esprit
Se former autrement

Conception Orientée Objet & Programmation JAVA

Chapitre 2 : Classe et Objet (partie2)

PLAN

- Introduction
- **Classe et objet**
- Encapsulation
- Héritage
- Polymorphisme
- Exceptions
- Interfaces
- Collection
- Interface Fonctionnelle
- Expression Lambda
- Stream

Objectifs du chapitre

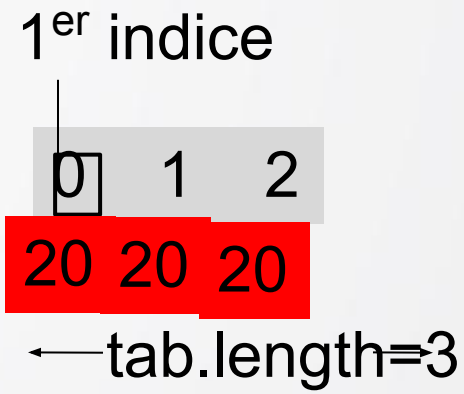
- ✔ Notion de référence
- ✔ Manipulation des tableaux
- ✔ Association entre les classes

Manipulation des tableaux

- Un tableau est une structure de donnée ayant un ensemble d'élements qui sont tous du même type (type primitifs ou classe)
- On utilise le symbole [ ] pour définir un tableau

Exemple :

```
int[] tab;
tab = new int[3];
tab[0] = 10; // initialiser le premier élément
tab[1] = 20; // initialiser le second élément
```



Manipulation des tableaux

Un tableau peut être initialisé :
int ti1 [] = { 1, 2, 3 , 4}; **char**[] tc = {’a’, ’b’, ’c’};

Pour allouer l’espace nécessaire au tableau il faut utiliser new :

- int tab1 = new int [10]; // tab1 est un tableau à une dimension de 10
- char tab2= new char [15]; // ti est un tableau à une dimension de 15

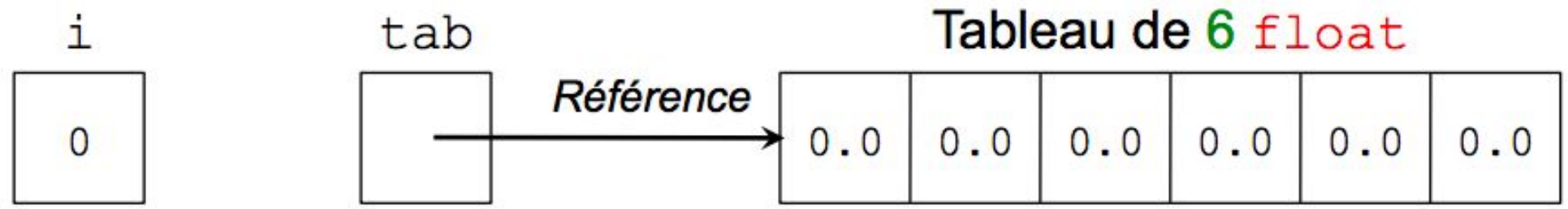
Accès à la taille du tableau avec **tab.length**

- Accès à un élément avec tab[indice]

Exemple : tab[i] = tab[j] * 2;
Attention : les éléments sont indexés à partir de 0
⇒ un tableau possède les éléments allant de 0 à tab.length-1

Accès aux tableaux

```
public static void main(String[] args) {
   float[] tab = new float[6];
   for(int i=0; i<tab.length; i++) {
      tab[i] = i + 2;
   }
}
```



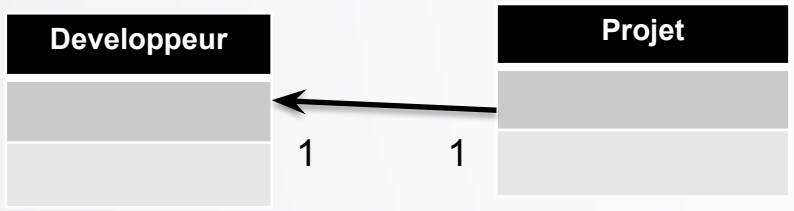
Les associations entre les classes

Les associations

- Une association est une relation entre deux classes (association binaire) ou plus (association n-aire), qui décrit les connexions structurelles entre leurs instances.
- Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.
- Il y a plusieurs type d’association:
  - Association one-to-one
  - Association one-to-many
  - Association many-to-many

