Application Programming

Week 4

Lecture 1

Deliverables

- Java Strings

Text in Java

- The `char` primitive type stores one character of the Unicode character set. (One letter)
- The `String` class facilitates handling multiple characters at once.
- A String can be 0 or more characters long.
  - “” is the empty string.
- Take a look at StringConstructors.java in the textbook.

Strings are Sequences of Characters

- You can get the character at an index in the string
  - starting with index 0

```
String stringObj = “This is a String”;
stringObj.charAt(index);
```

- Try this:

```
String test = "Hello";
System.out.println(test.charAt(0));
System.out.println(test.charAt(4));
```

- How would you get the second character?

String Manipulation

- In `java.lang.String`:
  - Use the `split()` method to get an array of Strings, based on a delimiter.
  - Use `trim()` to clear off any additional space from the text.
- Use `Double.parseDouble()` to translate a `String` into a `double` value.
- Use `Integer.parseInt()` to translate a `String` into a `int` value.

java.lang.String

- How do you add a double quote to a string?
  - for example to express: She said, "Hi there"
- To do this, use \"

```
String s = "She said, \"Hi there\"";
System.out.println(s);
```

>> She said, "Hi there"

- Other special characters:
  - \n for new line
  - \t for tab
  - \\ for \

java.lang.String

- Let’s test this out

```
String[] testString = {“Hi”,” class”};
System.out.println( testString );
```

- What is printed?

Parsing a Delimited String

- Use the `split()` method to get an array of Strings from a String.
  - This object method takes a delimiter as a parameter.

- Use `trim()` to clear off any additional space from the text
  - This object method takes no parameters.

Parsing a Delimited String

- Example:

```
String foo = “I’m out of candy corn. Send help!”;
for( int i = 0; i < foo.length(); i++ ){
 char c = foo.charAt(i);
 System.out.print( c );
}
String[] sentences = foo.split( “.” );
System.out.println( sentences[1].trim() );
```

Converting to a String

- Use `String.valueOf()` to get the String value of a given variable.
- This works on primitive types in Java.

```
String yae = String.valueOf( 350.4 ) + 1;
```

Comparing Strings

- Recall that in Java, primitive types can be compared with ==
- Since Strings are objects, the == operator will check to see if two Strings are the same instance, not whether they are equal in content.
- Use the equals() method to check whether two Strings contain the same text.

```
String a = new String( “#PSL4Life” );
String b = “#PSL4Life”;
boolean result1 = a.equals( b );
boolean result2 = (a==b);
```

Comparing Strings

- Check out these other useful methods for comparing Strings (in your textbook, or in the Java APIs):
  - `equalsIgnoreCase()`
  - `compareTo()`
  - `regionMatches()`