Unit 2: Using Objects
Strings

Adapted from:
1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp
2) Runestone CSAwesome Curriculum

https://longbaonguyen.github.io
Strings

• **string**: An object storing a sequence of text characters.
  – String is not a primitive type. String is an object type.
  – Three ways to initialize a string:
    1. `String a = new String("text");`
    2. `String b = "text";`
    3. `String c = expression;`

  – Examples:
    ```java
    String a = new String("John Smith");
    String b = "John Smith";
    String c = "John" + "Smith";
    ```

The String class is part of the java.lang package. Classes in the java.lang package are available by default.
Indexes

- Characters of a string are numbered with 0-based *indexes*:

```java
String a = "J. Smith";
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>J</td>
<td>.</td>
<td>S</td>
<td>m</td>
<td>i</td>
<td>t</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

- First character's index : 0
- Last character's index : 1 less than the string's length
**String concatenation**: primitive values can be concatenated with a String object using +. This causes implicit conversion of the values to String objects.

```
"hello" + 42      is "hello42"
1 + "abc" + 2     is "1abc2"
"abc" + 1 + 2     is "abc12"
1 + 2 + "abc"     is "3abc"
"abc" + 9 * 3     is "abc27"
"1" + 1           is "11"
4 - 1 + "abc"     is "3abc"
```

Use + to print a string and an expression's value together.

```
- System.out.println("Grade: " + (95.1 + 71.9) / 2);
```

- **Output**: Grade: 83.5
**Escape sequences**

**escape sequence**: A special sequence of characters used to represent certain special characters in a string.

- `\n` new line character
- `\"` quotation mark character
- `\\` backslash character

- **Example:**
  ```java
  System.out.println("\\hello\\nhow are "\"you\"?\\\\");
  ```

- **Output:**
  ```
  \hello
  how are "you"?
  ```
Questions

• What is the output of the following `println` statements?

```java
System.out.println("\\\\");
System.out.println("\\");
System.out.println("\\\\\\\\");
```

• Write a `println` statement to produce this output:

```
/ \ // \ \ // // // \ \ \ 
```
• Output of each `println` statement:

```java
\`
.'
"""
```

• `println` statement to produce the line of output:

```java
System.out.println("/ \ // \\\\
/// /// \\\\\");
```
<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String(String str)</td>
<td>Constructs a new String object that represents the same sequence of characters as str</td>
</tr>
<tr>
<td>int length()</td>
<td>Returns number of characters in this string</td>
</tr>
<tr>
<td>substring(index1, index2) or</td>
<td>Returns the characters in this string from ( index1 ) (inclusive) to ( index2 ) (exclusive); if ( index2 ) is omitted, grabs till end of string</td>
</tr>
<tr>
<td>substring(index1)</td>
<td></td>
</tr>
<tr>
<td>boolean equals(String other)</td>
<td>Returns true if this is equal to other; returns false otherwise</td>
</tr>
<tr>
<td>int compareTo(String other)</td>
<td>Returns a value &lt; 0 if this is less than other; returns zero if this is equal to other; returns a value &gt; 0 if this is greater than other</td>
</tr>
<tr>
<td>indexOf(str)</td>
<td>Returns index where the start of the given string appears in this string (-1 if not found)</td>
</tr>
</tbody>
</table>
String method examples

// index 0123456789012345678
String s1 = "programming in java";

System.out.println(s1.length()); // 19
System.out.println(s1.indexOf("i")); // 8
System.out.println(s1.indexOf("gram")); // 3
System.out.println(s1.indexOf("hi")); // -1

System.out.println(s1.substring(7, 10)); // "min"
System.out.println(s1.substring(12)); // "in java"
System.out.println(s1.substring(2,3)); // "o"
System.out.println(s1.substring(2,2)); // ", empty string

String s2 = s1.substring(10, 17); // "g in ja"
Given the following string:

```java
// index 0123456789012345678901
String book = "Building Java Programs";
```

How would you extract the word "Java"?