Les associations

- **Association one-to-one undirectionnelle**



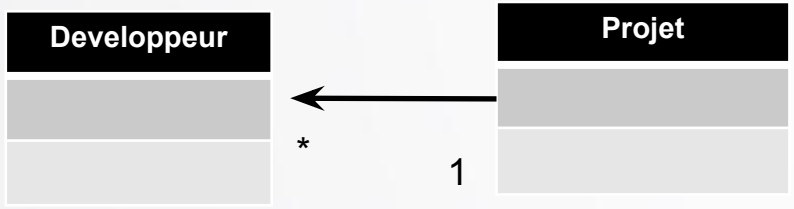
Mapping en java

```
class Developpeur {
        public void setProjet (Projet projet) {
                this. projet=projet;
        }
        public Projet getProjet () {
                return projet;
        }
}
```

```
class Projet{
        public Developpeur developpeur;
        public void setDeveloppeur (Developpeur developpeur) {
                this.developpeus=developpeur;
        }
        public Developpeur getDeveloppeur () {
                return developpeur;     }
}
```

Les associations

- **Association one-to-many unidirectionnelle**



Mapping en java

```
class Developpeur {
        public void setProjet (Projet projet) {
                this. projet=projet;
        }
        public Projet getProjet () {
                return projet;
        }
}
```

```
class Projet{
        public Developpeur [] developpeurs;
        public void setDeveloppeur (Developpeur[] developpeurs) {
                this.developpeurs=developpeurs;
        }
        public Developpeur [] getDeveloppeur () {
                return developpeurs;   }
}
```

Mot clé : static

- **static :** est un modificateur de comportement et non d'accès (comme public, private,…)

- il peut être appliqué sur une variable ou bien une méthode

- Un attribut/variable ou une méthode statique (déclaré avec le mot **static**) est dit **attribut de classe** ou **méthode de classe**.

- Une variable statique ou une méthode statique est partagée par toutes les instances de la classe.

Mot clé : static (exemple)

```
public class Chien {
int id ;
String race ;
int nbChien ;

public Chien () {
    nbChien ++;
  }

public Chien (int id , String race ) {
    nbChien ++;
  }
}
```

```
class Test{
   public static void main(String[] args){

     Chien c1=new Chien (123, « berger »);
     Chien c2=new Chien (321, « caniche »);


     System.out.println(c1. nbChien );
     System.out.println(c2. nbChien );
   }
}
```

C1 : 123, « berger »

C2 321, « caniche »

Console :
1
1

Mot clé : static (exemple)

```
public class Chien {
int id ;
String race ;
int static nbChien ;

public Chien () {
    nbChien ++;
  }

public Chien (int id , String race ) {
    nbChien ++;
  }
}
```

```
class Test{
   public static void main(String[] args){

     Chien c1=new Chien (123, « berger »);
     Chien c2=new Chien (321, « caniche »);


     System.out.println(c1. nbChien );
     System.out.println(c2. nbChien );
   }
}
```

C1 : 123, « berger »

C2 321, « caniche »

Console :
2
2

Mot clé : Static (exemple)

C1 :

id : 123
Race : berger
nbChien : 1

Mémoire :
id : 0
Race : null
nbChien : 1

C2 :

id : 321
Race : caniche
nbChien : 2

- **nbChien :** n'appartient pas à une instance particulière, elle appartient à la classe.
- est partagée par toutes les instances de la classe

Variable static final

- Une variable déclarée static **final** ne change plus de valeur une fois initialisée.
- si elle est aussi publique, la variable est utilisable partout.

- une variable publique, statique et finale est une constante.

- Par convention, elle est notée en majuscule, un blanc souligné séparant les mots.

Variable Static final(exemple)

```
class MesConstantes {
public static final double PI_APPROX = 3.1415;
}
```

```
class Test{
    public static void main(String[] args){
int i = 2 * MesConstantes . PI_APPROX ;


}
}
```

Méthodes static

- Bien que Java soit un langage objet, il existe des cas où une instance de classe est inutile.
- Le mot clé static permet alors à une méthode de s'exécuter sans avoir à instancier la classe qui la contient. L'appel à une méthode statique se fait alors en utilisant le nom de la classe, plutôt que le nom de l'objet
- Le comportement d’une méthode statique ne dépend pas de la valeur des variables d'instance

Méthodes static (exemple)

```
public class Compteur {
public static int compteur = 0;

    public void calculCompteur() {     compteur++;   }
}
```

```
public class Compteur {
public static int compteur = 0;

    public static void calculCompteur() {     compteur++;   }
}
```

Méthodes static (exemple)

```
public class Compteur {
public static int compteur = 0;

    public static void  calculCompteur() {     compteur++;   } }
```

```
class Test {
public static void main ( String [] args ){
        System.out.println(Compteur. calculCompteur() );
}
}
```

l'appel d'une méthode statique est fait a travers le nom de la classe

Merci pour votre attention