TD1

Exercice 1 : Quelles valeurs contiennent les variables n, p et q après évaluation de chaque expression?

```
int n = 3, p = 6, q = 2;
p = q = 10;
n += p += q;
q = n < p ? n++ : p++;
q = n > p ? n++ : p++;
n = q==p ? ++p : --q;
```

Exercice 2 :

- **Définition** : "Un palindrome est une séquence qui se lit identiquement de gauche à droite et de droite à gauche."
- **Exemples** :
  - Mots : radar, kayak.
  - Nombres : 12321, 1331.

1- Écrire un programme qui vérifie si une séquence de caractères (sans espaces) est un palindrome en utilisant uniquement des tableaux de char.

- Demander à l'utilisateur de saisir une séquence de caractères sans espaces (ex : radar, 12321).

  Utiliser la méthode `public String nextLine()` de la classe `Scanner`.
- Stocker cette séquence dans un tableau de char.

  Utiliser méthode `public char[] toCharArray()` de la classe `String`.
- Vérifier si la séquence est un palindrome en comparant les caractères deux à deux (premier avec dernier, deuxième avec avant-dernier, etc.).
- Afficher :
  - C'est un palindrome ! si la séquence est un palindrome.
  - Ce n'est pas un palindrome ! si la séquence n'est pas un palindrome.

2- Adapter le programme pour qu'il accepte des phrases (ex : "Ésope reste ici et se repose") en :

- Ignorant les espaces et la ponctuation.

  Utiliser la méthode `public String.replaceAll (String regex, String remplacement)` avec `regex = "[^a-zA-Z0-9]"` pour supprimer les caractères non alphanumériques.
- Ignorant la casse (majuscules/minuscules).

  Utiliser `public String toLowerCase()` de la classe `String` pour ignorer la casse.
- Vérifiant si la phrase est un palindrome à partir du `String` directement sans passer par un tableau de `char`.

  Utiliser la méthode `public char charAt( int index)` de la classes `String`.

3 - Vérifier si un nombre entier est un palindrome (ex : 12321).

- Lire un nombre entier depuis l'entrée utilisateur.
- Extraire chaque chiffre du nombre et les stocker dans un tableau.
- Vérifier si le tableau de chiffres est un palindrome.

Exemples :

- 12321 → Palindrome
- 12345 → Pas un palindrome"*

4 - Écrire un programme qui :

1. Génère tous les palindromes à 3 chiffres (de 100 à 999).
2. Affiche la liste des palindromes trouvés.
   Exemple : 101, 111, 121, ..., 999".

Exercice 3. On considère la classe `Point` ci-dessous :

```
package exo4 ;
class Point {
	int x,y;

	public double distance(Point p){
		int diffx = this.x-p.x;
		int diffy = this.y-p.y;
		return Math.sqrt(diffx*diffx+ diffy*diffy) ;
	}
}
```

Ajouter dans le même package une classe `Main` contenant le `main`. Afficher la distance entre l'origine et le point de coordonnées (4,6).

Modifier la classe `Point` de manière à respecter le principe d'encapsulation. Les attributs auront une visibilité privée. On ajoutera les getters correspondant. Ajouter un constructeur avec deux paramètres permettant de construire un point quelconque. On souhaite aussi proposer un constructeur permettant de définir l'origine sans passer de paramètres. Faut-il ajouter un constructeur pour cela ou bien est-il défini par défaut ?

Surcharger la méthode `public double distance(Point p)` en ajoutant une méthode `public double distance()` qui calcule la distance à l'origine.

Ajouter les méthodes suivantes :
`public void translater (int u, int v)` permettant de translater le point.
`public Point symetrie()` qui retourne le point symétrique du point courant obtenu par symétrie centrale par rapport à l'origine.

On souhaite numéroter les points au fur et à mesure de leur création. Apporter les modifications nécessaires. On pourra ajouter un attribut `private int num` pour représenter le numéro d'un point.

Qu'affiche le `main` ci-dessous ?

```
public class Main {
```

```
    public static void main(String[] args) {
           Point p1= new Point(4,6);
           System.out.println("num : "+p1.getNum());
           Point p0= new Point();
           System.out.println("distance :"+p1.distance(p0));
           System.out.println("distance :"+p1.distance());
           p0.translater(3,8);
           System.out.println(p0.getNum()+ " "+p0.getX()+ " "+p0.getY());
           Point ps=p0.symetrie();
           System.out.println(ps.getNum()+ " "+ps.getX()+ " "+ps.getY());
      }
}
```

Exercice 4.

Créer une classe `Temps` permettant d'enregistrer un temps exprimé en heures, minutes et secondes, les minutes et les secondes étant comprises entre 0 et 59.
La classe `Temps` sera caractérisée par 3 attributs de types `int` : `heures`, `minutes` et `secondes`.
Elle fournira 3 constructeurs :

- un constructeur sans paramètre,
- un constructeur ayant 3 paramètres de types `int` : `heures`, `minutes`, `secondes`,
- un constructeur ayant un paramètre de type `long` : `t` qui permettra d'initialiser un temps exprimé en secondes.

On souhaite pouvoir consulter et modifier les attributs `heures`, `minutes` et `secondes` d'un objet de type `Temps`.
Choisissez une visibilité adaptée pour les trois attributs et ajoutez des accesseurs si besoin sachant qu'on devra toujours garantir que les valeurs des attributs `minutes` et `secondes` soient bien comprises entre 0 et 59.

Ajouter les méthodes suivantes :

- `public long conversion ()` qui retourne l'entier long obtenu en convertissant les heures, minutes, secondes en secondes.
- `public void ajouterTemps (Temps t)` qui ajoute au temps le temps t passé en paramètre.

Exercice 4. On souhaite gérer des listes d'entiers et proposer les services suivants : connaître le nombre d'entiers de la liste, calculer la somme des entiers de la liste, extraire l'indice du plus grand entier de la liste, ajouter un entier à la liste, créer une nouvelle liste d'entiers en ajoutant un entier à la liste initiale.

1. Créer une classe `ListeEntiers` caractérisée par un attribut privé de type un tableau de `int` nommé `lesEntiers`.

La classe proposera deux constructeurs :

- un constructeur permettant d'initialiser la tableau d'entiers avec un tableau d'entiers passé en paramètre : `public ListeEntiers ( int[] lesEntiers ).`
- un constructeur permettant d'initialiser le tableau avec un tableau de chaînes de caractères passé en paramètre. On suppose que chacune des chaînes du tableau

correspond à un entier.

`public ListeEntiers ( String[] lesEntrees ).`

2. Créer une classe `Main` contenant la méthode `main`.

- Créer une liste d'entiers à partir d'un tableau d'entier puis une autre liste d'entiers à partir d'un tableau de `String`.

3. Ajouter les méthodes suivantes à la classe `ListeEntiers` et les tester dans la classe `Main`:

    - `public int longueur()` qui retourne le nombre d'entiers de la liste
    - `public int somme()` qui calcule la somme des entiers de la liste
    - `public int indiceMax()` qui retourne l'indice du plus grand entier de la liste. S'il y a des doublons, elle retourne le plus petit indice. S'il n'y a pas d'entiers dans la liste, elle retourne -1.
    - `public void ajoute ( int n )` qui ajoute l'entier passé en paramètre à la fin de la liste.

A-t-on accès aux entiers présents dans chacune des listes ? Comment peut-on donner accès à ces entiers ? Comment peut-on procéder pour afficher le contenu de chacune des listes ? Est ce que cela respecte le principe d'encapsulation ?