Intro to Computer Science
CSCI-UA-0101

String's built-in methods
Using methods to manipulate Strings

Readings:
Chapter 6 from the liang book and chapter 4 in "**Think Java, Second Edition**"

Prof. Sana Odeh
sana@nyu.edu

Midterm exam 10/27 during class

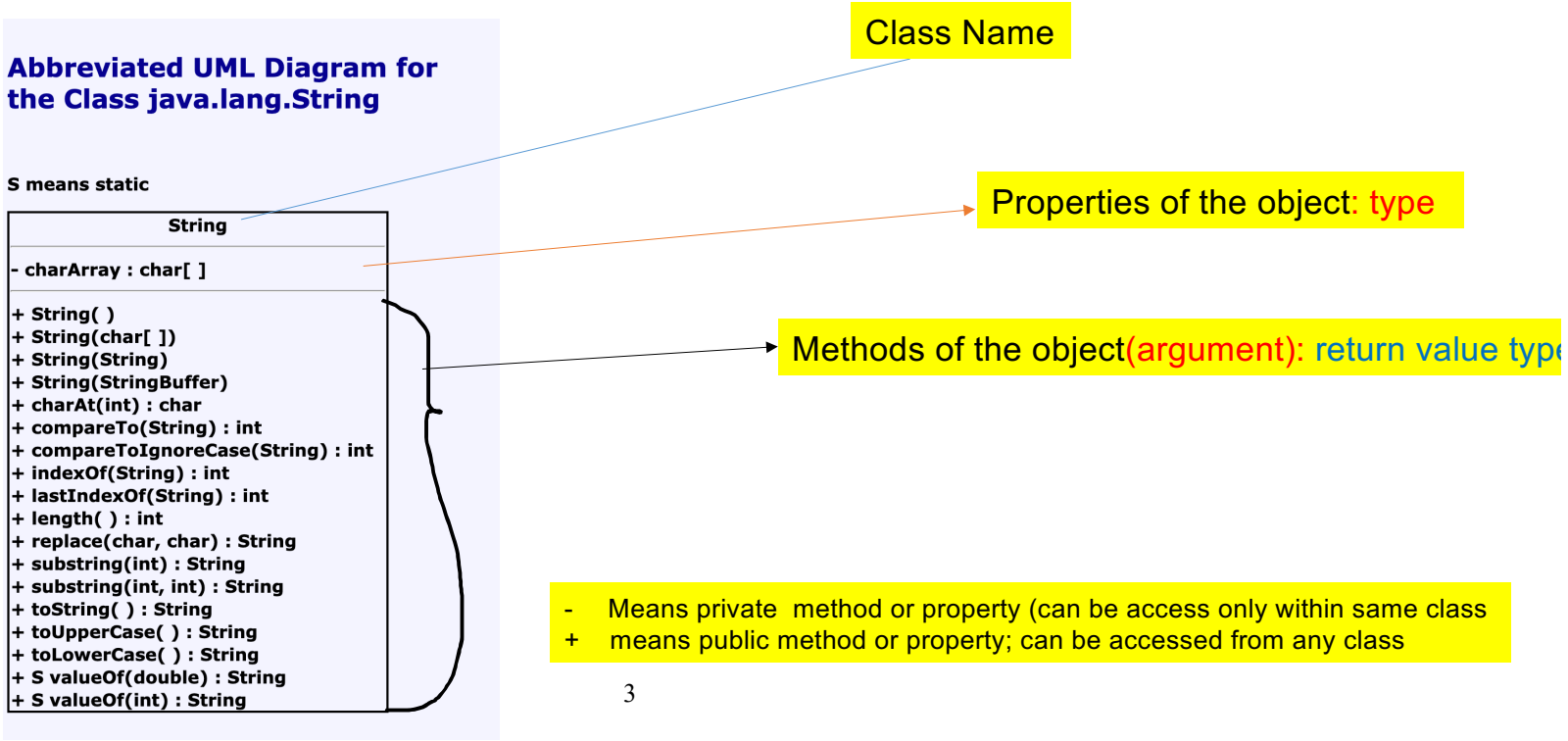
Not online, no books or computers. You will write into a booklet. You with get no help from no one or any resource.