```java
String word = book.substring(9,13);
```

**String’s equals:**

```java
String a = “hello”, b = “Hello”;
System.out.println(a.equals(b)); // false
System.out.println(a.equals(“hello”)); // true
```
Comparing strings

When the operator == is used with object variables it returns true when the two variables refer to the same object. These variables are called aliases for the same object and object references.

With strings this happens when one string variable is set to another or when strings are set to the same string literal.

```java
String a = "hi"; //String literal
String b = "hi";
System.out.println(a == b); /* true
the Java run-time will check if that string literal already exists as an object in memory, and if so reuse it. So a and b will refer to the same string object. */
String c = b;
System.out.println(b == c); //true
```
Comparing strings

With String objects, you must use the equals method to test if two strings have the same characters in the same order instead of `==` which is used for primitive types.

If you use the `new` keyword to create a string it will create a new string object. So, even if we create two string objects with the same characters using the new operator they will not refer to the same object.

```java
String a = new String("hi");
String b = new String("hi");
System.out.println(a == b);  //false, not same objects
System.out.println(a.equals(b));  //true, same characters & order
```
The compareTo method compares strings in dictionary (lexicographical) order:

- If `string1.compareTo(string2) < 0`, then `string1` precedes `string2` in the dictionary.
- If `string1.compareTo(string2) > 0`, then `string1` follows `string2` in the dictionary.
- If `string1.compareTo(string2) == 0`, then `string1` and `string2` are identical. (This test is an alternative to `string1.equals(string2)`.)

All you need to know is that all digits precede all capital letters, which precede all lowercase letters. Thus "5" comes before "R", which comes before "a".
String s1 = "HOT", s2 = "HOTEL", s3 = "dog";
String s4 = “hot”;
String s5 = new String("hot");

s1.compareTo(s2) returns a negative number(<0) since s1 < s2.

s4.compareTo(s3) returns a positive number(>0) since s4 > s3.

s4.compareTo(s5) returns 0 since s4 is equal to s5.
Methods like `substring` build and return a new string, rather than modifying the current string.

- String is **immutable**; once created, its value cannot be changed.

```java
String s = "kendrick";
s = "snoop dog";
// "kendrick" is discarded and a new String
// object "snoop dog" is created.

s.substring(0, 5); // returns snoop, not stored
System.out.println(s);
// snoop dog, s is not changed
```
To modify a variable's value, you must reassign it:

```java
String s = "lil bow wow";
s = s.substring(0, 3);
System.out.println(s);  // lil
```
**Value semantics**: String is the only object class that follows value semantics. Modifying the parameter will not affect the variable passed in.

```java
public class MyClass{
    public static void main(String[] args){
        String x = "hi";
        changeMe(x);
        System.out.println(x); // hi
    }
    public static void changeMe(String x){
        x = "hello";
    }
}

Note: The value of x in main did not change.
```
String beans;
String cheese;
String quartet;
String bikini;
String theory;

String school = "Hogwarts: School for Pigs";
String headmaster = "Kevin Bacon";
String teacher = "Brad Pig";
String gf = "Piggy Azalea";

String fav_book = "Hamlet";
String fav_char_star_wars = "Ham Solo";
String fav_char_star_wars_2 = "ChewBacon";
Login to repl.it classroom, do the 5 "String Methods" problems.
Lab 2

Create a new repl on repl.it.

Implement the driver class(Main.java) to include two methods: printName and pigLatin.

**printName**: static, void method which takes a String input which represents a full name in the format: first name, a space, middle name, space and last name. The method extracts first, middle and last name and prints them. Use Scanner class to take input. (String full_name = console.nextLine();).

*input: "Michael Benjamin Smith"

*print output:
First: Michael
Middle: Benjamin
Last: Smith
pigLatin: static, method which takes a String input and returns the string in Pig Latin: take the first letter and put it at the end of the word and add the letters “ay” to the end. For example:

input: "pig"

returns: "igpay"
References

1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp

2) Runestone CSAwesome Curriculum: https://runestone.academy/runestone/books/published/csawesome/index.html

For more tutorials/lecture notes in Java, Python, game programming, artificial intelligence with neural networks:

https://longbaonguyen.github.io