# Strings

# Strings

- Strings are fundamental part of all computing languages.
- At the basic level, they are just a data structure that can hold a series of characters
- However, strings are not implemented as a character array in Java as in other languages.

# Strings in Java

- Strings are implemented as two classes in Java
- java.lang.String provides an unchangeable String object
- java.lang.StringBuffer provides a String object that can be amended

# Basic String Methods

- length() returns the length of the string
- toLowerCase() converts the string to lower case
- toUpperCase() converts the string to upper case
- replace(char, char) replaces occurrences of one character with another character

# Basic Strings continued

- Basic strings are not meant to change frequently so there are no add or append methods
- However the concat(String) method does allow two strings to be concatenated together

# Basic Strings continued

- Substrings of a String object can also be accessed
- A portion of String object can be copied to a character array using the getChars() method
- The substring() method can return substring beginning at a specified offset

# Searching a string

- Methods for searching strings
  - indexOf(*x*) searches for the first occurrence of *x*
  - indexOf(*x*, *y*) searches for the first occurrence of *x* after the offset of y
  - lastIndexOf(*x*) searches backwards for the first occurrence of *x*
  - lastIndexOf(*x*, *y*) searches backwards for the first occurrence of *x* after the offset of y

# Example of string search

- indexOf(*x*) and indexOf(*x*, *y*) can find all occurrences of a character(s) in a string

```
String str = new String("Wish You Were Here");
        int count = 0;
        int fromIndex = 0;
        while(fromIndex != -1) {
                fromIndex = str.indexOf("er", fromIndex);
         if (fromIndex != -1) {
                count++;
                fromIndex++;
        }
System.out.println(String.valueOf(count), 10, 10); }
```

# StringBuffer class

- The StringBuffer class is provided for strings that need may need to be changed
- The StringBuffer class contains methods for both inserting and appending text
- An object created as a StringBuffer can easily be converted to an object of the String class if needed

# More on StringBuffer Class

- Conversion may be needed because many Java library methods expect a string
- The toString() method is used for converting a StringBuffer object to a String object
- Example of converting a StringBuffer to a String:

```
public void paint(Graphics g) {
  StringBuffer buf = new StringBuffer("Hello, World");
  g.drawString(buf.toString(), 10, 10);
}
```

# More on StringBuffer Class

- StringBuffer objects are mutable and capacity & length affect performance
- If the StringBuffer object needs to be expanded during an append or insert, a new array is created and the old data copied to it
- Use capacity() and ensureCapacity(int) methods to minimize expansions

# Length v. Capacity

- The length() method returns the length of the string in the StringBuffer
- The capacity() method returns the total “space” in a StringBuffer
- The ensureCapacity(int) method insures the StringBuffer has at least the specified amount of capacity remaining

# Length v. Capacity con't

- Examples of length() and capacity() methods

```
StringBuffer buf = new StringBuffer(25);// creates
  StringBuffer with length 25
buf.append("13 Characters"); // appends 13
  characters
int len = buf.length(); // length() returns 13
int cap = buf.capacity(); // capacity returns 25
```

# Bibliography


- http://www.eimc.brad.ac.uk/java/tutorial/Project/4/string.htm
- http://www.scism.sbu.ac.uk/jfl/Appa/appa5.html
- http://docs.rinet.ru:8080/WebJPP/contents.htm
- http://www.javaworld.com/javaworld/jw-10-2002/jw-1004-java101guide.html