# Handout 3

## Strings and String Class.
## Input/Output with JOptionPane.

**Strings** In Java strings are represented with a class type String.

**Examples:**

1. objects of class string:
   - `"This is a short string"`
   - `"a" // different from 'a'`
   - `""  // an empty string`
   - `"123.5"`
   - `"      "`

2. Declaring a variable of class String, assigning values to string variables, Concatenation operator (+).

```
String  name, lastName;
name  =  "John";
lastName  =  "Lennon";

String  theBeatle = name + "  "  + lastName;
```

3. Primitive type values that are concatenated with a string are converted into strings automatically, so

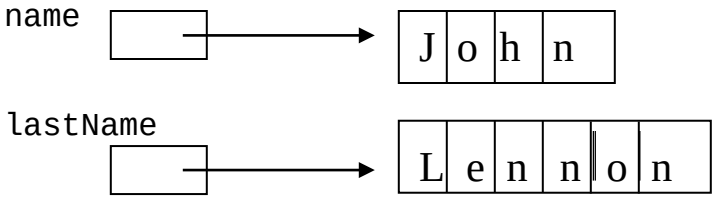
```
String name  =  "John" + 3.2;
```

Stores `"John3.2"` in `name`

What does the following segment print?

```
String resOne = "5 and 3 " +  5 + 3;
String resTwo  = "5 and 3 " +  (5 + 3);
System.out.println (resOne);
System.out.println (resTwo);
```

Variables of class type store **references** to the object. String is a class type, so variables of type String store a reference to the actual string object, depicted below.
For Example 1 from above.




Characters within the string are indexed (i.e. numbered) **starting with 0.** The index of a character within a string is its position (counting from 0).

The length of a string is equal to the number of characters in it.

String **methods** (see charts in Chapter 2 and Chapter 9, p. 566 - 579) include methods for:

- finding out the length (i.e. number of characters in a string) - `length()`
- extracting a character at a certain position – `charAt(position)`
- replacing a character with another – `replace (oldchar, newchar)`
- extracting a substring at given position – `substring(start), substring(Start, End)`
- checking if a string contains another as a substring: `contains (aStr)`
- searching for a substring: `contains (aStr), indexOf (aStr), indexOf(aStr, Start), lastIndexOf (aStr), lastIndexOf(aStr, Start)`
- changing letter case: `toUpperCase(), toLowerCase()`
- string comparison – `equals (anotherStr), compareTo(anotherString)`

Most of these methods require *parameters* to be passed, and *return* a certain value.

```
/* Demo program on String methods: */
public class StringMethodsDemo {
  public static void main (String[] args) {
        String phone = "  781-345-4758  ";

        // remove spaces before and after
        phone = phone.trim();
        System.out.println("\""+ phone + "\"");

        // Searching methods:
        System.out.println(" contains a dash? " + phone.contains("-"));
        System.out.println(" contains \"foo\"? " + phone.contains("foo"));

       // Find the position of the first dash
        int posFirstDash = phone.indexOf("-");
        int posLastDash =  phone.lastIndexOf("-"); // last dash
        System.out.println("Dashes at positions "+
                                posFirstDash + " and " + posLastDash);
        int posDashAfterFirst = phone.indexOf("-", posFirstDash+1);

        System.out.println("Second dash at position "+  posDashAfterFirst);

        // Find the position of "475"
        System.out.println("475 is at position " + phone.indexOf("475"));
        System.out.println("*** is at position " + phone.indexOf("***"));
                                                                  // prints -1
        // Extract the part between the two dashes
        String inBetweenDashes = phone.substring(posFirstDash+1, posDashAfterFirst);
        System.out.println(inBetweenDashes);

        // Compare two strings alphabetically
        String one = "apple", two = "arithmetic", three = "apple";
        System.out.println(one.compareToIgnoreCase(two));
        System.out.println(two.compareToIgnoreCase(one));

        // Compare for equality
        System.out.println(one.equals(two));
        System.out.println(one.equals(three));  }}
```

---

**Example: <u>Notice the syntax of invoking a String method (see examples from class)</u>**

**Note**: When comparing the content of two strings for equality use the equals method! (Do not use == ). The `equals`() method returns a boolean value, `true` or `false`.

**Example:**

```
String strOne, strTwo;
Scanner kbrd = new Scanner(System.in);
System.out.println("Please enter two strings.");
strOne = kbrd.nextLine();
strTwo = kbrd.nextLine();

System.out.println(strOne + " is the same as "
                   + strTwo));
System.out.println("is " + strOne.equals(strTwo));
```

**Practice problems:**

- A course ID is a string that has the following format: it starts with two letters followed by the dash ('-') and any number of digits, e.g. CS-230, MA-1220, CH-12. Write a program that reads in a course ID from the user and prints out the number part of the ID separately.

