DUT informatique – Algorithmique/Programmation – TP Morpion

- Testez chaque méthode **DÈS QUE VOUS L'AVEZ PROGRAMMÉE** et laissez les tests dans la méthode `main` de la classe concernée. Les tests font partie de la note.
- À la fin du TP, envoyez vos fichiers `.java` à l'adresse suivante : `Pierre.Courtieu@cnam.fr`.
- Pendant tout le TP vous devez consulter la documentation de la librairie standard de Java (Java API Specification), située à l'adresse `http://java.sun.com/j2se/1.5.0/docs/api/` (attention à la version de java installée dans la salle de TP). Cette documentation est rédigée dans un anglais très simple.

Exercice 1 Nous allons partir des différentes classes utilisées par la classe `Morpion` (`Plateau`, `Coordonnées` et `Coche`) afin de programmer le jeu de *Puissance 4*. Il s'agit ici aussi d'aligner 4 pions du même joueur, mais les pions ne peuvent se placer que juste au-dessus d'un autre pion ou tout en bas de la grille.

Initialement nous laisserons la grille avoir la taille 10x10.

- De quelle(s) méthode(s) avez-vous besoin pour vous assurer que les joueurs ne peuvent placer des coches qu'aux endroit permis ?
- Où (c'est-à-dire dans quelle(s) classe(s)) ce(s) méthode(s) doivent-elles être implantées ?
- Implantez et testez ces méthodes.
- Implantez le reste des méthodes nécessaires. Testez.
- Implantez le jeu de Puissance 4.
- Modifiez le programme de Morpion et de Puissance 4 afin de permettre aux joueurs de décider de la taille de la grille.
- Même question pour permettre de décider combien de coches alignées donnent la victoire.

□

Exercices

Exercice 2 Consultez la documentation. Cherchez la classe `String` (dans le cadre en bas à gauche) et cliquez dessus. Descendez à la section `Method Summary` (dans le cadre principal à droite). Cette section vous donne la liste des *méthodes* que l'on peut appeler sur une chaîne de caractère. Par exemple la fonction `charAt` (**char** `charAt(`**int** `index)`) permet d'obtenir le caractère d'une chaîne à la position `index`. Vous pouvez cliquer sur le nom d'une méthode pour accéder à une documentation un peu plus précise.

1. Écrivez, compilez et lancez le programme suivant (dans un fichier appelé `Tp1.java`) :

```
class Tp1 {
    public static void main(String[] args) {
        String s = "Bonjour"; // Création d'un objet de la classe String
        char c;               // Création d'un char (minuscule: pas un objet)
        c = s.charAt(0); // On peut utiliser les méthodes de String sur s

        Terminal.ecrireString("le␣caractère␣0␣est␣" + c + "\n");
        c = s.charAt(3);
        Terminal.ecrireString("le␣caractère␣3␣est␣" + c + "\n");
        c = s.charAt(7);
        Terminal.ecrireString("le␣dernier␣caractère␣est␣" + c + "\n");
    }
}
```

2. Corrigez la méthode `main` en retournant un caractère par défaut lorsque l'exception est levée (**try** ... **catch** ...). **Testez**.

3. Écrivez dans la méthode `main` une boucle affichant tous les caractères de la chaînes (chacun sur une ligne). **Testez**.
4. Écrivez une méthode `afficheChars` qui prend en argument une chaîne de caractères et affiche ses caractères chacun sur une ligne. **Testez** sur plusieurs chaînes.
5. Écrivez une méthode `estPalindrome` qui prend en argument une chaînes de caractères et retourne **true** si celle-ci est un palindrome et **false** sinon. **Testez** sur plusieurs chaînes.
6. Écrivez une méthode `sansVoyelle` qui prend en argument une chaînes de caractères et retourne une *nouvelle* chaîne qui contient les même lettres que l'argument sans les voyelles. **Testez** sur plusieurs chaînes.
7. Écrivez une méthode `inverse` qui prend en argument une chaînes de caractères et retourne une nouvelle chaîne contenant les mêmes lettres dans l'ordre inverse.

□

Exercice 3 Regardez la documentation des méthodes `replacefirst` et `contains` de la classe `String` (considérez que la classe `CharSequence` est équivalente à `String`).

1. Ajoutez à la méthode `main` les lignes suivantes, puis testez.

```
String s2=s.replaceFirst("nj","t");
Terminal.ecrireString("la␣chaine␣s␣est␣" + s + "\n");
Terminal.ecrireString("la␣chaine␣s2␣est␣" + s2 + "\n");
if (s.contains("onjo")) Terminal.ecrireString("la␣chaine␣s␣contient␣\"onjo\"\n");
else Terminal.ecrireString("la␣chaine␣s␣ne␣contient␣pas␣\"onjo\"\n");
if (s2.contains("onjo")) Terminal.ecrireString("la␣chaine␣s2␣contient␣\"onjo\"\n");
else Terminal.ecrireString("la␣chaine␣s2␣ne␣contient␣pas␣\"onjo\"\n");
```

   Déduisez de l'exemple de `replaceFirst` une caractéristique importante à connaître pour chaque fonction. Consultez la classe `StringBuffer` et voyez la différence.

2. Sachant qu'on peut copier une chaîne de caractère dans une `StringBuffer` grâce à l'instruction suivante (`s` est une chaîne déclarée avant) :

```
StringBuffer sb = new StringBuffer(s);
```

   et vice-versa (sb étant un `StringBuffer` déclaré avant) :

```
String s = new String(sb);
```

   Reprogrammez `inverse` en utilisant la fonction `reverse` de la classe `StringBuffer`.

3. Testez ce morceau de code :

```
StringBuffer sb = new StringBuffer("HelloWorld");
Terminal.ecrireString("la␣stringbuffer␣sb␣=␣" + sb + "\n");
StringBuffer sb2 = sb.append('a');
Terminal.ecrireString("le␣stringbuffer␣sb␣=␣" + sb + "\n");
Terminal.ecrireString("la␣stringbuffer␣sb2␣=␣" + sb2 + "\n");
```

   Reprogrammez une nouvelle méthode `sansVoyelle` qui, au lieu de retourner une nouvelle chaîne sans voyelle, supprime les voyelles dans la chaîne elle-même. La fonction prendra donc un `StringBuffer` en argument.

□