

CSC 175 – Intermediate Programming

Lecture 5 –Strings

How Do Computer Handle Character Data?

- Like all other data that a computer handles, characters are stored in numeric form. A particular code represents a particular character. The most commonly used code is ASCII (*American Standard Code for Information Interchange*). Java uses a code called Unicode.
- A character variable can be declared by writing:
`char c;`

Example: Comparing Characters

```
import java.util.Scanner;

public class CharTest {
    public static void main(String[] args) {
        char char1 = 'a', char2 = 'b', char3 = 'A';

        if (char1 > char2)
            System.out.println("Very good");
        else
            System.out.println("Very bad");

        if (char1 > char3)
            System.out.println("Very good");
        else
            System.out.println("Very bad");
    }
}
```

What are Objects?

- An object is similar to a variable, but it has its own properties and methods that are either written for it by the programmer or are a part of the Java Development Kit (JDK).
- When a programmer specifies all the properties and methods that a group of objects may have, (s)he has written a **class of objects**.

What are Strings?

- A collection of characters that are read and written together to form words, numbers and so on are called ***strings***. While strings are a built-in data type in some programming languages, this is not the case in Java; they are a ***standard class of objects***.
- Strings have certain methods that can be used to manipulate them. At the same time, they can be used in some ways that are like the basic data type in Java, such as **int**, **double** and **char**.

How does Java handle strings?

- In Java, a string is declared as an object, completed with a ***constructor call***.
`String s = new String();`
- Strings can be assigned values in a similar way to how numeric variables are assigned a value.
`s = "This is a test";`
- You can also initialize their value in the constructor call.
`String s = new
String("This is another test.");`

Java String Constructor - Another Example

```
public class TestString {
    public static void main(String[] args) {
        String s = new String("This is the first");
        String t = new String();
        t = "This is the second";
        System.out.println("Your string is \'\" + s
                            + "\'\".");
        System.out.println("Your other string is \'\"
                            + t + "\'\".");
    }
}
```

Output

```
Your string is 'This is the first'.
Your other string is 'This is the second'.
```

How does Java handle input and output?

- In Java, it is very easy to read in data as a string.
- Using a Scanner object, you can read in either up to the next white space character (blank, tab or newline) or the entire line.
- If you want to print a string, you use **System.out.println**, **System.out.print** or **System.out.printf**.

Java String Input/Output - An Example

```
import java.util.Scanner;
public class TestString {
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);
        String s = new String();
        System.out.println("Enter your string");
        s = keyb.next();
        System.out.println("Your string is \"
                            + s + "\".");
        System.out.println("Enter your string");
        s = keyb.nextLine();
        System.out.println("Your new string is \"
                            + s + "\".");
    }
}
```

Java String Input/Output – The output

Enter your string

This is the first

Your string is "This".

Enter your string

Your new string is " is the first".

The Java String methods

- The way in which you specify an object's method is to write:

objectName.methodName ();

- **s.length()** - Returns the number of characters in s
- **s.trim()** - Returns s with leading and trailing white space characters removed.
- **s.concat()** - Returns s with another string concatenated to it.
- **s.substring()** - Returns a substring of s
- **s.indexOf()** - Returns the starting position of the first occurrence of a substring within s.

s.length()

- **s.length()** returns the number of characters that s contains:

```
public class TestString {
    public static void main(String[] args) {
        String s = new String("This Is The " +
            "First");

        System.out.println(s.length());
    }
}
```

- This program prints 17

s.trim()

- Returns s with leading and trailing white space characters removed.

```
public static void main(String[] args) {
    String s
        = new String("  This Is The First  ");
    s = s.trim();
    System.out.println("My String is \'\" + s
        + "\'");
    System.out.println("It has " + s.length()
        + " characters.");
}
```

- The output is:

```
My String is 'This Is The First'
It has 17 characters.
```

s.concat()

- **s.concat()** returns s with another string concatenated to it.
- Example

```
public static void main(String[] args) {
    String s = new String("John"),
        t = new String("Smith"), u;
    u = s.concat(t);
    System.out.println("My String is \'\" + u
        + "\'");
}
```

- Output

```
My String is 'JohnSmith'
```

s.concat () – Another Example

- We should remember the blank space between the first and last names

```
public static void main(String[] args) {
    String s = new String("John"),
        t = new String("Smith"), u;
    u = s.concat(" ");
    u = u.concat(t);
    System.out.println("My String is \' " + u
        + "\'");
}
```

- Output

```
My String is 'John Smith'
```

s.concat () – Another Example

- We don't need the extra statement; we can cascade our methods

```
public static void main(String[] args) {
    String s = new String("John"),
        t = new String("Smith"), u;

    The object
    u = (s.concat(" ")).concat(t);
    System.out.println("My String is \' "
        + u + "\'");
}
```

- Output

```
My String is 'John Smith'
```


s.substring()

- **s.substring()** returns a substring of **s**.
- There are two such methods:
 - **s.substring(12)** returns a substring of **s** that contains everything after position 12.
 - **s.substring(12, 17)** returns a substring whose first character appears in position 12 and where the first character in **s** after the substring is 17.

