

Table associative, Method générique, Wildcard et Vue.

Exercice 1 - Générification

- 1 Générifier le code ci-dessous, en transformant Pair en Pair<U,V>.

```
public class Pair {
    private final Object first;
    private final Object second;

    public Pair(Object first, Object second) {
        this.first=first;
        this.second=second;
    }
    public Object getFirst() {
        return first;
    }
    public Object getSecond() {
        return second;
    }
    public static void main(String[] args) {
        Pair p1=new Pair("toto","titi");
        Pair p2=new Pair(2,p1);

        Pair p3=(Pair)p2.getSecond();
    }
}
```

Exercice 2 - Association et contrat sur les Objets

Qu'affiche le code ci-dessous ?

```
Map<Pair<String,String>,String> map=new
HashMap<Pair<String,String>,String>();
map.put(new Pair<String,String>("jean-paul","sartre"),"mort");
map.put(new Pair<String,String>("elvis","presley"),"vivant");

map.remove(new Pair<String,String>("elvis","presley"));

System.out.println(map);
```

Que doit-on faire pour résoudre le problème ?

Exercice 3 - Pair et Wildcard

- 1 Rappeler pourquoi le code suivant ne compile pas :

```
public static void main(String[] args) {
    ArrayList<String> l1=new ArrayList<String>();
    ArrayList<Object> l2=l1;
}
```

- 2 Changer le type de l2 pour que le code suivant compile :

```
public static void main(String[] args) {
    ArrayList<String> l1=new ArrayList<String>();
    ArrayList<Object> l2=l1;
}
```

```

        for(Object o:l2)
            System.out.println(o);
    }

```

- 3 En utilisant la classe Pair, générer ensuite le code suivant :

```

public static void main(String[] args) {
    Pair p1=new Pair("toto","titi");
    Pair p2=new Pair(2,p1);

    Pair p4;
    if (args.length%2==0)
        p4=p1;
    else
        p4=p2;

    Comparable c=(Comparable)p4.getFirst();
}

```

Exercice 4 - ? extends c'est super, non ?

- 1 Ecrire la méthode copy pour que le code suivant fonctionne :

```

public static void main(String[] args) {
    List<String> l1=Arrays.asList(args);
    List<String> l2=new ArrayList<String>();
    copy(l2,l1);
}

```

- 2 Vérifier que l'exemple suivant compile aussi :

```

public static void main(String[] args) {
    List<Integer> l1=Arrays.asList(2,3);
    List<Number> l2=new ArrayList<Number>();
    copy(l2,l1);
}

```

- 3 Ecrire une méthode fill qui initialise une liste avec une valeur unique

```

public static void main(String[] args) {
    List<String> l1=Arrays.asList(args);
    fill(l1,"toto");
}

```

Exercice 5 - Ensemble et Bag

- 1 Écrire un programme indiquant quels sont les mots qui se trouvent sur la ligne de commande, on affichera les doublons une unique fois.

```

java Unique toto tutu toto titi tutu

```

Dans un premiers temps, afficher "toto", "titi" et "tutu" dans n'importe quel ordre.

Ensuite, afficher "toto", "tutu" "titi" dans cet ordre, c-a-d l'ordre d'insertion.

Enfin, afficher "titi", "toto", "tutu" dans cet ordre, c-a-d l'ordre lexicographique.

- 2 Écrire un second programme pour qu'il compte le nombre de fois qu'un mot apparait sur sa ligne de commande et affiche chaque mot suivi de son nombre d'occurrence.

N.B : penser à faire des fonctions assez générales et si possible utilisables avec autre

chose que des String.