

**Université Sultan Moulay Slimane**

**Beni-Mellal**



**École nationale des sciences appliquées**

**Khouribga**



# COMPTE RENDU :

## TP6 JAVA POO

Filière : Informatique et Ingénierie des données (iid1)

Réalisé par :

- LAHMAMA Fatima-Zahraa
- EL FATHI Zakaria

Encadré par :

- Mr. GHERABI Noredline

## OBJECTIFS :

Manipulation des collections (HashSet, TreeSet, LinkedHashSet)

### Exercice 1 :

Un programme java qui permet de créer une liste de type « TreeSet » et ajout des éléments couleur de type chaîne puis affichage en utilisant l'itérateur :

*Code :*

```
package tp6;

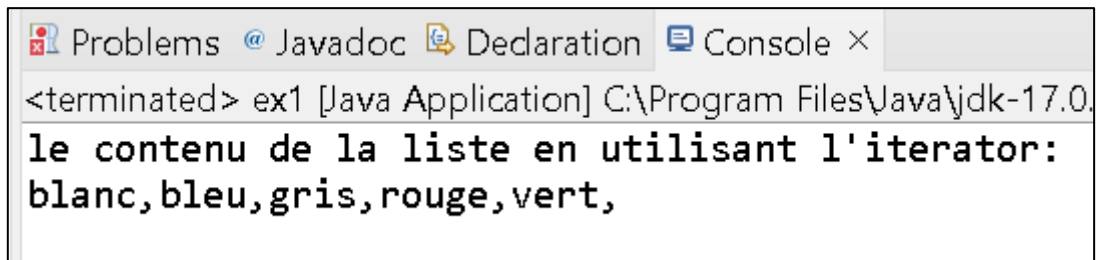
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;

public class ex1 {

    public static void main(String[] args) {
        //instanciation de l'objet Couleur de type TreeSet()
        TreeSet couleurs=new TreeSet();
        //Ajout des element couleur dans la liste
        couleurs.add("bleu");
        couleurs.add("vert");
        couleurs.add("rouge");
        couleurs.add("gris");
        couleurs.add("blanc");
        //parcours des elements en utilisant l'iterator
        Iterator<String> it = couleurs.iterator();

        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

*Exécution :*



```
<terminated> ex1 [Java Application] C:\Program Files\Java\jdk-17.0.  
le contenu de la liste en utilisant l'iterator:  
blanc,bleu,gris,rouge,vert,
```

## Exercice 2 :

Un programme java qui ajoute tous les éléments d'une liste de type « TreeSet » dans une autre de type « HashSet » et suppression de la 1ere liste:

*Code :*

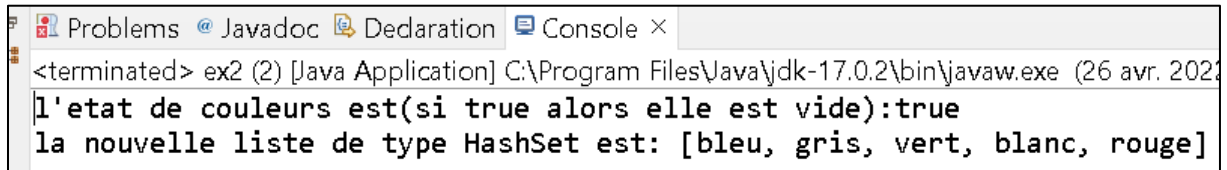
```
package tp6;  
  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.Set;  
import java.util.TreeSet;  
  
public class ex2 {  
  
    public static void main(String[] args) {  
  
        // remplissage de treeSet!  
        TreeSet<String> couleurs=new TreeSet();  
  
        couleurs.add("bleu");  
        couleurs.add("vert");  
        couleurs.add("rouge");  
        couleurs.add("gris");  
        couleurs.add("blanc");  
  