- I'll post a sample midterm exam this Thursday so you can practice from. Try to work in groups to set study together
- I will do a review next Tuesday
- Make sure to do all the readings (books and and focus on lectures material)
- Topics include every thing we cover this week
- Practice all of the examples from class (focus on class examples and homework) and try to also do examples from both books
- **Format:**
    - **True/false questions**
    - **Multiple choice**
    - **1 complete program**
    - **One method**
- You will Not get help from books, web, notes, tools, resource, or anyone else on earth – you will fail the course if you cheat. You with get no help from no one or any resource.

Good luck

Uniform Modeling Language (UML) representing String Class Methods
– will study more in the next few weeks
is a universal visual chart representing a class/blue print for the object data type
This UML represents the String class properties and methods for each class
This is always helpful to find what are the properties and methods for each class

UML is a diagram that has three rows; each row represents specific info about the class; class name, class properties and class methods



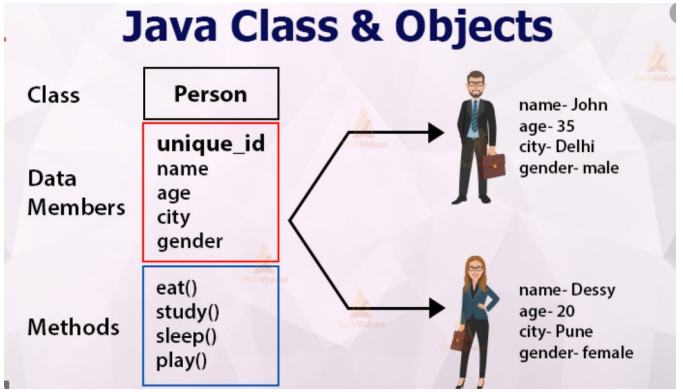
String data type

- String is an object reference Data type
  String is a class/blueprint
  which defines properties and methods (behaviors) for that object
  Every string created/ instantiated from that class is an object which inherits properties and methods from that class

Classes and objects (more on this in the next few weeks)

class: A program entity that represents either:

1. A program / module, or
2. A type of objects.



- A class is a blueprint or template for constructing objects.

- Example: The Person class (type) is a template for creating many `person(s)` objects (windows).
  - Java has 1000s of classes. Later (Ch.8) we will write our own.

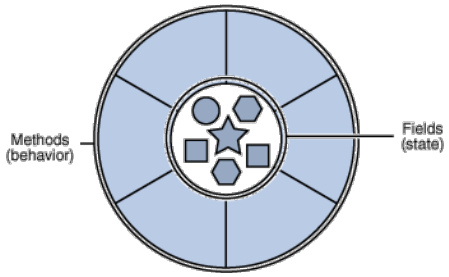
object: An entity that combines data and behavior.

- **object-oriented programming (OOP)**: Programs that perform their behavior as interactions between objects.

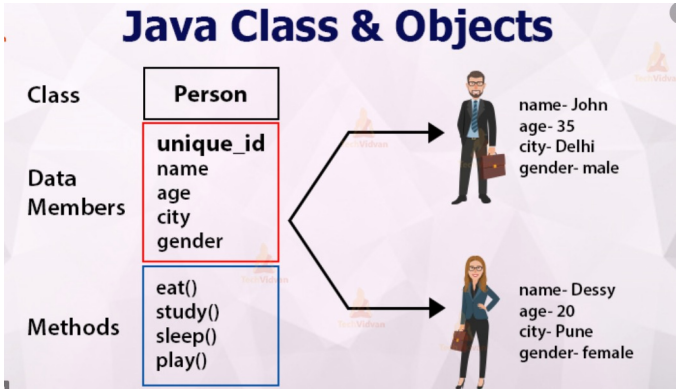
Objects

object: An entity that contains data and behavior.

- *data*: variables inside the object
- *behavior*: methods inside the object



- You interact with the methods;
  the data is hidden in the object.



