ENSET
ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA


# Programmation Orientée Objet

## Les collections et les streams

**Mme Hassna BENSAG**
Email: **h.bensag@gmail.com**

# Les collections

- les collections sont des structures de données qui permettent de stocker, manipuler un ensemble d'objets de manière flexible et dynamique. Voici quelques types de collections en java :
  - Les Listes: Une liste est une collection ordonnée d'éléments où chaque élément peut être accédé par sa position indexée. Les implémentations populaires de l'interface List comprennent ArrayList et LinkedList .
  - Les Set: Un ensemble est une collection qui ne permet pas les doublons. Les éléments sont stockés de manière non ordonnée. Exemple d'implémentation de l'interface Set est HashSet.
  - Les Map: Une carte (Map) est une collection qui associe des clés uniques à des valeurs correspondantes. Chaque clé est associée à une seule valeur. Exemple d'implémentation de l'interface l'interface c'est Map comprennent est HashMap.

# Méthodes de la classe ArrayList

- Déclaration: List <String> list= new ArrayList<>();

| Méthodes | Description |
|---|---|
| Add(E e) | Ajoute l'élément e à la fin de la liste. |
| addAll(Collection<? extends E> c) | Ajoute tous les éléments de la collection c à la fin de la liste. |
| get(int index) | Renvoie l'élément à l'index spécifié index. |
| remove(int index)<br>remove(Object o) | Supprime l'élément à l'index spécifié index.<br>Supprime la première occurrence de l'élément spécifié o (si présent) de la liste. |
| set(int index, E element) | Remplace l'élément à l'index spécifié index par element. |
| contains(Object o) | Renvoie true si l'élément o est présent dans la liste. |
| clear() | Supprime tous les éléments de la liste. |
| size() | Renvoie le nombre d'éléments dans la liste. |
| forEach(action) | Applique l'action spécifiée à chaque élément de la liste |

# Méthodes de la classe HashMap

- Déclaration: Map<String, Double> map= new HashMap<>();

| Méthodes | Description |
|---|---|
| put(K key, V value) | Associe la clé key à la valeur value. Si la clé est déjà présente, la valeur est mise à jour. |
| get(Object key) | Renvoie la valeur associée à la clé key, ou null si la clé n'existe pas dans la Map. |
| remove(Object key) | Supprime l'entrée associée à la clé key et renvoie sa valeur. Retourne null si la clé n'existe pas. |
| containsValue(Object o) | Renvoie true si la valeur value est présente dans la carte. |
| clear() | Vide la carte. |
| size() | Renvoie le nombre d'éléments de la carte. |
| forEach(action) | Applique l'action spécifiée à chaque paire clé-valeur de la carte. |

# Méthodes de la classe HashSet

- Déclaration: Set<String> cin= new HashSet<>();

| Méthodes | Description |
|---|---|
| Add(E e) | Ajoute l'élément e à l'ensemble. Si l'élément est déjà présent, l'ensemble ne change pas et renvoie false. |
| get(int index) | Renvoie l'élément à l'index spécifié index. |
| remove(Object o) | Supprime l'élément o de l'ensemble. Renvoie true si l'élément a été trouvé et supprimé, sinon false. |
| contains(Object o) | Renvoie true si l'élément o est présent dans l'ensemble sinon false. |
| clear() | Supprime tous les éléments de l'ensemble. |
| size() | Renvoie le nombre d'éléments de l'ensemble. |

# Les streams

- un "stream" fait référence à une séquence d'éléments sur laquelle on peut effectuer des opérations de traitement de données. L
- Les streams ont été introduits dans Java 8. Ils fournissent une manière concise de manipuler des collections de données, en permettant des opérations de filtrage, de mapping, de tri et d'autres opérations de traitement de données.
- Les streams sont souvent utilisés avec des collections telles que des listes (List), des ensembles (Set) ou des tableaux (Array).

# Les opérations des streams

- filter: Retourne un nouveau stream contenant les éléments qui satisfont la condition définie par le prédicat.
- map : Applique une fonction à chaque élément du stream et retourne un nouveau stream contenant les résultats.
- flatMap: Similaire à map, mais peut produire un stream de zéro, un ou plusieurs éléments pour chaque élément d'entrée.
- forEach : Applique une action à chaque élément du stream.
- reduce : Effectue une réduction sur les éléments du stream en utilisant un accumulateur.
- count : Retourne le nombre d'éléments dans le stream.
- Collect : Elle est utilisée pour transformer les éléments d'un stream en une autre forme comme les listes.

# Exemple d’utilisation de stream

Exemple4.java

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Exemple4 {
    public static void main(String[] args) {
        List <String> list= new ArrayList<>();
        list.add("Exemple");
        list.add("Java");
        list.add("String");
        list.add("Stream");

        //Obtenir une liste qui contient les chaînes qui commencent par "S"
        List<String> listS= list.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());
        System.out.println(listS);
        //Obtenir une liste ou chaque chaîne en majuscules
        List<String> listMaj=list.stream().map(String::toUpperCase).collect(Collectors.toList());
        System.out.println(listMaj);
        //Diviser chaque chaîne en caractères et les concaténer
        String concat=list.stream().flatMap(s-> Arrays.stream(s.split( regex: ""))).reduce( identity: "",String::concat);
        System.out.println(concat);
        //compter le nombre de chaine
        long count=list.stream().count();
        System.out.println("Nombre de chaînes : " + count);



    }
}
```