A propos de String

Calculer intersection de deux ensembles de lettres

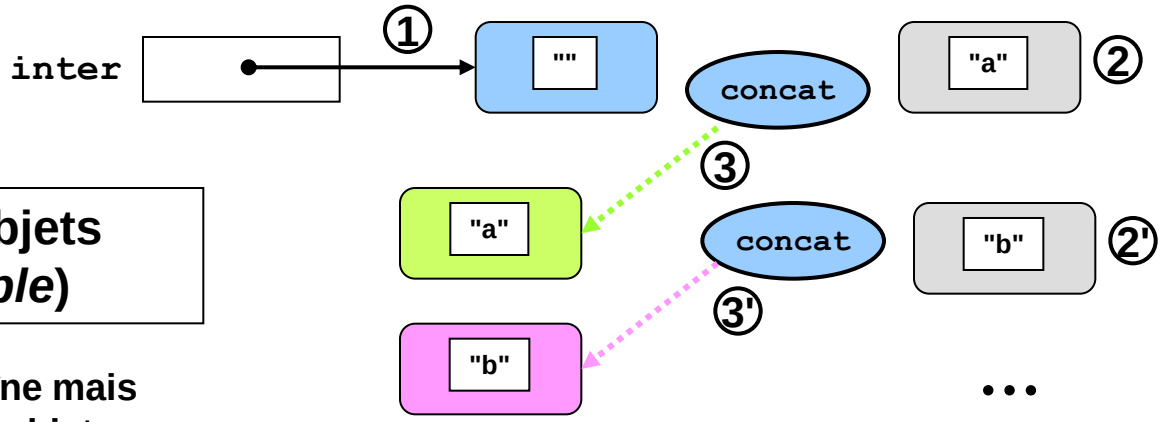
```
public EnsembleDeLettres intersection1(EnsembleDeLettres e) {
    String inter = "";
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        if (this.present[i] && e.present[i]) {
            inter.concat(toChar(i));
        }
    }
    return new EnsembleDeLettres(inter);
}
```

```
EnsembleDeLettres e1 = new EnsembleDeLettres("abcdefghijklmnopqrstuvwxyz");
EnsembleDeLettres e2 = new EnsembleDeLettres("abcdefghijklmnopqrstuvwxyz");
EnsembleDeLettres e3 = e1.intersection1(e2);
```

`e3 == {}` **Donne l'ensemble vide. Pourquoi ?**

A propos de String

```
public EnsembleDeLettres intersection1(EnsembleDeLettres e) {
 ① String inter = "";
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        if (this.present[i] && e.present[i]) {
            inter.concat(toChar(i));
        }          ③          ②
    }
    return new EnsembleDeLettres(inter);
}
```

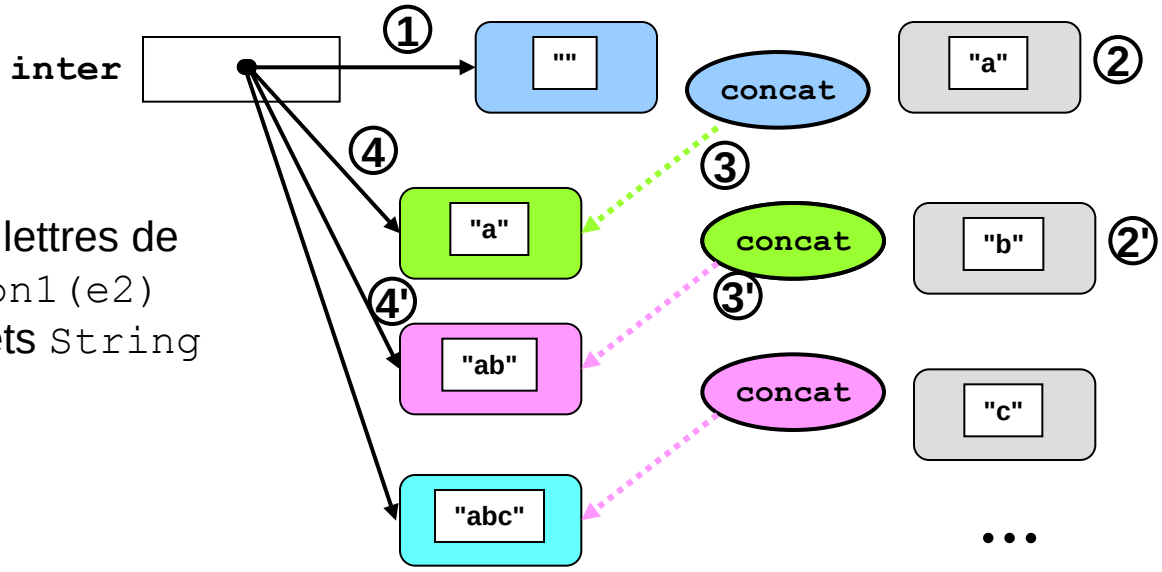


Les String sont des objets immuables (*immutable*)

concat ne modifie pas la chaîne mais crée et retourne un nouvel objet

A propos de String

```
public EnsembleDeLettres intersection1(EnsembleDeLettres e) {
 ① String inter = "";
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        if (this.present[i] && e.present[i]) {
              inter = inter.concat(toChar(i));  // inter = inter + toChar(i);
        }      ④        ③          ②
    }
    return new EnsembleDeLettres(inter);
}
```



si `e1` et `e2` contiennent les 26 lettres de l'alphabet, `e1.intersection1(e2)` provoque la création de **53** objets `String`