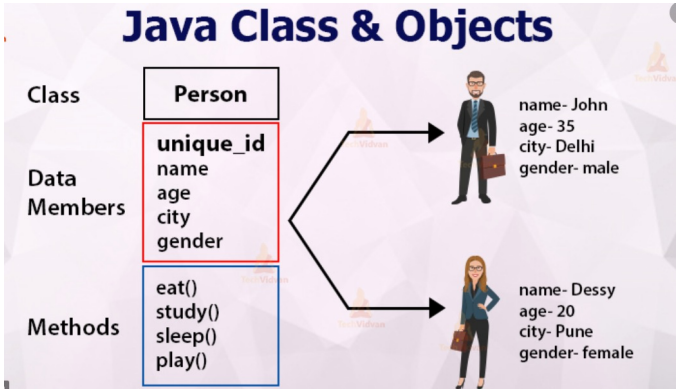
Constructing (creating) an object:

```
Type objectName = new Type(parameters);
```

Calling an object's method:

```
objectName.methodName(parameters);
```

The String is an Object Data Type
Class is a blueprint that defines object data (properties) and behavior(methods) when you create/instantiate the object from a class



String is actually a predefined class in the Java library just like the System class and Scanner class.

The String type is not a primitive type. It is known as a *reference type*.

Any Java class can be used as a reference type for a variable.

Reference data types will be thoroughly discussed in Chapter 9, **"Objects and Classes."**

For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, how to concatenate strings, and to perform simple operations for strings.

Advance String instantiation of declaration of an object- we will learn more about this notation when we study OOP after a few weeks:

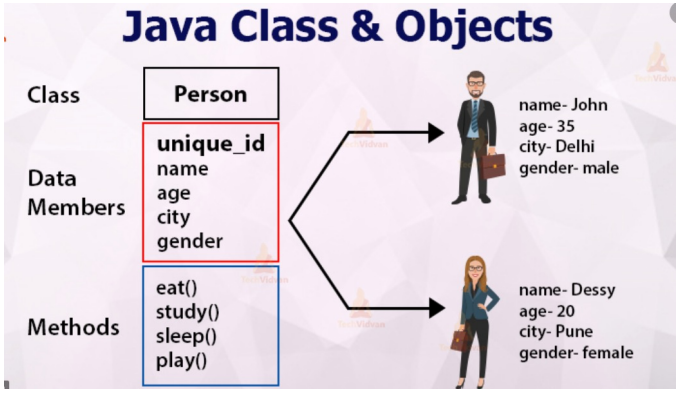
```
String message = new String(); // instantiating or creating an object from a class
```

The String Data Type (an object reference)

An object reference means that the class data type (blue print) defines the data (properties) and behavior (methods) for the objects created/instantiated from that class.

We will spend a lot of time in the 2nd part of the semester to discuss OOP classes and objects.

For now, just remember that the String class has defined many important methods to define strings and its objects.



The String Data Type
String is a class which instantiates/creates an object reference Data Type

The char type only represents one character and also it's a **primitive type** or a literal – just one value in a memory location.

A String data type is an object reference and can store zero or as many characters from the keyboard. To represent a string of characters, use the data type called String. For example,

String message = "Welcome to Java";

String is actually a predefined class in the Java library just like the System class and Scanner class.

The String type is not a primitive type. It is known as a *reference type (object)*.

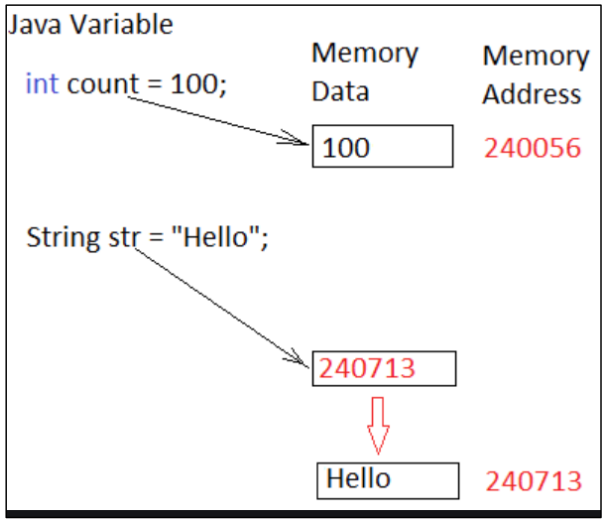
Any Java class can be used as a reference type for a variable.

Reference data types will be thoroughly discussed in Chapter 9, "Objects and Classes."

For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, how to concatenate strings, and to perform simple operations for strings.

