TD N°07 JAVA

Exercise 1 :

```
public class TestChaine {

  public static void main(String[] args) {
    // Attention, les caractères sont numérotés à partir de 0 !
//    String chaine = "La chaine de test";
    String chaine = "123456789ABCdef";
    // 3 premiers caractères
    System.out.println("-" + chaine.substring(0, 3) + "-");
    String chaine2 = chaine.substring(0, 3);
    // Les 3 derniers
    System.out.println("-" + chaine.substring(chaine.length() - 3) + "-");
    // Le caractère central
    // Si le nombre de caractères est pair, il y a 2 caractères au centre
    // Le code suivant n'est pas optimisé ; vous pouvez vous amuser à
l'optimiser.
    int longueur = chaine.length();
    if (longueur % 2 == 0) {
      // Longueur paire ; affiche les 2 caractères centraux.
      // Longueur avant et après
      // (et aussi numéro du 1er caractère central car numérotation
commence à 0)
      int demiLongueur = (longueur - 2) / 2;
      System.out.println("-" + chaine.substring(demiLongueur, demiLongueur
+ 2) + "-");
      String avant = chaine.substring(0, demiLongueur);
      System.out.println("Avant : " + avant);
      String apres = chaine.substring(demiLongueur + 2);
      System.out.println("Après : " + apres);
    }
    else {
      // Longueur impaire ; affiche le caractère central
      // Longueur avant et après (et aussi numéro du caractère central)
      int demiLongueur = longueur / 2;
      System.out.println("-" + chaine.substring(demiLongueur, demiLongueur
+ 1) + "-");
      String avant = chaine.substring(0, demiLongueur);
      System.out.println("Avant : " + avant);
      String apres = chaine.substring(demiLongueur + 1);
      System.out.println("Après : " + apres);
    }
    // Majuscules puis minuscules
    System.out.println(chaine.toUpperCase());
    System.out.println(chaine.toLowerCase());
    // Test à la casse près
    System.out.println(chaine.equalsIgnoreCase("123456789abCdEf"));
    // Comparaison lexicographique de 2 chaînes
    String chaine21 = "123";
    String chaine22 = "13";
    int v = chaine21.compareTo(chaine22);
    if (v < 0) {
      System.out.println(chaine21 + " est avant " + chaine22);
    }
    else if (v > 0) {
      System.out.println(chaine21 + " est après " + chaine22);
```

PROGRAMMING COURSES

```
    }
    else {
      System.out.println(chaine21 + " est égal à " + chaine22);
    }
    chaine21 = "écrire";
    chaine22 = "ecrire";
    v = chaine21.compareTo(chaine22);
    if (v < 0) {
      System.out.println(chaine21 + " est avant " + chaine22);
    }
    else if (v > 0) {
      System.out.println(chaine21 + " est après " + chaine22);
    }
    else {
      System.out.println(chaine21 + " est égal à " + chaine22);
    }
  }
}
```

Exercise 2 :

```
/**
 * Quelques tests de manipulations de noms de fichiers.
 */
public class TestNomFichier {
  public static boolean isJavaSource(String nomFichier) {
    return nomFichier.endsWith(".java");
  }

  public static void main(String[] args) {
    String nomFichier = args[0];
    int dernierSlash = nomFichier.lastIndexOf("/");
    // Nom terminal
    System.out.println(nomFichier.substring(dernierSlash + 1));
    // Répertoire racine
    int deuxiemeSlash = nomFichier.indexOf("/", 1);
    System.out.println(nomFichier.substring(0, deuxiemeSlash));
    // Répertoire dans lequel le fichier se trouve
    //  - son nom absolu
    String nomAbsoluRepertoire = nomFichier.substring(0, dernierSlash);
    System.out.println(nomAbsoluRepertoire);
    //  - son nom relatif
    dernierSlash = nomAbsoluRepertoire.lastIndexOf("/");
    System.out.println(nomAbsoluRepertoire.substring(dernierSlash + 1));
    // Test de isJavasource
    if (isJavaSource(nomFichier)) {
      System.out.println("Source Java");
    }
    else {
      System.out.println("Pas source Java");
    }
  }
}
```

PROGRAMMING COURSES

Exercise 3 :

```
import java.util.Arrays;

public class Chaine {
  public static String[] eclater(String chaine, String separateur) {
    String[] parties = new String[100];
    int i = 0;
    int position;
    while ((position = chaine.indexOf(separateur)) != -1) {
      parties[i++] = chaine.substring(0, position);
      chaine = chaine.substring(position + separateur.length());
      System.out.println(position + ";" + chaine);
    }
    // Traitement de la fin de la chaîne
    if (!chaine.isEmpty()) {
      parties[i++] = chaine;
    }
    // Retourne un tableau plein
    return Arrays.copyOf(parties, i);
  }

  public static String[] eclater(String chaine, String separateur, boolean
enleverEspaces) {
    String[] parties = chaine.split(separateur);
    if (enleverEspaces) {
      for (int i = 0; i < parties.length; i++) {
        parties[i] = parties[i].trim();
      }
    }
    return parties;
  }

  public static void main(String[] args) {
    String prenoms = "vincent;; françois;; paul;; et les autres;; ";
    System.out.println(Arrays.toString(eclater(prenoms, ";;")));
    System.out.println(Arrays.toString(eclater(prenoms, ";;", true)));
  }
```

PROGRAMMING COURSES

Exercise 4 :

```
/**
 * Comparaison de la concaténation par String et StringBuilder.
 */
public class Concatenation {

  public static void main(String[] args) {
    // Par String
    long debut = System.currentTimeMillis();
    for (int i = 0; i < 100000; i++) {
      String concat = "";
      for (String c : args) {
        concat += c;
      }
    }
    long fin = System.currentTimeMillis();
    System.out.println("Temps pour String : " + (fin - debut));
    // Par StringBuilder
    debut = System.currentTimeMillis();
    for (int i = 0; i < 100000; i++) {
      StringBuilder concat = new StringBuilder();
      for (String c : args) {
        concat.append(c);
      }
    }
    fin = System.currentTimeMillis();
    System.out.println(fin - debut);
  }

}
```

PROGRAMMING COURSES