s.substring(12) - An Example

```
public static void main(String[] args) {  
    String s = new String(  
        "The quick brown fox jumped over the lazy dogs"),  
        t = new String();  
    t = s.substring(31);  
    System.out.println("My String is\n \'"  
        + t + "\'");  
}
```

- Output

```
My String is  
'own fox jumped over the lazy dogs'
```

s.substring(12) - An Example

```
public static void main(String[] args) {
    String s = new String(
        "The quick brown fox jumped over the lazy dogs"),
        t = new String();
    t = s.substring(12);
    System.out.println("My String is\n \'
                        + t + "\'");
}
```

- Output

```
My String is
'own fox jumped over the lazy dogs'
```

s.indexOf()

- **s.indexOf()** can be used to find where a substring appears within **s**.
- Example

```
public static void main(String[] args) {
    String
        s = new String("John Francis Xavier Smith"),
        t = new String();
    int    i = s.indexOf("Fran");
    t = s.substring(i, i+7);
    System.out.println("My String is \'
                        + t
                        + "\'");
}
```

- Output

```
My String is 'Francis'
```

`s.charAt ()`

- `s.charAt (i)` returns the character at position `i`.
- This is a useful tool if you need to extract a single character from a string.

`s.charAt ()` - An Example

```
public static void main(String[] args) {
    String
        s = new String("John Francis Johnson"),
    int    i = 6;
    char   c;

    c = s.charAt(i);
    System.out.println("Character # " + i
                       + " is " + c);
}
```

- Output
Character # 6 is r

Comparing Strings

- We do not use the normal relational operators to compare strings, even though the compiler will not give an error message. This is something to be discussed at a later date.
- We compare strings using the methods, **`equals`**, **`equalsIgnoreCase`** and **`compareTo`**.

String Comparison Methods

- **`s.equals(t)`** – true if s and t are the same string
- **`s.equalsIgnoreCase(t)`** – true if s and t differ in case only
- **`s.compareTo(t)`**
 - is zero (0) if the s and t are the same
 - is positive if s comes after t
 - is negative if s comes before t.

Collating Sequence

- The order in which characters are assumed to appear is called the collating sequence.
- For now, we are most concerned with the following facts about the collating sequence:
 - Digits (0-9) come before letters.
 - All 26 upper case letters come before the lower case letters.
 - Within upper case and within lower case, the letters all fall within alphabetical order.

CompareStrings.java

```
import java.util.Scanner;

public class CompareStrings {
    public static void main(String[] args) {
        String s = new String("First");
        String t = new String("first");
        String u = new String("Second");

        if (s.equals(t))
            System.out.println("'" + s + "' and '"
                + t + "' are the same.");
        else
            System.out.println("'" + s + "' and '"
                + t + "' are different.");
    }
}
```

```

if (s.equalsIgnoreCase(t))
    System.out.println("\'" + s + "\' and \''
        + t + "\' are almost the same.");
else
    System.out.println("\'" + s + "\' and \''
        + t + "\' are very different.");

if (s.compareTo(u) == 0)
    System.out.println("\'" + s + "\' and \''
        + u + "\' are the same.");
else if (s.compareTo(u) > 1)
    System.out.println("\'" + s
        + "\' comes after \''
        + u + "\'.");
else
    System.out.println("\'" + s
        + "\' comes before \''
        + u + "\'.");

```

```

if (s.compareTo(t) == 0)
    System.out.println("\'" + s + "\' and \''
        + t + "\' are the same.");
else if (s.compareTo(t) > 1)
    System.out.println("\'" + s + "\' comes
after \''
        + t + "\'.");
else
    System.out.println("\'" + s + "\' comes
before \''
        + t + "\'.");

}

}


```

Example: Writing Changing a Form Letter

- Let's write a program to read a file and change every occurrence of the name "John" to "Robert"
- Initial algorithm:
 1. Instruct the user
 2. Change every occurrence on each line of "John" to "Robert"

Refining the Form Letter Algorithm

1. Instruct the user
2. Change every occurrence on each line of "John" to "Robert"

- 2.1 Get the first line
 - 2.2 As long as it isn't "The End", replace every occurrence of John with Robert
- 

Refining the Form Letter Algorithm

1. Instruct the user
- 2.1 Get the first line
- 2.2 As long as it isn't "The End", replace every occurrence of John with Robert

- 2.2 WHILE the line \neq "The End"
 - 2.2.1 Replace each occurrence of "John" with Robert
 - 2.2.2 Print the new line
 - 2.2.3 Get the next line

Refining the Form Letter Algorithm

1. Instruct the user
- 2.1 Get the first line
- 2.2 WHILE the line \neq "The End"
 - 2.2.1 Replace each occurrence of "John" with Robert
 - 2.2.2 Print the new line
 - 2.2.3 Get the next line