        HashSet<String> hashset1 = new HashSet<>();  
        hashset1.addAll(couleurs);  
        // suppression de la 1ere liste  
        couleurs.clear();  
  
        // on verifier que couleurs est vide!  
        System.out.println("l'etat de couleurs est(si true alors elle est  
vide):"+couleurs.isEmpty());
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
// on affiche les elements de HashSet!  
    System.out.println("la nouvelle liste de type HashSet est: "+hashset1);  
}  
}
```

*Exécution :*



```
<terminated> ex2 (2) [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (26 avr. 2022)  
l'etat de couleurs est(si true alors elle est vide):true  
la nouvelle liste de type HashSet est: [bleu, gris, vert, blanc, rouge]
```

### Exercice 3 :

Un programme java qui ajoute tous les éléments d'une liste de type « TreeSet » dans une autre de type « HashSet » et suppression de la 1ere liste:

*Code :*

```
package tp6;  
  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.Set;  
import java.util.TreeSet;  
  
public class ex3 {  
  
    public static void main(String[] args) {  
  
        Set<Integer> entierInitial=new HashSet();  
        entierInitial.add(2);  
        entierInitial.add(7);  
        entierInitial.add(9);  
        entierInitial.add(88);  
        entierInitial.add(101);  
  
        Set<Integer> entierFinal=new TreeSet();  
        Iterator<Integer> it = entierInitial.iterator();  
        while (it.hasNext()) {  
            int n=it.next();  

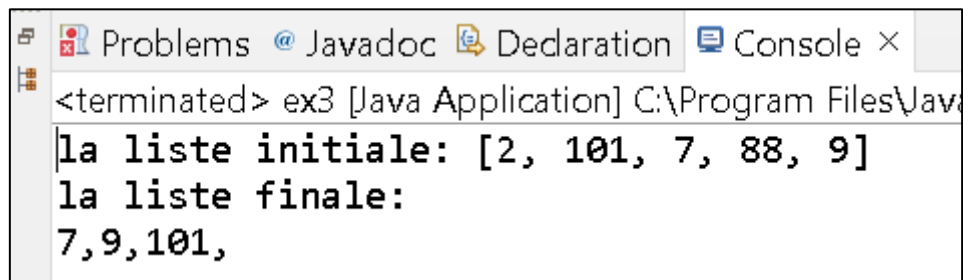
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
        if (n %2 !=0) {
            entierFinal.add(n);
        }
    }
    System.out.println("la liste initiale: "+entierInitial );
    System.out.println("la liste finale: ");
    Iterator <Integer> it2= entierFinal.iterator();
    while(it2.hasNext()) {
        System.out.print(it2.next()+"");
    }
}
```

*Exécution :*



```
<terminated> ex3 [Java Application] C:\Program Files\Java\
la liste initiale: [2, 101, 7, 88, 9]
la liste finale:
7,9,101,
```

### Exercice 4 :

Un programme java contenant une liste « HashSet » qui contient les objets d'une classe **Voiture** :

Donc on commence tout d'abord par créer une la classe Voiture avec ses propres attributs comme suit :

*Code:*

```
package tp6;
public class Voiture implements Comparable {
    String marque,matricule;
    int puissance;
    public Voiture(String marque,String matricule, int puissance) {
        this.marque=marque;
        this.matricule=matricule;
        this.puissance=puissance;}

    // la methode d'affichage toString()
    @Override
    public String toString() {
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
        return "Voiture [marque=" + marque + ", matricule=" + matricule + ",  
puissance=" + puissance + "]\n";  
    }  
    // Getter & Setters de la classe  
    public String getMarque() {  
        return marque;  
    }  
    public void setMarque(String marque) {  
        this.marque = marque;  
    }  
  
    public String getMatricule() {  
        return matricule;  
    }  
  
    public void setMatricule(String matricule) {  
        this.matricule = matricule;  
    }  
  
