# Strings: Introduction

## Strings

Object of class ***java.lang.String***

```
String s = new String(); // empty string
String s = new String("hello!");

char[] cArray = {'h', 'e', 'l', 'l', 'o', '!'};
String s = new String(cArray);

String s = "hello!"; // string literal
```

- String class uses **character array** to store text
- Java uses **UTF-16** for characters
- String is *sequence of unicode characters*
- String is *immutable*

String object ~ *immutable sequence of unicode characters*

## String is Special

- String literal
- **+** operator

```
String s = "hello" + " world!"; // "hello world!"
```

- String pool ~ *saves memory*

# Common Operations

- Comparing
- Searching
- Examining individual characters
- Extracting substrings
- Case translation
- Replace
- Split

## String Literal vs Using new

- String (via *string literal*)
  - Stored in *string pool* on heap
  - Literals with same content *share storage*
- String (via *new*)
  - Same as regular object
  - No storage sharing

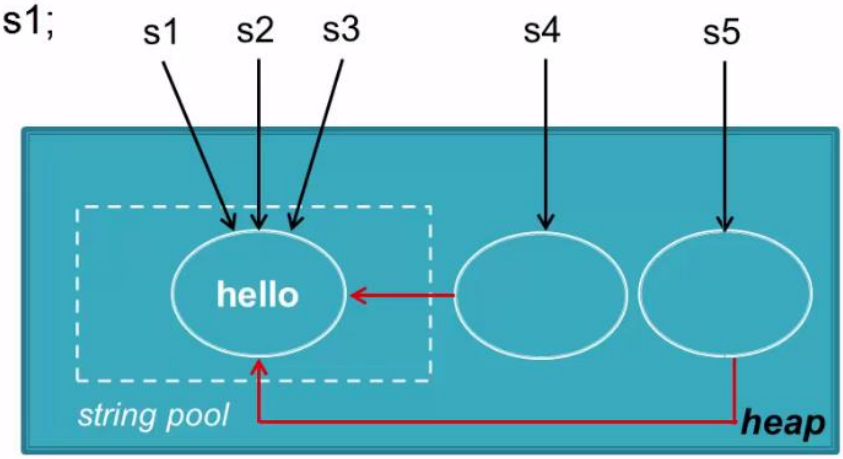
String *s1* = "hello";

String *s2* = "hello";

String *s3* = s1;

String *s4* = **new** String("hello");

String *s5* = **new** String("hello");




s1 == s2? True

s4 == s5? False

# String Concatenation

- StringBuilder
- StringBuffer

## StringBuilder

- From Java 5
- Example

```
StringBuilder sb = new StringBuilder();
sb.append("hello");
sb.append(" world!");
String s = sb.append(" Good").append(" morning").toString();
```

- Other methods: *length*, *delete*, *insert*, *reverse*, *replace*
- Not synchronized

# StringBuffer

- Obsolete. Use *StringBuilder!*
- Synchronized ~ slow
- API compatible with StringBuilder