class has many useful methods, including:
  - S1.length()
  - S1.charAt(int index)
  - S1.getChars: gets entire set of character in String
    s1.getChars(0, 5, charArray, 0); // copies some of s1 to charArray
  - S1.indexOf(char ch)
  - S1.lastIndexOf(char ch)
  - S1.compareTo(String s2) // uses lexicographical comparison
  - S1.equals (String s2) // uses lexicographical comparison
  - S1.substring(int beginIndex, int endIndex)
  - S1.startWith("all") // returns true if s1 starts with "all"
  - S1.endWith("est"); // returns true of s1 ends with "est"
  - S1.substring(3, 6); // returns a subset starting at 3, ending at 6
  - S1.substring(4); // returns a subset starting at 4, to the end
  - etc.

Strings… (continued)

- The String class has many useful methods, including:
  - S1.concat(s2) // concatenates s2 and s1
  - S1.replace('m', 'M'); // replace every occurrence of m with M
  - S1.toLowerCase(); // converts every character to lower case
  - S1.toUpperCase(); // converts every character to upper case
  - S1.trim(); // trims leading and trailing white-spaces
  - String.valueOf(45); // returns string representation of 45
  - etc.

  - String objects can be be concatenated with the + and += operators.
    - Since strings are fixed length, a third string is actually created when two strings are concatenated.

Strings… (continued)

- Most classes provide a toString() method so that an object can be concatenated or printed out using println().

Example:

```
Public class Point
{
    private double x, y;
    …
    public String toString() {
        return "(" + x + "," + y + ")";
    }
}
```

Converting Strings to Primitive Data Types

- You can convert primitive types to strings by using one of the valueOf() class methods.
- Example:

```
int ival = 3;
String s = String.valueOf(ival);
```

- You can convert a string to a primitive type by using one of the following:
  - Integer.parseInt(String s)
  - new Float(String s).floatValue()
  - new Double(String s).doubleValue()
- Example:

```
int i = Integer.parseInt("100");
float x = (new Float("33.34")).floatValue();
```

StringBuffer

- *Class* `StringBuffer`
  - Used for creating and manipulating dynamic string data
  - Can store characters based on capacity
    - Capacity expands dynamically to handle additional characters
  - Uses operators + and += for `String` concatenation
- `StringBuffer` *has three constructors*

```
StringBuffer b1 = new StringBuffer();
StringBuffer b2 = new StringBuffer( 100 );
StringBuffer b3 = new StringBuffer( "hello" );
```

- Default creates `StringBuffer` with no characters, and capacity of 16 characters.

StringBuffer

- Constructor Examples:

- Other Methods
  - length
  - capacity
  - setLength
  - charAt
  - setCharAt
  - getCharAt
  - reverse
  - append: allows to append values of data of strings
  - Insert: Examples: insert(i, " "): inserts a space at the ith element
  - deleteCharAt(i);
  - Etc.

Class java.util.StringTokenizer

- The following code segment shows how to use the objects of class StringTokenizer to partition a string based on a delimiter.
- The default delimiter is " \n\t\r". Which are all known white spaces:

```
StringTokenizer tokens =   new StringTokenizer("cat dog fish");
System.out.println( "Number of tokens: " + tokens.countTokens());
while ( tokens.hasMoreTokens() )  {
            System.out.println( tokens.nextToken() );
}
```