Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
String c = "abc".substring(2, 3);
String d = cde.substring(1, 2);
```

The class `String` includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the `Character` class.

The Java language provides special support for the string concatenation operator (+), and for conversion of other objects to strings. For additional information on string concatenation and conversion, see *The Java Language Specification*.

Unless otherwise noted, passing a `null` argument to a constructor or method in this class will cause a `NullPointerException` to be thrown.

A `String` represents a string in the UTF-16 format in which *supplementary characters* are represented by *surrogate pairs* (see the section Unicode Character Representations in the `Character` class for more information). Index values refer to `char` code units, so a supplementary character uses two positions in a `String`.

The `String` class provides methods for dealing with Unicode code points (i.e., characters), in addition to those for dealing with Unicode code units (i.e., `char` values).

Unless otherwise noted, methods for comparing Strings do not take locale into account. The `Collator` class provides methods for finer-grain, locale-sensitive String comparison.

Implementation Note:

The implementation of the string concatenation operator is left to the discretion of a Java compiler, as long as the compiler ultimately conforms to *The Java Language Specification*. For example, the `javac` compiler may implement the operator with `StringBuffer`, `StringBuilder`, or `java.lang.invoke.StringConcatFactory` depending on the JDK version. The implementation of string conversion is typically through the method `toString`, defined by `Object` and inherited by all classes in Java.

See *Java Language Specification*:

15.18.1 String Concatenation Operator +

Since:

1.0

See Also:

`Object.toString()`, `StringBuffer`, `StringBuilder`, `Charset`, Serialized Form

Field Summary

Fields