    public int getPuissance() {  
        return puissance;  
    }  
  
    public void setPuissance(int puissance) {  
        this.puissance = puissance;  
    }  
}
```

Après on revient à une classe principale(main) pour instancier des objets de la classe **Voiture**, comme suit :

*Code :*

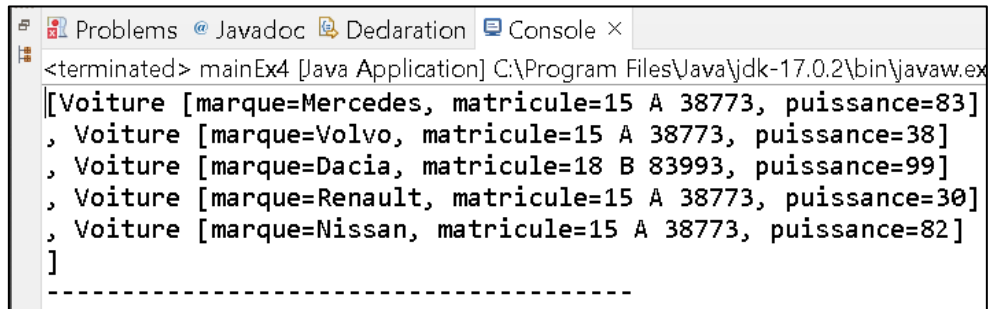
```
package tp6;  
  
import java.util.Collections;  
import java.util.HashSet;  
  
public class mainEx4 {  
  
    public static void main(String[] args) {  
        // instantiation d'un HashSet de nom voiture  
        HashSet <Voiture> voiture=new HashSet<Voiture>();  
        // ajout de qlq elements  
        voiture.add(new Voiture("Nissan","15 A 38773",82));  
        voiture.add(new Voiture("Dacia","18 B 83993",99));  
        voiture.add(new Voiture("Mercedes","15 A 38773",83));  
        voiture.add(new Voiture("Volvo","15 A 38773",38));  
    }  
}
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
voiture.add(new Voiture("Renault","15 A 38773",30));  
// affichage de la liste 'voiture'  
System.out.println(voiture);  
System.out.println("-----");}
```

*Exécution :*



```
<terminated> mainEx4 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe  
[Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]  
 , Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]  
 , Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]  
 , Voiture [marque=Renault, matricule=15 A 38773, puissance=30]  
 , Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]  
 ]  
-----
```

Maintenant on copie la liste dans une autre de type « TreeSet » donc dans la même classe principale on écrit :

*Code :*

```
//copie  
  
TreeSet<Voiture> voiture2=new TreeSet<Voiture>(voiture);  
System.out.println("la liste TreeSet resultante:"+ voiture2);
```

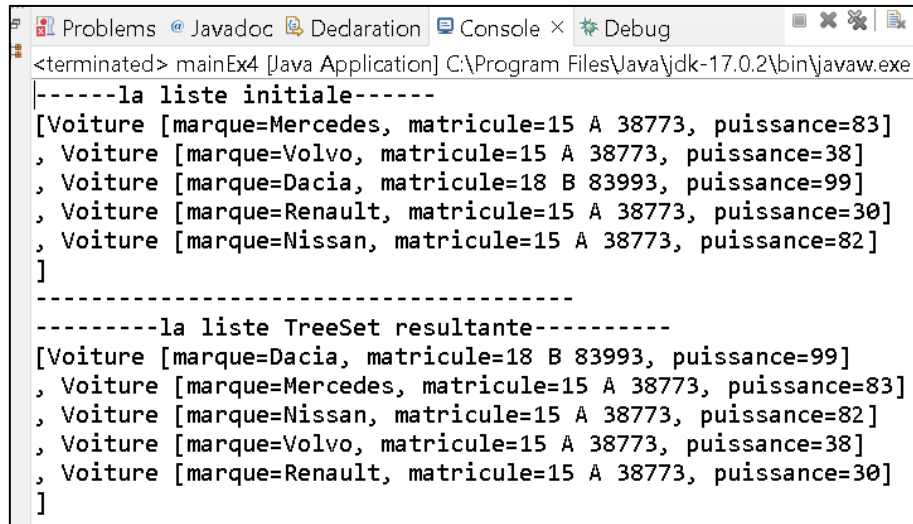
Mais tout d'abord une implémentation des **Comparable** redéfinition de la méthode **compareTo** est nécessaire dans la classe **Voiture**