Advance String instantiation of declaration of an object- we will learn more about this notation when we study OOP after a few weeks:

String message = new String();



String Interning or intern

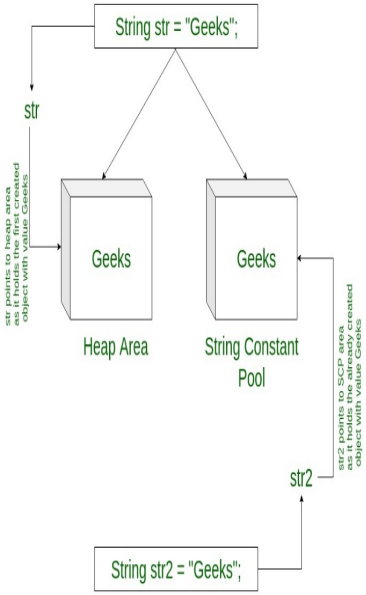
- String Interning is a method of storing only one copy of each distinct String Value, which must be immutable. By applying String.intern() on a couple of strings will ensure that all strings having the same contents share the same memory.
- For example, if a name 'Amy' appears 100 times, by interning you ensure only one 'Amy' is actually allocated memory.
- **intern() method :** In Java, when we perform any operation using intern() method, it returns a canonical representation for the string object. A pool is managed by String class.

•When the intern() method is executed then it checks whether the String equals to this String Object is in the pool or not.

•If it is available, then the string from the pool is returned. Otherwise, this String object is added to the pool and a reference to this String object is returned.

•It follows that for any two strings s and **t, s.intern() == t.intern() is true if and only if s.equals(t) is true.**

It is advised to use equals(), not ==, to compare two strings. This is because == operator compares memory locations, while equals() method compares the content stored in two objects.



Example of Comparing memory location and content with a string

```
// Java program to illustrate

// intern() method

public class GFG {

public static void main(String[] args)  {

        // S1 refers to Object in the Heap Area

        String s1 = new String("GFG");

        // S2 refers to Object in SCP Area

        String s2 = s1.intern(); // Line-2

        // Comparing memory locations

               // s2 is in SCP

        System.out.println(s1 == s2);

        // Comparing only values

        System.out.println(s1.equals(s2));
```

```
// S3 refers to Object in the SCP Area
        String s3 = "GFG"; // Line-3

        System.out.println(s2 == s3);
    }
}
```

Strings input

```
Scanner input = new Scanner(System.in);

System.out.print("Enter three words separated by spaces: ");

String s1 = input.next();

String s2 = input.next();

String s3 = input.next();

System.out.println("s1 is " + s1);

System.out.println("s2 is " + s2);

System.out.println("s3 is " + s3);
```

String Concatenation

String s3 = s1.concat(s2); or String s3 = s1 + s2;

```
// Three strings are concatenated
String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with character B
String s1 = "Supplement" + 'B'; // s1 becomes SupplementB
```

String defined methods
Class String defined many methods to manipulate string

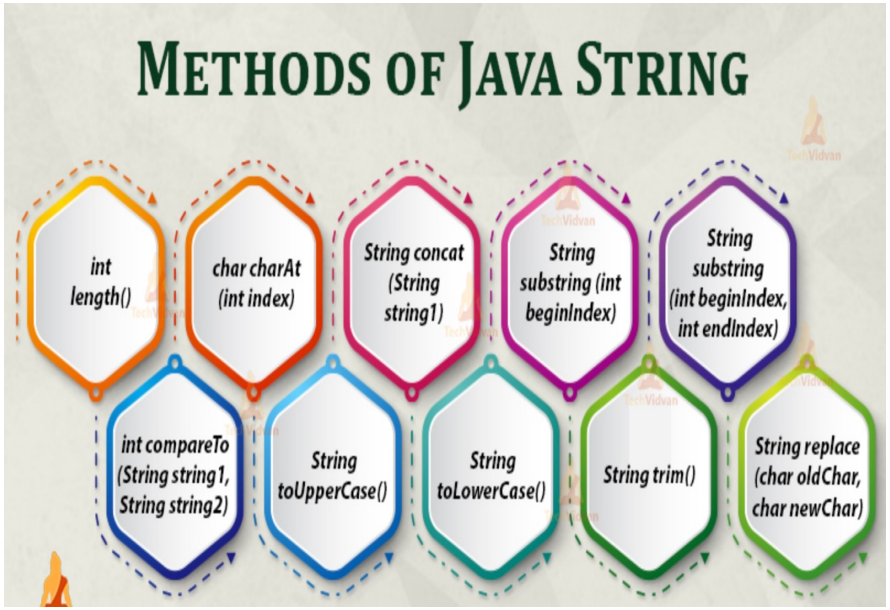
a method is a special function that is defined with respect to a particular object.