- How would you extract the letter and the number parts of a course ID that may have any number of letters, followed by a single dash and any number of digits, e.g. COMP-230, LIT-31, MATH-645 ?

---

**String methods practice:** in one java main method include the code to do the following (in parentheses I've included the String methods that would be helpful for each task. It may be possible to accomplish each task using other methods):

1. Given a string that contains a phone number, except without dashes, e.g. "7813456789", compose another string that contains the phone with the dashes: "781-345-6789". `(+, substring)`
2. Given a string, compose another one, which contains the first character and the last character, in uppercase. For example, for string "Bentley", the resulting string should be "BY". (`substring, charAt, length, toUpperCase, +`),
3. Given a string of even length, produce the second half. So the string "WooHoo" yields "Hoo". (`substring, length`)
4. Given a string containing a sentence with parentheses, print out the test inside parentheses, removing all surrounding spaces; for example, given "There was snow ( a lot of it! ) last week." Output "a lot of it!" `(indexOf, substring, trim)`
5. Given a string of text, replace all space characters with dashes, e.g. for "There was snow" the output should be "There-was-snow". (`replace`)
6. Given a word that contains letter a, e.g. "blackboard" produce one that has two letters after the first 'a' capitalized, i.e. "blaCKboard" `(indexOf, substring, toUpperCase).`
7. Given a string of text with several words, e.g. "brevity is the soul of wit" change it so that it has
    a. the first word capitalized, i.e. "BREVITY is the soul of wit",
    b. the last word capitalized, i.e. "brevity is the soul of WIT"
    c. the first two words capitalized "BREVITY IS the soul of wit".

**Input/Output using JOptionPane**

**JOptionPane class**

- provides methods for input/output via a pop-up window. Methods:
  - showInputDialog
  - showMessageDialog
- part of the Swing package – predefined collection of classes that comes with the Java distribution
- the package must be explicitly imported using the **`import`** statement that comes before the class definition:

```
import javax.swing.JOptionPane; // imports class JOptionPane
```

Alternatively, to import all classes from a package:

```
import javax.swing.*; //imports all classes in the swing pckg.
```

For both **`showInputDialog`** and **`showMessageDialog`** methods

- Note the calling convention - must precede name of method with its class name
  - **`JOptionPane.showInputDialog("prompt for input here");`**
  - **`JOptionPane.showMessageDialog(null,"Hello World!");`**

- Window becomes invisible after user clicks "ok" button. The program must be terminated using the following statement (last statement of the main method);
  - **`System.exit(0);`**`// ends program and releases resources`
    - The 0 indicates normal termination

**Example:**

```
import javax.swing.JOptionPane;
// or import javax.swing.*;

/*   Get user input as a String.  */

public class JOptionPaneSimple {

  public static void main(String [] args) {

      // get user input as a string
      String strIn = JOptionPane.showInputDialog("Enter a string");
      System.out.println("The user entered: " + strIn);


      JOptionPane.showMessageDialog(null, "The user entered: "
                                  + strIn);

      System.exit(0); //must be the last statement of main if
                      // JOptionPane methods are used
  }
}
```

The next example demonstrates how to convert the entered string into a number type.

```
import javax.swing.JOptionPane;

public class JOptionPaneDemo {

   public static void main(String[] args)  {
      String podString =
         JOptionPane.showInputDialog("Enter number of pods:");

      // convert string stored in podString into an integer number
      int numberOfPods = Integer.parseInt(podString);

      // if user input was not a whole number, a RUN-TIME error
      // will occur while executing the above line.

      //This can also be done as follows (combine two method calls)

      int peasPerPod = Integer.parseInt(JOptionPane.showInputDialog(
                               "Enter number of peas in a pod:"));

      int totalNumberOfPeas = numberOfPods*peasPerPod;

      JOptionPane.showMessageDialog(null, "The total number of peas = "
                                     + totalNumberOfPeas);
      System.exit(0);
   }
}
```

**Summary:**

### JOptionPane .`showInputDialog()` method

- Displays a **popup window** for accepting user input
- Takes one parameter – the string to be displayed in the pop-up window
- Returns user input as a **`String`**
- To convert input to an integer, or double, etc.. must call the converter methods
    - Integer.parseInt()
    - Double.parseDouble()

### JOptionPane .`showMessageDialog()` method

- Displays a **popup window** for displaying messages, such as output
- Takes two parameters
- Precede name of method with its class name if using it within a different class
    - **`JOptionPane.showMessageDialog(null,"put output here");`**
        - The `null` value tells Java to create a default window for showing the dialog.

- Also requires **`System.exit(0);`** for proper termination.