```
package tp6;  
  
public class Voiture implements Comparable <Voiture> {  
    -----  
    ----  
    ---  
    -  
  
    @Override  
    public int compareTo(Voiture v) {  
        if (this.getPuissance() < v.getPuissance()) {  
            return 1;  
        }else {  
            return -1;  
        }  
    }  
}
```

*Exécution :*

## TRAVAUX PRATIQUES 6

### – JAVA POO



```
<terminated> mainEx4 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe
-----la liste initiale-----
[Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
, Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
, Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
, Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
, Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
]
-----
-----la liste TreeSet resultante-----
[Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
, Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
, Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
, Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
, Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
]
```

Affichage des deux listes avec le temps d'exécution en nanosecondes :

*Code :*

```
//le temps
long avantH, apresH,avantT, apresT;
avantH=System.nanoTime();
for(Voiture i : voiture) {
    System.out.println(i);
}
apresH=System.nanoTime();
System.out.println("le temps est :"+ (apresH-avantH));

avantT=System.nanoTime();
for(Voiture i : voiture2) {
    System.out.println(i);
}
apresT=System.nanoTime();
System.out.println("le temps est :"+ (apresT-avantT));

System.out.println("-----");
```

*Exécution :*



## TRAVAUX PRATIQUES 6

### – JAVA POO

```
Problems @ Javadoc Declaration Console × Debug
<terminated> mainEx4 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe
-----
Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
le temps est :614700
Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
le temps est :581100
-----
```

**ON REMARQUE** que la liste de type « HashSet » prend beaucoup plus de temps d'exécution que celle de type « TreeSet »

Maintenant on met les objets des deux listes dans une 3eme de type « LinkedHashSet » :

*Code :*

```
// ajouter leus deux list dans une 3eme
LinkedHashSet <Voiture> voiture3=new LinkedHashSet(voiture2);
voiture3.addAll(voiture);
System.out.println(voiture3);
System.out.println("-----");
```

*Exécution :*

```
Problems @ Javadoc Declaration Console × Debug
<terminated> mainEx4 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (2)
-----
[Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
, Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
, Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
, Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
, Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
]
-----
```

## TRAVAUX PRATIQUES 6

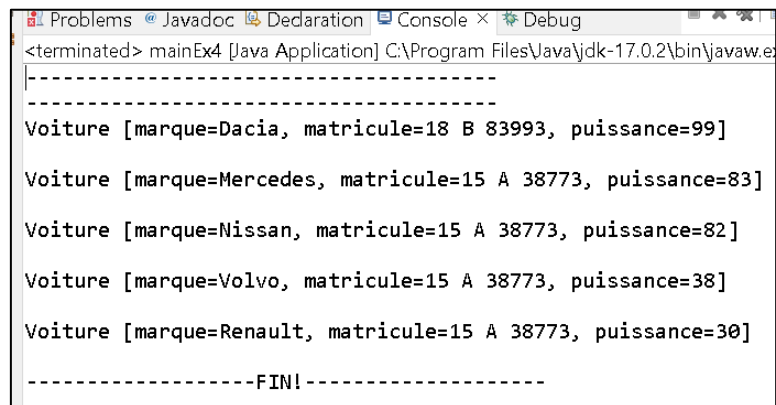
### – JAVA POO

Maintenant on arrive à transformer la liste obtenue en un tableau de objets de type **Voiture** :

*Code :*

```
//transformer la liste en tableau
Object[] tab = voiture3.toArray();
for(int i=0;i<tab.length;i++) {
    System.out.println(tab[i]);
}
System.out.println("-----FIN!-----");
```

*Exécution :*