The syntax is

```
<object>.<method>(<parameters>)
```

```
>>> dna = "ACGT";
```

```
>>> dna.replace("T", "C");
```



Simple Methods for **String** Objects

Strings are objects in Java.

The methods in the preceding table can only be invoked from a specific string instance. For this reason, these methods are called *instance methods*.

referenceVariable.methodName(arguments).

Simple Methods for **String** Objects

Method	Description
`length()`	Returns the number of characters in this string.
`charAt(index)`	Returns the character at the specified index from this string.
`concat(s1)`	Returns a new string that concatenates this string with string s1.
`toUpperCase()`	Returns a new string with all letters in uppercase.
`toLowerCase()`	Returns a new string with all letters in lowercase.
`trim()`	Returns a new string with whitespace characters trimmed on both sides.

split()	**splits string on a specific delimited character such as space or , or : it returns an array (similar to a list)**
replace()	**replaces one character with another**

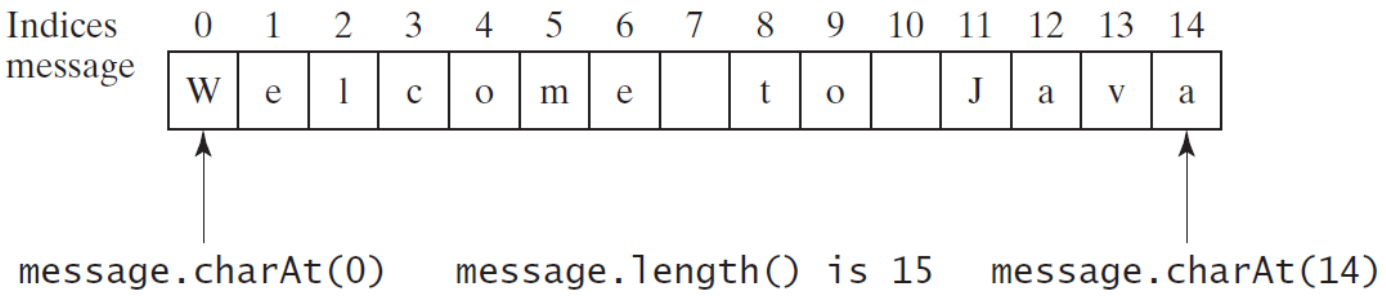
Methods for String class (String Data type) that manipulates a one character in a String (String data type)

Method	Description
`isDigit(ch)`	Returns true if the specified character is a digit.
`isLetter(ch)`	Returns true if the specified character is a letter.
`isLetterOfDigit(ch)`	Returns true if the specified character is a letter or digit.
`isLowerCase(ch)`	Returns true if the specified character is a lowercase letter.
`isUpperCase(ch)`	Returns true if the specified character is an uppercase letter.
`toLowerCase(ch)`	Returns the lowercase of the specified character.
`toUpperCase(ch)`	Returns the uppercase of the specified character.

Getting String Length

```
String message = "Java";
System.out.println("The length of " + message + " is "
  + message.length());
```

Getting Characters from a String
charAt() method



String message = **"Welcome to Java"**;

System.out.println(**"The first character in message is "**
  + message.charAt(0));

Reading a string from the Console
Then assigning one character from the string to a char

```
Scanner input = new Scanner(System.in);

System.out.print("Enter a sentence: ");

String s = input.nextLine();

char ch = s.charAt(0);

System.out.println("The first character of the sentence that you entered is " + ch);
```

Extracting one character (Type char) from a string using `charAT(index)` method which is on of the String's methods

You should use **`charAt(index)`** to parse the character from user input (String)

Index is the integer representing the position of the character in the string or its index

For example:

```
char c;

String line;

 System.out.println("Enter a string.");

 String line = in.nextLine();

 System.out.println("Your string is " + line);

 letter = cAsString.charAt(0);
```

Will grab the first character from the user's input.

Extracting one character (Type char) from a string using `charAT(index)` method which is on of the String's methods

```
Scanner in = new Scanner(System.in);

System.out.println("Enter a string.");

String line = in.nextLine(); // input a string from the keyboard

System.out.println("Your string is " + line);

// extract the first character from the string at index/position 0
// using charAt() method which is one of the Strings methods
// allowing you to extract one character from the String at a specific index

char letter1 = line.charAt(0); // extract the first character
System.out.println("Your character at index 0 is: " + letter1);

// extract the 3rd character from the string at index/position 3

char letter2 = line.charAt(3); // extract the 3rd character

System.out.println("Your character at index 0 is: " + letter2);

in.close();
```

Length method
returns the length of the string

```
//intro to strings

public class length_string

{

                public static void main(String[] args)

                {

                String s = “abcdef";

                int len = s.length();

                System.out.println(s.charAt(0) );

                System.out.println(len);

                System.out.println(s.charAt(len-1) );



                }

}
```

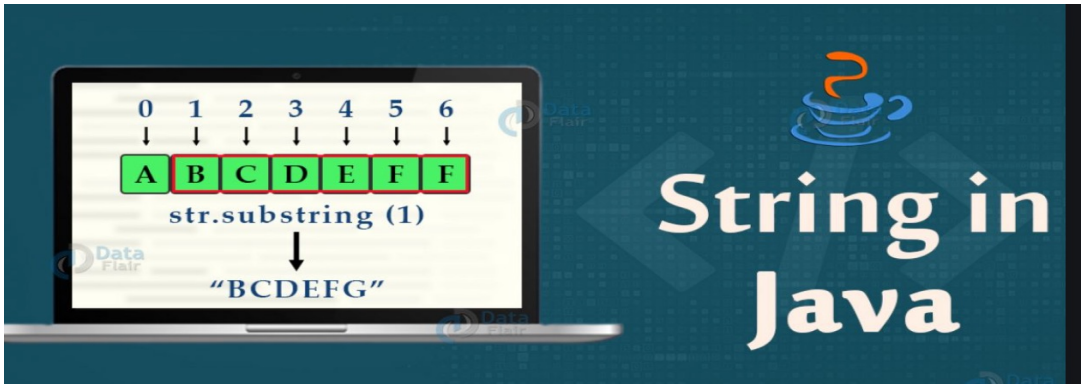
What would this print???

Method length return the **length of the string s which is 6**

Output:
a
6

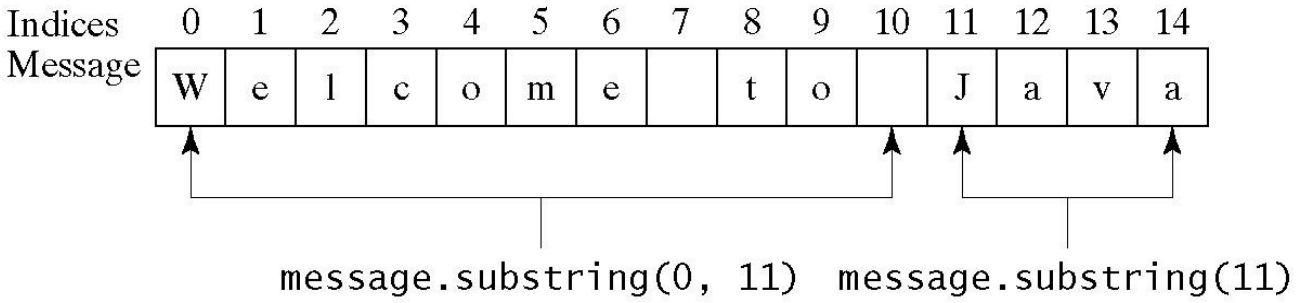
Obtaining Substrings

Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`, as shown in Figure 9.6. Note that the character at `endIndex` is not part of the substring.



