(Non-)Egalité de `Strings`

Les opérateurs suivants :

`==`	égalité
`!=`	non-égalité

testent si deux variables `String` **font référence** (ou non) à la même zone mémoire (occupée par une chaîne de caractères).

Ceci est le cas lorsque les variables de types `String` ont été initialisées au moyen de **littéraux**

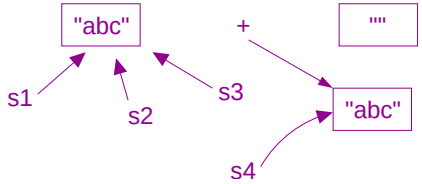
Exemple : utilisation de l'opérateur `!=`

```
while (reponse != "oui") ....;
```

(Non-)Egalité de `Strings` (2)

```
String s1 = "abc";     // s1 pointe vers le littéral "abc"
String s2 = "abc";     // idem (donc même zone mémoire que s1)
String s3 = s2;        // s3 stocke la même adresse que s2
String s4 = s1 + "";   // s4 contient l'adresse d'une nouvelle chaîne
                       // (construite par concaténation)
System.out.println((s1==s2) && (s2==s3)); // affiche true
System.out.println(s4);                   // affiche abc
System.out.println((s1==s4));             // affiche false
```

Situation en mémoire :



Comment faire pour **comparer les contenus référencés** plutôt que les références ?

☞ **Traitement spécifique** aux `String`

Comparaison de `String`

`chaine1.equals(chaine2)` teste si les chaînes de caractères référencées par `chaine1` et `chaine2` sont constituées des mêmes caractères

```
String s1 = "abc";
String s2 = "aBc";
String s4 = s1 + "";

System.out.println(s1.equals(s4)); // true
System.out.println(s1.equals(s2)); // false
```