```
Problems Javadoc Declaration Console × Debug
<terminated> mainEx4 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe
-----
Voiture [marque=Dacia, matricule=18 B 83993, puissance=99]
Voiture [marque=Mercedes, matricule=15 A 38773, puissance=83]
Voiture [marque=Nissan, matricule=15 A 38773, puissance=82]
Voiture [marque=Volvo, matricule=15 A 38773, puissance=38]
Voiture [marque=Renault, matricule=15 A 38773, puissance=30]
-----FIN!-----
```

## Exercice 5 :

Pour la raison de copier les éléments d'un tableau des **Personnes** et les manipuler dans des collections

On doit tout d'abord créer une la classe '**Personne**' avec ses propres attributs comme suit :

*Code :*

```
package tp6;

public class Personne {

    int code,age;
    String nom;
    //constructeur avec parametres pour initialiser les attributs
    public Personne(int code, int age,String nom) {
        this.code=code;
        this.nom=nom;
        this.age=age;
    }
}
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
// Getters & Setters
public int getCode() {
    return code;
}

public void setCode(int code) {
    this.code = code;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getNom() {
    return nom;
}

public void setNom(String nom) {
    this.nom = nom;
}
// redéfinition de la méthode d'affichage
@Override
public String toString() {
    return "Personne [code=" + code + ", age=" + age + ", nom=" + nom + "];"
}
}
```

Maintenant on peut commencer par la 1ère question :

1. Extraction des éléments d'un tableau de type **Personne** selon la condition suivante :
  - Age > 20 : le nom de la personne est ajouté à une *ArrayList* appelée « ListeNom »
  - Sinon : on garde dans une HashTable appelée « MapPersonne » le code et le nom de la personne

Donc la partie code sera comme suit : `package tp6;`

```
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Map;
public class mainEx5 {
    public static void main(String[] args) {
        ArrayList <Personne> pers=new ArrayList();
```

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
// Remplissage de la liste « pers »
pers.add(new Personne(112,16,"rabie"));
pers.add(new Personne(442,32,"med"));
pers.add(new Personne(921,11,"hmama"));
pers.add(new Personne(212,55,"zack"));
pers.add(new Personne(726,18,"zaid"));
System.out.println("les element du ArrayList sont:"+pers);
System.out.println("-----");

//instanciation de ArrayList et HashTABLE
ArrayList<String> ListeNom=new ArrayList();
Hashtable<Integer,String> MapPersonne =new Hashtable();

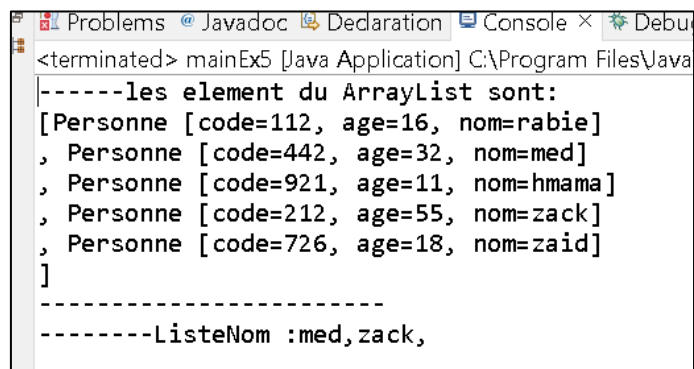
for(int i=0;i<pers.size();i++) {
    if((pers.get(i)).age>=20) {
        ListeNom.add((pers.get(i)).nom);
    }
    else {
        MapPersonne.put((pers.get(i)).code,pers.get(i).nom);
    }
}
}
```

2. Affichage, à l'aide d'un iterator, des éléments de « ListeNom »

```
// Q2

System.out.println("-----ListeNom :");
Iterator <String> it= ListeNom.iterator();
while(it.hasNext()) {
    System.out.println(it.next());
}
```

