BTS
BREVET DE TECHNICIEN SUPÉRIEUR
LYCÉE QUALIFIANT MOHAMED V
ESSAOUIRA

PROGRAMMATION ORIENTÉE OBJET (JAVA)

Les collections

1. Définition

Une collection est un objet qui contient d'autres objets

- **Par exemple, un tableau est une collection**
- **Le JDK fournit d'autres types de collections sous la forme de classes et d'interfaces**
- **Ces classes et interfaces sont dans le paquetage java.util**

1. Les interfaces: Collection

Deux hiérarchies principales :

- **List<E> correspond aux interfaces des collections proprement dites**
- **Map<K,V> correspond aux collections indexées par des clés ; un élément de type V d'une Map est retrouvé rapidement si on connaît sa clé de type K(comme les entrées d'un dictionnaire ou les entrées de l'index d'un livre)**

Les listes : ArrayList

- La gestion des ArrayList est similaire à la gestion des tableaux puisque le programme crée une liste par ajout de données au fur et à mesure des besoins de l'utilisateur.
- Les donnés sont enregistrées dans l'ordre d'arrivée (indice)
- Les données enregistrées dans une ArrayList sont en réalité rangées dans un tableau interne créé par l'interpréteur. La taille du tableau interne est gérée automatiquement par Java.
- Lorsque la liste des éléments à ajouter dépasse la taille du tableau interne, un nouveau tableau est créé et les anciennes valeurs y sont copiés.

ArrayList

- Déclaration : **ArrayList** liste=new **ArrayList**();
- Les méthodes :
  - ***add****(objet) // Ajoute un élément objet en fin de liste*
  - ***add****(indice, objet) // insère un élément objet à l'indice donné*
  - ***get****(indice) // retourne stocké à l'indice donné*
  - ***set****(indice,objet) // remplace l'élément situé en position indice par l'objet*
  - ***size****() // retourne le nombre d'élément dans la liste*
  - ***remove****(indice) // supprime l'objet dont l'indice est donné par paramètre*

- **removeRange**(i,j) // supprime tous les éléments compris entre les indices (i valeur comprise) et (j valeur comprise)
- **clear**() // supprime tous les éléments de la liste
- **indexof**(objet) // retourne l'indice dans la liste du premier objet donné ou -1 si objet n'existe pas
- **lastIndexof()** // retourne l'indice dans la liste dernier objet donné ou -1 si objet n'existe pas
- **contains** (objet) // retourne true ou false si l'objet existe ou non dans la liste

Exemple

```
class Animal {
    public void marche() {
        System.out.println("L'animal marche...");
    }
}
```

```
class Chien extends Animal {
    @Override
    public void marche() {
        System.out.println("Le chien court joyeusement !");
    }
}
```

```
class Chat extends Animal {
    @Override
    public void marche() {
        System.out.println("Le chat marche discrètement...");
    }
}
```

Exemple

```java
// Classe principale avec la méthode main
public class Main {
    public static void main(String[] args) {
        // Création d'une liste d'animaux
        List<Animal> animaux = new ArrayList<>();

        // Ajout de deux chiens, deux chats et un animal générique
        animaux.add(new Chien());
        animaux.add(new Chien());
        animaux.add(new Chat());
        animaux.add(new Chat());
        animaux.add(new Animal());

        // Parcours de la liste avec foreach et appel de la méthode marche()
        for (Animal animal : animaux) {
            animal.marche();
        }
    }
}
```

```
Le chien court joyeusement !
Le chien court joyeusement !
Le chat marche discrètement...
Le chat marche discrètement...
L'animal marche...
```

Exercice d’application

1. Classe Abstraite: Vehicule

•Elle a une méthode deplacer()

2. Classes dérivées :

•Voiture : définit deplacer() pour afficher un message spécifique.

•Velo : définit deplacer() pour un vélo.

•Bateau : définit deplacer() pour un bateau.

3. Classe Main

•Créer une List<Vehicule> contenant **2 voitures, 2 vélos, 1 bateau**.

•Utiliser une boucle **foreach** pour afficher le déplacement de chaque véhicule.