Strings

- Testing for equality with strings.
- Lexicographic ordering of strings.
- Other string methods.
The `==` operator is also not appropriate for determining if two objects, such as strings, have the same value.

```java
String s1 = "hello";
String s2 = "hello";

if (s1 == s2)
    System.out.println("Strings are equal!");
else
    System.out.println("Strings are not equal!");
```

The above code does not compare the strings in `s1` and `s2`; why?
To test the equality of objects of class String, use method `equals`:

```java
String s1 = "hello";
String s2 = "hello";
if (s1.equals(s2))
    System.out.println("Strings are equal!");
else
    System.out.println("Strings are not equal!");
```

Could have used `s2.equals(s1)` in the above.
Another example:

```java
String s1 = "dog";
String s2 = "cat";
if (s1 == s2)
    System.out.println("Strings are equal!"); // Correct answer, but
else //for the wrong reason!!
    System.out.println("Strings are not equal!");

s1 = s2; // What happens to the memory used by s1?
System.out.println(s1); // What is output here?
if (s1 == s2)
    System.out.println("Strings are equal!");
else
    System.out.println("Strings are not equal!");

=> "a CAT, is NOT, a DOG!" (from the musical Cats)
- Note that `equals` is case-sensitive! (what does that mean?)

- Normally these methods are called on variables, but not always:
  - `s1.equals(s2)` // The typical case
  - `s1.equals("dog")` // Also very common
  - "dog".equals(s1) // Unusual, but occasionally used
  - "dog".equals("cat") // Least likely

- To test for equality ignoring case, use `equalsIgnoreCase`:
  - `if (s1.equalsIgnoreCase("Hello"))` would evaluate to `true` if `s1` contained "hello"
Strings and Testing for Equality

```java
import java.util.*;

public class StringEqualityDemo {
    public static void main(String[] args) {
        String s1, s2;
        System.out.println("Enter two lines of text:");
        Scanner keyboard = new Scanner(System.in);
        s1 = keyboard.nextLine();
        s2 = keyboard.nextLine();
        if (s1.equals(s2))
            System.out.println("The two lines are equal.");
        else
            System.out.println("The two lines are not equal.");
        if (s2.equals(s1))
            System.out.println("The two lines are equal.");
        else
            System.out.println("The two lines are not equal.");
    }
}
```

Sample Screen Dialog

Enter two lines of text:
Java is not coffee.
Java is NOT COFFEE.
The two lines are not equal.
The two lines are not equal.
But the lines are equal, ignoring case.
Get ready, because “this” is going to be a bit strange…

Note on terminology – method calls in Java take the following form:

- `object.method(parameter_1, parameter_2,...,parameter_N)`

In such cases `object` is frequently referred to as “this.”

Example:

- `equals(Other_String)` – Returns true if this string and `Other_String` are equal. Otherwise, returns false.
- `s1.equals(s2)` – Here, `s1` is “this.”
Lexicographic Order

- Lexicographic order is similar to alphabetical order.

- Based on the order of all the ASCII and Unicode characters:
  - All digits come before all the letters.
  - All uppercase letters come before all lower case letters.

- The same as alphabetical ordering when all characters are either upper or lower, but not when upper and lower case are mixed.
  - Zip comes before apple

- Sorting and ordering are really big deals, both algorithmically and historically (check the wikipedia).
Lexicographic Order

- Some additional string methods:

  `toUpperCase()` – Returns a new string having the same characters as `this` string, but with any lowercase letters converted to uppercase.

  `toLowerCase()` – Returns a new string having the same characters as `this` string, but with any uppercase letters converted to lowercase.

  `compareTo(A_String)` – Compares `this` string with `A_String` to see which string comes first in lexicographic ordering. Returns a negative integer if `this` string is first, returns zero if the two strings are equal, and returns a positive integer if `A_String` is first.
Lexicographic Order

Example:

String s1 = "Hello";
String s2 = s1.toLowerCase();
String s3 = "hello";
if (s2.compareTo(s3) == 0)
    System.out.println("Equal!");
else if (s2.compareTo(s3) < 0)
    System.out.println("s2 comes first lexicographically!");
else
    System.out.println("s3 comes first lexicographically!");
Example:

```java
String s1 = "Hello";
String s2 = "dog";
String s3 = "hello";
if (s2.compareTo(s3) == 0)
    System.out.println("Equal!");
else if (s2.compareTo(s3) < 0)
    System.out.println("s2 comes first lexicographically!");
else
    System.out.println("s3 comes first lexicographically!");
```
Lexicographic Order

Example:

```java
String s1 = "Hello";
String s2 = "dog";
String s3 = "Hello";
if (s2.compareTo(s3) == 0)
    System.out.println("Equal!");
else if (s2.compareTo(s3) < 0)
    System.out.println("s2 comes first lexicographically!");
else
    System.out.println("s3 comes first lexicographically!");
```
Example ignoring upper and lower case:

```java
String s1 = "Hello";
String s2 = "hello";
if (s1.compareToIgnoreCase(s2) == 0)
    System.out.println("Equal!");
else if (s1.compareToIgnoreCase(s2) < 0)
    System.out.println("s1 comes first lexicographically!");
else
    System.out.println("s2 comes first lexicographically!");
```