*Exécution :*



```
<terminated> mainEx5 [Java Application] C:\Program Files\Java
-----les element du ArrayList sont:
[Personne [code=112, age=16, nom=rabie]
, Personne [code=442, age=32, nom=med]
, Personne [code=921, age=11, nom=hmama]
, Personne [code=212, age=55, nom=zack]
, Personne [code=726, age=18, nom=zaid]
]
-----
-----ListeNom :med,zack,
```

3. Affichage des valeurs de la liste « MapPersonne », en utilisant Enumérateur :

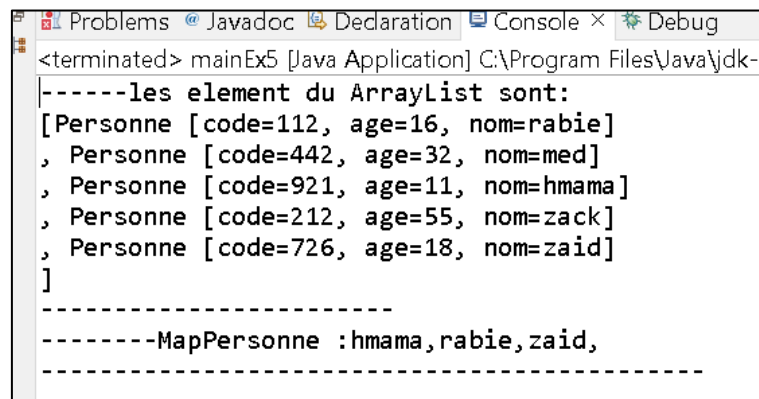
*Code :*

## TRAVAUX PRATIQUES 6

### – JAVA POO

```
//Q3
Enumeration MapPers;
MapPers = MapPersonne.elements();
System.out.print("-----MapPersonne :" );
while (MapPers.hasMoreElements()) {
    System.out.print(MapPers.nextElement()+" ,");
}
System.out.println("\n-----");
```

*Exécution :*



```
<terminated> mainEx5 [Java Application] C:\Program Files\Java\jdk-
-----les element du ArrayList sont:
[Personne [code=112, age=16, nom=rabie]
 , Personne [code=442, age=32, nom=med]
 , Personne [code=921, age=11, nom=hmama]
 , Personne [code=212, age=55, nom=zack]
 , Personne [code=726, age=18, nom=zaid]
]
-----
-----MapPersonne :hmama,rabie,zaid,
-----
```

4. Dans la liste « MapPersonne » on cherche si un code donne existe ou non :

*Code :*

```
Scanner sc=new Scanner(System.in);
System.out.println("-----entrer le code a rechercher:  ");
int tempCode= sc.nextInt();
boolean existe=false;
for(int i=0;i<MapPersonne.size();i++) {
    if(MapPersonne.containsKey(tempCode)) existe=true;
}
if (existe==true) System.out.println("-----le code est
bien existant!-----");
else System.out.println("-----le code n'est pas
existent!-----");

System.out.println("-----
-----");

/*
```

*Exécution :*

TRAVAUX PRATIQUES 6  
– JAVA POO

```

Problems @ Javadoc Declaration Console × Debug
<terminated> mainEx5 [Java Application] C:\Program Files\Java\jdk-1
-----les element du ArrayList sont:
[Personne [code=112, age=16, nom=rabie]
, Personne [code=442, age=32, nom=med]
, Personne [code=921, age=11, nom=hmama]
, Personne [code=212, age=55, nom=zack]
, Personne [code=726, age=18, nom=zaid]
]
-----
-----MapPersonne :hmama,rabie,zaid,
-----
-----entrer le code a rechercher:
921
|-----le code est bien existant!-----
-----