Obtaining Substrings

Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`, as shown in Figure 9.6. Note that the character at `endIndex` is not part of the substring.



string.replace() method- a very important method
The replace() method searches a string for a specified character, and returns a new string where the specified character(s) are replaced.
replace – replaces one charcter (first argument with the 2nd argument)

What's the result of this program?

```
public class Main {
  public static void main(String[] args) {
      String myStr = "Hello";
      System.out.println(myStr.replace('l', 'p'));
   }
}
```

What's the result of this 2nd version?

```
public class Main {
  public static void main(String[] args) {
    String myStr = "Hello";
    myStr = myStr.replace('e', 'p');
    System.out.println(myStr);
  }
}
```

string.split() method- a very important method
It's one of the Methods for **String** Objects
split returns an array which is similar to lists in python- more on this later

```
public class StringMethods {
  public static void main(String[] args) {
    String myStr = "Hello";
    myStr = myStr.replace('l', 'p');
    System.out.println(myStr);
    String animals = "cat mouse camel";
    System.out.println(animals);

    String animal []= animals.split(" ");
    System.out.println(animal); // array is similar to list but all array has to be same type;
    // this prints the address of array stored in memory
    //using java foreach loop to – print an array element with each iteration of the loop
    for(String word : animal){
                    System.out.println(word);
          }
// using index to iterate through the array; array length here is a property and not a method
          for(int i = 0; i< animal.length; i++){
                    System.out.println(animal[i]);

} }}
```

Comparing Strings- you can't use == to test for equality between two strings
You need to use the following methods:

Method	Description
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	Returns an integer greater than `0`, equal to `0`, or less than `0` to indicate whether this string is greater than, equal to, or less than `s1`.
`compareToIgnoreCase(s1)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.

OrderTwoCities Run

The `equals` method

Comparing Strings- you can't use == to test for equality between two strings
You need to use the following methods such as equals():

- Objects are compared using a method named `equals`.

```
Scanner console = new Scanner(System.in);
System.out.print("What is your name? ");
String name = console.next();
if (name.equals("Barney")) {
    System.out.println("I love you, you love me,");
    System.out.println("We're a happy family!");
}
```

- Technically this is a method that returns a value of type `boolean`, true of equal and false if not equal.
- If you compare two strings with == this will test if they are the same object and not if the content of the string are the same. We will learn more about == when we study more OOP in the next few weeks

`String` test methods

Comparing Strings- you can NOT use == to test for equality between two strings

You need to use the following methods:

Method	Description
`equals(`**str**`)`	whether two strings contain the same characters
`equalsIgnoreCase(`**str**`)`	whether two strings contain the same characters, ignoring upper vs. lower case
`startsWith(`**str**`)`	whether one contains other's characters at start
`endsWith(`**str**`)`	whether one contains other's characters at end
`contains(`**str**`)`	whether the given string is found within this one

```
String name = in.next();

if (name.startsWith("Prof")) {
    System.out.println("When are your office hours?");
} else if (name.equalsIgnoreCase("SANA")) {
    System.out.println("Let's talk about life!");
}
```

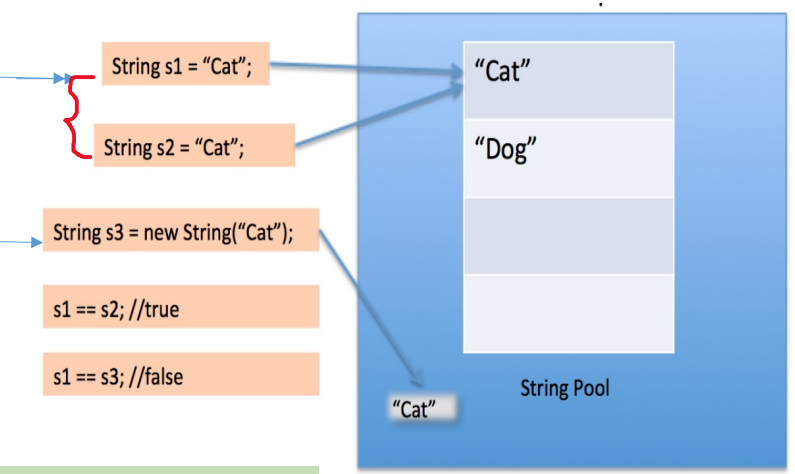
Comparing Strings
other available string methods in java
https://docs.oracle.com/javase/8/docs/api/

