Documentation

The Java™ Tutorials

Trail: Learning the Java Language
Lesson: Numbers and Strings
Section: Strings

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use technology no longer available.
See Java Language Changes for a summary of updated language features in Java SE 9 and subsequent releases.
See JDK Release Notes for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Comparing Strings and Portions of Strings

The `String` class has a number of methods for comparing strings and portions of strings. The following table lists these methods.

Methods for Comparing Strings

Method	Description
`boolean endsWith(String suffix)` `boolean startsWith(String prefix)`	Returns `true` if this string ends with or begins with the substring specified as an argument to the method.
`boolean startsWith(String prefix, int offset)`	Considers the string beginning at the index `offset`, and returns `true` if it begins with the substring specified as an argument.
`int compareTo(String anotherString)`	Compares two strings lexicographically. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument.
`int compareToIgnoreCase(String str)`	Compares two strings lexicographically, ignoring differences in case. Returns an integer indicating whether this string is greater than (result is > 0), equal to (result is = 0), or less than (result is < 0) the argument.
`boolean equals(Object anObject)`	Returns `true` if and only if the argument is a `String` object that represents the same sequence of characters as this object.
`boolean equalsIgnoreCase(String anotherString)`	Returns `true` if and only if the argument is a `String` object that represents the same sequence of characters as this object, ignoring differences in case.
`boolean regionMatches(int toffset, String other, int ooffset, int len)`	Tests whether the specified region of this string matches the specified region of the String argument. Region is of length `len` and begins at the index `toffset` for this string and `ooffset` for the other string.
`boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)`	Tests whether the specified region of this string matches the specified region of the String argument. Region is of length `len` and begins at the index `toffset` for this string and `ooffset` for the other string. The boolean argument indicates whether case should be ignored; if true, case is ignored when comparing characters.
`boolean matches(String regex)`	Tests whether this string matches the specified regular expression. Regular expressions are discussed in the lesson titled "Regular Expressions."

The following program, `RegionMatchesDemo`, uses the `regionMatches` method to search for a string within another string:

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

```
            String searchMe = "Green Eggs and Ham";
            String findMe = "Eggs";
            int searchMeLength = searchMe.length();
            int findMeLength = findMe.length();
            boolean foundIt = false;
            for (int i = 0;
                 i <= (searchMeLength - findMeLength);
                 i++) {
               if (searchMe.regionMatches(i, findMe, 0, findMeLength)) {
                  foundIt = true;
                  System.out.println(searchMe.substring(i, i + findMeLength));
                  break;
               }
            }
            if (!foundIt)
                System.out.println("No match found.");
        }
    }
```

The output from this program is `Eggs`.

The program steps through the string referred to by `searchMe` one character at a time. For each character, the program calls the regionMatches method to determine whether the substring beginning with the current character matches the string the program is looking for.

Previous page: Manipulating Characters in a String
Next page: The StringBuilder Class