```

5. Dans la même liste « MapPersonne » on recherche si un nom existe ou non !

*Code :*

```

System.out.println("entrer le nom a rechercher: ");
String tempNom= sc.next();
boolean exist=false;
    for(int i=0;i<MapPersonne.size();i++) {
        if(MapPersonne.containsValue(tempNom)) exist=true;
    }
if(exist=true) System.out.println("-----NOM EXISTANT!-----");
else System.out.println("----- Ce nom n'existe pas-----");
System.out.println("-----");

```

*Exécution:*

```

Problems @ Javadoc Declaration Console × Debug
<terminated> mainEx5 [Java Application] C:\Program Files\Java\jdk-1
-----les element du ArrayList sont:
[Personne [code=112, age=16, nom=rabie]
, Personne [code=442, age=32, nom=med]
, Personne [code=921, age=11, nom=hmama]
, Personne [code=212, age=55, nom=zack]
, Personne [code=726, age=18, nom=zaid]
]
-----
-----MapPersonne :hmama,rabie,zaid,
-----
entrer le nom a rechercher:
hmama
|-----NOM EXISTANT!-----
-----

```

– JAVA POO

6. Création d'une nouvelle liste de type `HashMap` de nom « `CopiePersonne` » contenant une copie de la liste « `MapPersonne` » :

*Code:*

```
//Q6
Map<Integer,String> CopiePersonne =new HashMap();
CopiePersonne.putAll(MapPersonne);
System.out.println("-----CopiePersonne:"+CopiePersonne);
```

*Exécution :*

```

<terminated> mainEx5 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\
-----les element du ArrayList sont:
[Personne [code=112, age=16, nom=rabie]
, Personne [code=442, age=32, nom=med]
, Personne [code=921, age=11, nom=hmama]
, Personne [code=212, age=55, nom=zack]
, Personne [code=726, age=18, nom=zaid]
]
-----
-----MapPersonne :hmama,rabie,zaid,
-----
-----CopiePersonne:{112=rabie, 921=hmama, 726=zaid}

```

7. En utilisant une méthode `entrySet ()` on affiche les éléments de la liste « CopiePersonne » :

*Code :*

```
System.out.println("-----affichage En utilisant entrySet()-----");  
    for(Map.Entry<Integer,String> e:CopiePersonne.entrySet()) {  
        System.out.println("Cle:"+e.getKey()+"\t\t\t\t\t\t  
Valeur:"+e.getValue());  
    }
```

*Exécution :*

```

<terminated> mainEx5 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe
|-----les element du ArrayList sont:
|[Personne [code=112, age=16, nom=rabie]
|, Personne [code=442, age=32, nom=med]
|, Personne [code=921, age=11, nom=hmama]
|, Personne [code=212, age=55, nom=zack]
|, Personne [code=726, age=18, nom=zaid]
|]
|-----
|-----MapPersonne :hmama,rabie,zaid,
|-----
|-----affichage En utilisant entrySet()-----
Cle:112 Valeur:rabie
Cle:921 Valeur:hmama
Cle:726 Valeur:zaid

```

## TRAVAUX PRATIQUES 6

### – JAVA POO

8. Test de performance d'affichage en termes de temps entre les deux listes « CopiePersonne » et « MapPersonne » :

*Code :*

```
//Q8
double t2 = System.nanoTime();
System.out.println("-----CopiePersonne-----\n"+CopiePersonne);
double t1 = System.nanoTime();
System.out.println("===CopiePersonne took: "+(t1-t2)+"===");

double a2 = System.nanoTime();
System.out.println("-----MapPersonne-----\n"+CopiePersonne);
double a1 = System.nanoTime();
System.out.println("===MapPersonne took: "+(a1-a2)+"===");
```

*Exécution :*

```
-----CopiePersonne-----
{112=rabie, 921=hmama, 726=zaid}
===CopiePersonne took: 34300.0===
-----MapPersonne-----
{112=rabie, 921=hmama, 726=zaid}
===MapPersonne took: 52400.0===
```

# Fin !