Java Memory called the heap

```
public class Equal1 {
  public static void main(String[] args) {

    String name1 = "cat";

    String name2 = "cat";

     String name10 = new String("cat")
     System.out.println( name1 == name2);

    System.out.println( name10 == name1 );

    System.out.println("name1.equals(name2): " + (name1.equals(name2)));

    String name5="Cat";
    System.out.println("name1.equals(name2): " + (name5.equals(name2)));
    System.out.println("name1.compareTo(name2): " + (name1.compareTo(name2)));
    System.out.println("name1.compareTo(name2): " + (name5.compareTo(name2)));

    System.out.println("name1.compareTo(name2): " + (name2.compareTo(name5)));

    String name3= "cat mouse";
    System.out.println("name3.indexOf(name2): " + (name3.indexOf(name2)));

    String  name4 = name3.substring(0, 3);

    System.out.println("name4: "+ name4) }
```

both string literals refer the same object since they have the same content. In general, you should use the string literal notation when possible. It is easier to read and it gives the compiler a chance to *optimize* your code.

String instantiate object have different address in memory even if they have same content



Output

```
true
false
name1.equals(name2): true
name5.equals(name2): false
name1.compareTo(name2): 0
name5.compareTo(name2): -32
name2.compareTo(name5): 32
name3.indexOf(name2): 0
name4: cat
```

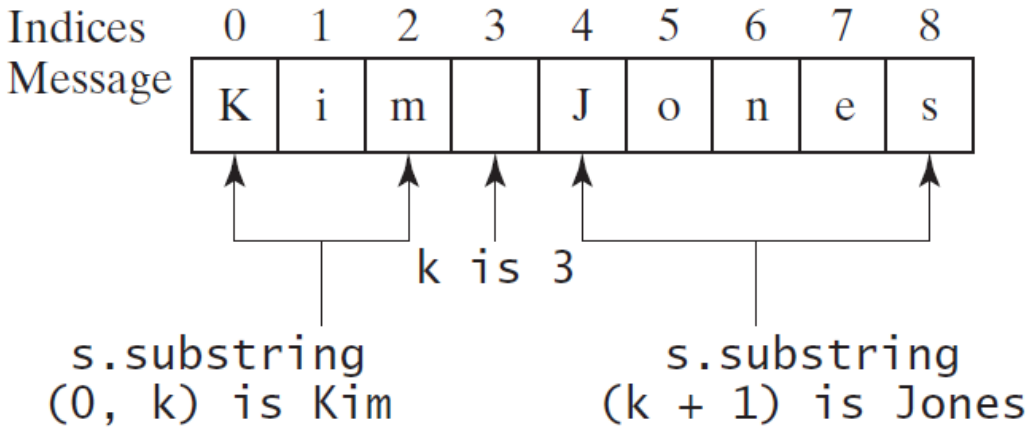
Finding a Character or a Substring in a String

(Searching for a keyword or a character in the string)

Method	Description
`indexOf(ch)`	Returns the index of the first occurrence of `ch` in the string. Returns `-1` if not matched.
`indexOf(ch, fromIndex)`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns `-1` if not matched.
`indexOf(s)`	Returns the index of the first occurrence of string `s` in this string. Returns `-1` if not matched.
`indexOf(s, fromIndex)`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns `-1` if not matched.
`lastIndexOf(ch)`	Returns the index of the last occurrence of `ch` in the string. Returns `-1` if not matched.
`lastIndexOf(ch, fromIndex)`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns `-1` if not matched.
`lastIndexOf(s)`	Returns the index of the last occurrence of string `s`. Returns `-1` if not matched.
`lastIndexOf(s, fromIndex)`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns `-1` if not matched.

Finding a Character or a Substring in a String

```
int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
```



Conversion between Strings and Numbers

```
String intString= "20.5";
int intValue = Integer.parseInt(intString); // convert from String to int
double doubleValue = Double.parseDouble(doubleString); // convert from String to double

// what would be the answer?

String s = "" + intValue + doubleValue;


// what would be the answer?

String s = intValue + " " + doubleValue;

// what would be the answer?

String s = intValue  + doubleValue;
```