Compléments : String et StringBuffer

Exercice 1 : `classes String et StringBuffer,` profil des méthodes
Le but de cet exercice est de faire explorer la classe `java.lang.String` en testant ses différentes méthodes sur des chaînes et d'autres valeurs lues au clavier. Il s'agit d'écrire la classe exécutable `TestChaines` dont la méthode principale effectue les opérations suivantes :

1. Lire une chaîne *s1* et afficher sa longueur (méthode `length`).
2. Lire deux chaînes *s1* et *s2* et afficher les résultats renvoyés par les expressions :
   - `s1==s2`
   - `s1.equals(s2)`
   - `s1.compareTo(s2)`
   - `s1.compareToIgnoreCase(s2)`

   Entre autres, essayer les couples "abcd" et "abcd", puis "abcd" et "AbcD".
3. Lire deux chaînes *s1* et *s2* et afficher la réponse à la question : « ces deux chaînes commencent-elles par le même caractère ? ». Utiliser la méthode d'instance `charAt`.
4. Lire deux chaînes *s1* et *s2* et afficher la réponse aux questions :
   - *s1* commence-t-elle par *s2* ? (méthode `startsWith`)
   - *s1* finit-elle par *s2* ? (méthode `endsWith`)
   - *s1* contient-elle *s2* ? (méthode `contains` : `String` implémente `CharSequence` → une donnée de type `String` peut être transmise en argument)
5. Lire deux chaînes *s1* et *s2* et concaténer *s2* à *s1*. Par exemple, si s1= "abc" et s2= "de" , le résultat est "abcde". Opérateur + ou méthode `concat`.
6. Lire une chaîne *s1* et remplacer toutes les instances de caractère 'o' par 'O' (méthode `replace`).
7. Lire deux chaînes *s1* et *s2* et si s1 contient s2, renvoyer *s1* privée de *s2* sinon renvoyer *s1*. Utiliser les méthodes `indexOf`, `substring`.
8. Lire une chaîne représentant un nom de ville et l'afficher entièrement en majuscules (méthode `toUpperCase`).
9. Lire une chaîne représentant un nom de ville, lui enlever les éventuels blancs en début et à la fin (méthode `trim`). Exemple d'appel : `TestChaines "  Burnhaupt le haut  "`
10. Lire une chaîne *s1* comportant plusieurs noms de villes séparés par un et un seul « blanc ».
    Décomposer la chaîne pour accéder directement à chacun des noms de ville
    Méthode `split` dont la profil est : `public String[] split (String regex)`
    Vous donnerez à `regex` la valeur " " (un blanc).
    Afficher un nom de ville par ligne.
    Exemple d 'appel : `TestChaines "Paris Lyon Marseille Strasbourg"`

Les objets de type `String` ne sont pas modifiables. Leur manipulation conduit à la création de nouvelles chaînes et de petits changements dans le contenu de la chaîne peuvent être coûteux. Dans les programmes manipulant intensivement les chaînes, la perte de temps et la place mémoire sont importantes. Ainsi, en particulier, l'opérateur += est très inefficace. Par exemple, `str+='A'` est implémenté comme : `str=str+'A'`. Cela signifie qu'une nouvelle chaîne est crée dont la valeur est le résultat de `str+'A'` et `str` référence une nouvelle chaîne.

Faire le test suivant :

- Lire une chaîne `str`
- afficher `str+'A'` ou `str.concat('A')`
- Afficher `str`

La classe `StringBuffer` permet de manipuler les chaînes en modifiant les objets.
Effectuer les opérations suivantes sur la classe `StringBuffer` :

1. Utiliser les 3 constructeurs de la classe :
   - `public StringBuffer()` (→ longueur = 0)
   - `public StringBuffer(int capacity)` (→ longueur = 0)
   - `public StringBuffer(String str)`
2. Utiliser les méthodes pour modifier votre chaîne : `append, insert, setCharAt, replace, reverse`...
   Noter que le 3ème argument de la méthode replace est de type `String` ; `StringBuffer` n'étant pas une sous-classe de `String`, on ne peut transmettre un `StringBuffer` en 3ème argument.
3. Faire le même test que pour une instance de `String` pour vérifier que l'instance de `StringBuffer` est modifiable.