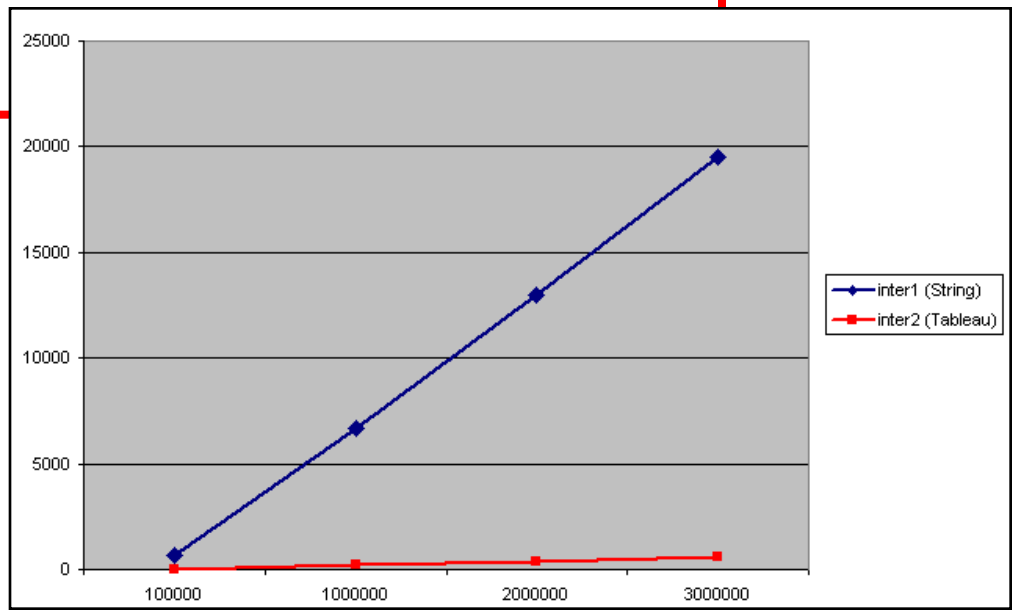
Performances

```
public EnsembleDeLettres intersection1(EnsembleDeLettres e) {
    String inter = "";
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        if (this.present[i] && e.present[i]) {
            inter += toChar(i));
        }
    }
    return new EnsembleDeLettres(inter);
}
```

```
public EnsembleDeLettres intersection2(EnsembleDeLettres e) {
    EnsembleDeLettres res = new EnsembleDeLettres(true); // res == ens vide
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        res.present[i] = this.present[i] && e.present[i];
    }
    return res;
}
```

- La création d'objets coûte cher.

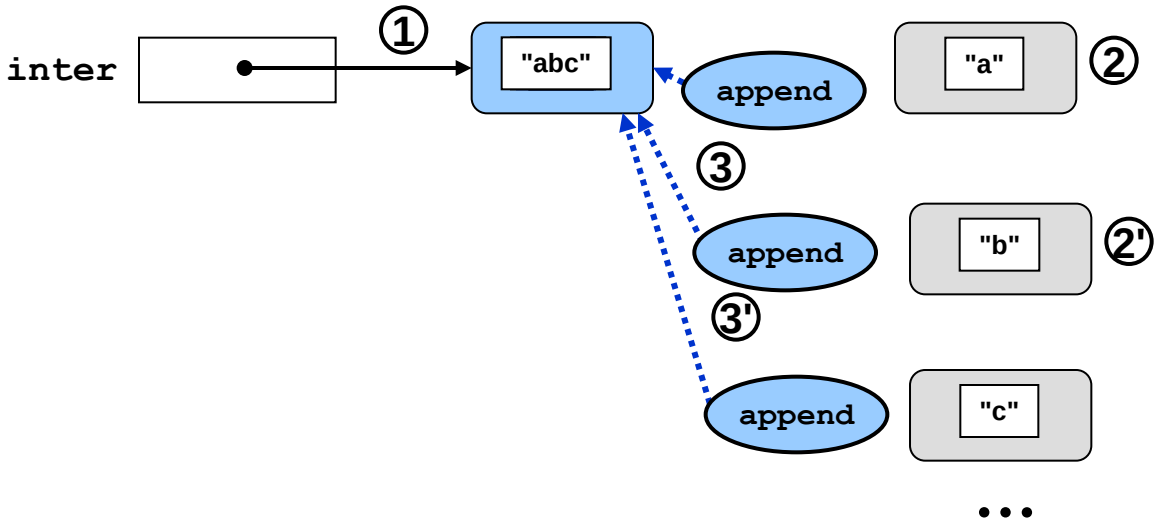
```
e1 = new EnsembleDeLettres(
        "abcdefghijklmnopqrstuvwxyz");

e2 = new EnsembleDeLettres(
          "abcdefghijklmnopqrstuvwxyz");

e3 = e1.intersection1(e2);
```



StringBuilder vs. String

- A `StringBuilder` implements a mutable sequence of characters. A string buffer is like a `String`, but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls. .

```
public EnsembleDeLettres intersection3(EnsembleDeLettres e) {
    String inter = "";          StringBuilder inter = new StringBuilder();
    for (int i = 0; i < present.length; i++) {
        // l'element i est dans l'interection si il est dans this ET dans e
        if (this.present[i] && e.present[i]) {
            inter += toChar(i);          inter.append(toChar(i));
        }                                          ③          ②
    }
    return new EnsembleDeLettres(inter);
}                       return new EnsembleDeLettres(inter.toString());
```



Performances String/StringBuilder

Nombre d'itérations	100000	1000000	2000000	3000000
inter1 (String)	687	6640	13000	19547
inter2 (Tableau)	16	203	406	625
inter3 (StringBuilder)	141	1282	2578	3843

inter1/inter2	42,94	32,71	32,02	31,28
inter1/inter3	4,87	5,18	5,04	5,09