- 2.2.1.1 Create a substring of everything up to the first occurrence of "John" WHILE the line \neq "The End"
- 2.2.1.2 Concatenate "Robert" at the end of this substring
- 2.2.1.3 Concatenate the input string after John to this Substring
- 2.2.1.4 This replaces the input string

1. Instruct the user
- 2.1 Get the first line
- 2.2 WHILE the line \neq "The End"
 - 2.2.1.1 Create a substring of everything up to the first occurrence of "John" WHILE the line \neq "The End"
 - 2.2.1.2 Concatenate "Robert" at the end of this substring
 - 2.2.1.3 Concatenate the input string after John to this Substring
 - 2.2.1.4 This replaces the input string
- 2.2.2 Print the new line
- 2.2.3 Get the next line

```
System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
```

```
System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
```

- 2.2 WHILE the line \neq "The End"
 - 2.2.1.1 Create a substring of everything up to the first occurrence of "John" WHILE the line \neq "The End"
 - 2.2.1.2 Concatenate "Robert" at the end of this substring
 - 2.2.1.3 Concatenate the input string after John to this Substring
 - 2.2.1.4 This replaces the input string
- 2.2.2 Print the new line
- 2.2.3 Get the next line

```
while
    (!inString.equalsIgnoreCase("The End")) {
```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
2.2.1.1 Create a substring of everything up to the
2.2.1.1.1 first occurrence of "John" WHILE the line ≠ "The End"
2.2.1.2 Concatenate "Robert" at the end of this substring
2.2.1.3 Concatenate the input string after John to this
2.2.1.3.1 Substring
2.2.1.4 This replaces the input string
2.2.2 Print the new line
2.2.3 Get the next line
    }
outString
    = inString.substring(0, indexOfJohn);

```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
outString
    = inString.substring(0, indexOfJohn);
2.2.1.2 Concatenate "Robert" at the end of this substring
2.2.1.3 Concatenate the input string after John to this
2.2.1.3.1 Substring
2.2.1.4 This replaces the input string
2.2.2 Print the new line
2.2.3 Get the next line
    }
outString = outString.concat("Robert");

```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
outString
    = inString.substring(0,indexOfJohn);
outString = outString.concat("Robert");
2.2.1.3 Concatenate the input string after John to this
Substring
2.2.1.4 This replaces the input string
2.2.2 Print the new line
2.2.3 Get the next line
}

```

```

outString = outString.concat (
    inString.substring(indexOfJohn+4));

```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\');
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
outString
    = inString.substring(0,indexOfJohn);
outString = outString.concat("Robert");
outString = outString.concat (
    inString.substring(indexOfJohn+4));
2.2.1.4 This replaces the input string
2.2.2 Print the new line
2.2.3 Get the next line
}

```

```

inString = outString;

```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\'");
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
    outString
        = inString.substring(0,indexOfJohn);
    outString = outString.concat("Robert");
    outString = outString.concat(
        inString.substring(indexOfJohn+4));
    inString = outString;
2.2.2 Print the new line
2.2.3 Get the next line
}

```

```
System.out.println(outString);
```

```

System.out.println("Please begin typing. "
    + "      End by typing \'The End\'");
inString = keyb.nextLine();
while
    (!inString.equalsIgnoreCase("The End")) {
    outString
        = inString.substring(0,indexOfJohn);
    outString = outString.concat("Robert");
    outString = outString.concat(
        inString.substring(indexOfJohn+4));
    inString = outString;
    System.out.println(outString);
2.2.3 Get the next line
}

```

```
inString = keyb.nextLine();
```

Example: ChangeLetter.java

```
import java.util.Scanner;

public class ChangeLetter {
    // Change every occurrence of "John" in the
    // text of a form letter to "Robert"
    public static void main(String[] args) {
        Scanner keyb = new Scanner(System.in);

        // There are two strings - the input string
        // inString and the output string outString
        String inString = new String(),
            outString = new String();
```

```
        // indexOfJohn is the position within the
        // string where the next occurrence of John
        // begins.
        int indexOfJohn;

        // Prompt the user and instruct him/her how
        // to indicate the end of the letter
        System.out.println("Please begin typing. "
            + "    End by typing \'The End\'");
        inString = keyb.nextLine();

        // Keep changing as long as (s)he didn't
        // type "the end"
        while
            (!inString.equalsIgnoreCase("The End")) {
            // Find the occurrence of John
            indexOfJohn = inString.indexOf("John");
```

```

// As long as there are more occurrences of
// John, replace it with Robert
while (indexOfJohn != -1){
    // create a string with everything up to
    // "John"
    outString
        = inString.substring(0,indexOfJohn);

    // Add "Robert" at the end of the
    // substring
    outString = outString.concat("Robert");

    // Concatenate everything in the input
    // string after the next occurrence of
    // "John"
    outString = outString.concat(
        inString.substring(indexOfJohn+4));
}

```

```

// This replaces the input string - get
// the new value for indexOfJohn
inString = outString;
indexOfJohn = inString.indexOf("John");
}
// Output the new line of text
System.out.println(outString);

// Get the next line
inString = keyb.nextLine();
}
}
}

```