in this chapter you will learn how to determine whether two objects are equal

API stands for application programming interface. In Java, an interface is something special. In the context of an API, it can be a group of class or interface definition that gives you access to a service or functionality.

* Creating and manipulating Strings:
  + These two snippets both create a string: String name = “fluffy” String name = new String (“Fluffy”);
    - Both give you a reference variable of type name pointing to the string object.
* Concatenation
  + I understand it
* Immutability:
  + Once a string object is created, it is not allowed to change. It cannot be made larger or small, and you canot change of the characters inside it.
* The String Pool
  + Since strings are everywhere in Java, they use up a lot of memory. In some production applications, they can use up 25-40 percent of the memory in the entire program. Java realizes that many strings repeat in the program and solves this issue by reusing common ones.
  + The string pool, also known as the intern pool, is a location in the JVM that collects all these strings.
  + The string pool contain literal values that appear in your program.
    - String name = “fluffy”; //efficient because it goes into the string pool
    - String name = new String (“fluffy”); //not efficient because it does not go into the string pool and it creates a new object
* Important String methods:
  + indexOf(): looks at the character in the string and finds the first index that matches the desired value.
    - indexOf can work with an individual character or a whole string as input.
    - It can also start from a requested position.
    - Returns -1 when there is no match found
  + Substring():
    - Looks for a character in a string. It returns parts of the string.
    - The first parameter is the index to start with for the returned string. As usual, this is a zero-based index
    - If starting and ending index are equal it returns an empty string
    - If starting index is less than ending index it throws an exception
    - If ending index is out of range it throws an exception
* toLowerCase() and toUpperCase()
  + String string = “animals” System.out.println(string.toLowerCase()) does not change the string object just prints it out
    - Do not forget that strings are immutable
* Equals() and equalsIgnoreCase()
  + This methods checks whether two string objects contain exactly the same characters in the same order
* startsWith() and endsWith()
  + they look at whether the provided value matches part of the String.
* Contains(String value)
  + I got it
* Replace()
  + It does a simple search and replace on the string. There is a version that takes char parameters as well as a version that takes CharSequence parameters
* Trim()
  + There is a technique called method chaining
    - String result = “animal “.trim().toLowerCase().replace(“a”,”A”);
* Using the StringBuilder class
  + This class creates a String without storing all those iterim String value.
  + StringBuilder is mutable
* Mutability and chaining
  + StringBuilder a = new StringBuilder(“abc”) StringBuilder b = a.append(“de”)
    - Both a and b are pointing to a so all changes will occur in both
* Creating String Builder
  + You can create by giving it a string or empty or a number to show how big the value will eventually be
* Important StringBuilder Methodss:
  + charAt(),indexOf(), and substring(): work same as string class
  + append(): fully understand it
  + insert(): adds characters to the StringBuilder at the requested index and returns a reference to the current StringBuilder, however it does not replace the character
  + delete() and deleteCharAt(): removes characters from the sequence and returns a reference to the current StringBuilder.
  + Reverse(): reverses the characters in the sequences and returns a reference to the current StringBuilder
  + toString(): converts a StringBuilder to a String
* StringBuilder vs StringBuffer
  + When writing new code that concatenates a lot of String Objects together, you should use StringBuilder
  + StringBuffer is older than StringBuilder, it does the same thing but it is slower since it is thread safe.
* Understanding Equality:
  + In chapter 2, you learned how to use == to compare numbers and that object references refer to the saem object.
  + StringBuilder one = newStringBuilder(); StringBuilder two = new StringBuilder(); StringBuilder three = one.append (“a”);
    - System.out.println(one == two); this is false because they do not refer to the same object
    - System.out.println(one == three); this is true because they refer to the same object
  + Never use == to compare string objects. The only time you should have to deal with == for String is on the exam
  + Authors of StringBuilder() did not implement equals(). If you call it on two StringBuilder instances, it will check reference equality
  + An object has to implement equals in order for it to evaluate, otherwise it will return false
* Understanding Java Arrays:
  + Int [] numbers1 = new int[3];
    - When using this form to instantiate an array, all elements are set to the default value for that type, which is 0.
  + Int [] numbers2 = new int[] {42,55,99};
    - Another way of initializing above array is int [] numbers2 = {42,55,99};
* Creating an array with Reference Variables: