Mis-a-niveau-JAVA

Hossein Khani

Dauphine
UNIVERSITÉ PARIS

Exercice

- Pour ecrire un logiciel bancaire, il faudra représenter dans notre langage de programmation l'ensemble des informations caractérisant un compte et coder les actions qui s'effectuent sur les comptes (retrait, dépot). L' état d'un compte, pourra être défini par son numéro, le nom de son titulaire, son solde ; son comportement est caractérisé par les opérations de dépot, de retrait et d'affichage du solde.

1. Ecrivez la classe Compte().
2. Ecrivez une méthode afficher(), pour afficher le solde.
3. Codez les méthodes retirer() et déposer().

Exercice

- Considérons une classe Personne par la donnée des deux variables d'instances, l'une contenant la date de naissance de la personne et l'autre son nom. La date de naissance n'est pas un type pré défini. Il faut donc aussi définir une classe Date.

1. Ecrivez la classe Personne().
2. Ecrivez la classe Date().

Exercice

- On propose de pouvoir comparer des objets de différentes classes au moyen d'une conversion vers les nombres entiers. Pour cela on va utiliser une interface avec la méthode de conversion.

```
interface Convertible{
int toInt();
}
```

1. Modifiez les classes Compte() et Date() vues en cours pourqu'ells implémentent cette interface.

2. Ecrivez une classe proposant des méthodes statiques pour comparer deux objets convertibles : une pour le test plus grand strict, une pour le test plus petit strict, une pour le test d'égalité, en comparant les entiers obtenus par conversion.

3. Ajoutez à la classe précédente une méthode statique permettant de trier en ordre croissant un tableau d'objets convertibles.

Polymorphisme

```java
public class Machine {
    public void start() {
        System.out.println("Machine Start");
    }

    public void stop() {
        System.out.println("Machine Stop");
    }


}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Machine();
        machine1.start();
        machine1.stop();
    }
}
```

Output:

Machine Start

Machine Stop

Polymorphism

```java
public class Machine {
    public void start() {
        System.out.println("Machine Start");
    }

    public void stop() {
        System.out.println("Machine Stop");
    }

}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Car();
        machine1.start();
        machine1.stop();
    }
}
```

Output:

Car Start

Car Stop

Polymorphism

```java
public class Machine {
      public void start() {
          System.out.println("Machine Start");
      }

      public void stop() {
          System.out.println("Machine Stop");
      }

}
```

```java
public class Car extends Machine {

    @Override
    public void start() {
        System.out.println("Car Start");
    }

    @Override
    public void stop() {
        System.out.println("Car Stop");

    }

    public void windShield() {
        System.out.println("Windshield Start");
    }

}
```

```java
public class Main {

    public static void main(String[] args) {
        Machine machine1 = new Car();
        machine1.start();
        machine1.stop();
        machine1.windshield;
    }
}
```

Erreur de Compilation

Classe Abstraire

- Une classe abstraite est une classe qui est déclarée avec le mot clé abstract,
- elle peut inclure ou non des méthodes abstraites.
- Les classes abstraites ne peuvent pas être instanciées, mais elles peuvent être sous-classées.

- Une méthode abstraite est une méthode qui est déclarée sans implémentation (sans accolades et suivie d'un point-virgule), comme ceci :

abstract TypeDeTetour nom(les paramètres);

Example:

```
public abstract class GraphicObject{
    int x;
    int y;
 void Tomove(){
\\ implementation
}

abstract void draw();
}
```

Interface et Classe Abstraire

```java
public abstract class GraphicObject {

    String name;
    int x;
    int y;

    void Tomove(int dx , int dy) {
        this.x = this.x +dx;
        this.y = this.y +dy;
    }
    abstract void  draw();

}
```

Question: Si GraphicObject est une interface ?

```java
public class Rectangle extends GraphicObject {
    public Rectangle(String name , int x , int y) {
        this.name = name;
        this.x = x;
        this.y = y;
    }
    @Override
    public void draw() {
        System.out.println("drawing a Rectangle!!");

    }
```

```java
public class Main {
      public static void main (String[] args) {
          Rectangle r = new Rectangle("rec1", 2,3);
          r.Tomove(3, 2);
          System.out.println(r.x);
          System.out.println(r.y);


      }
}
```

Collection

Pourqoui Collection?

Aller au-delà des limites des tableaux :

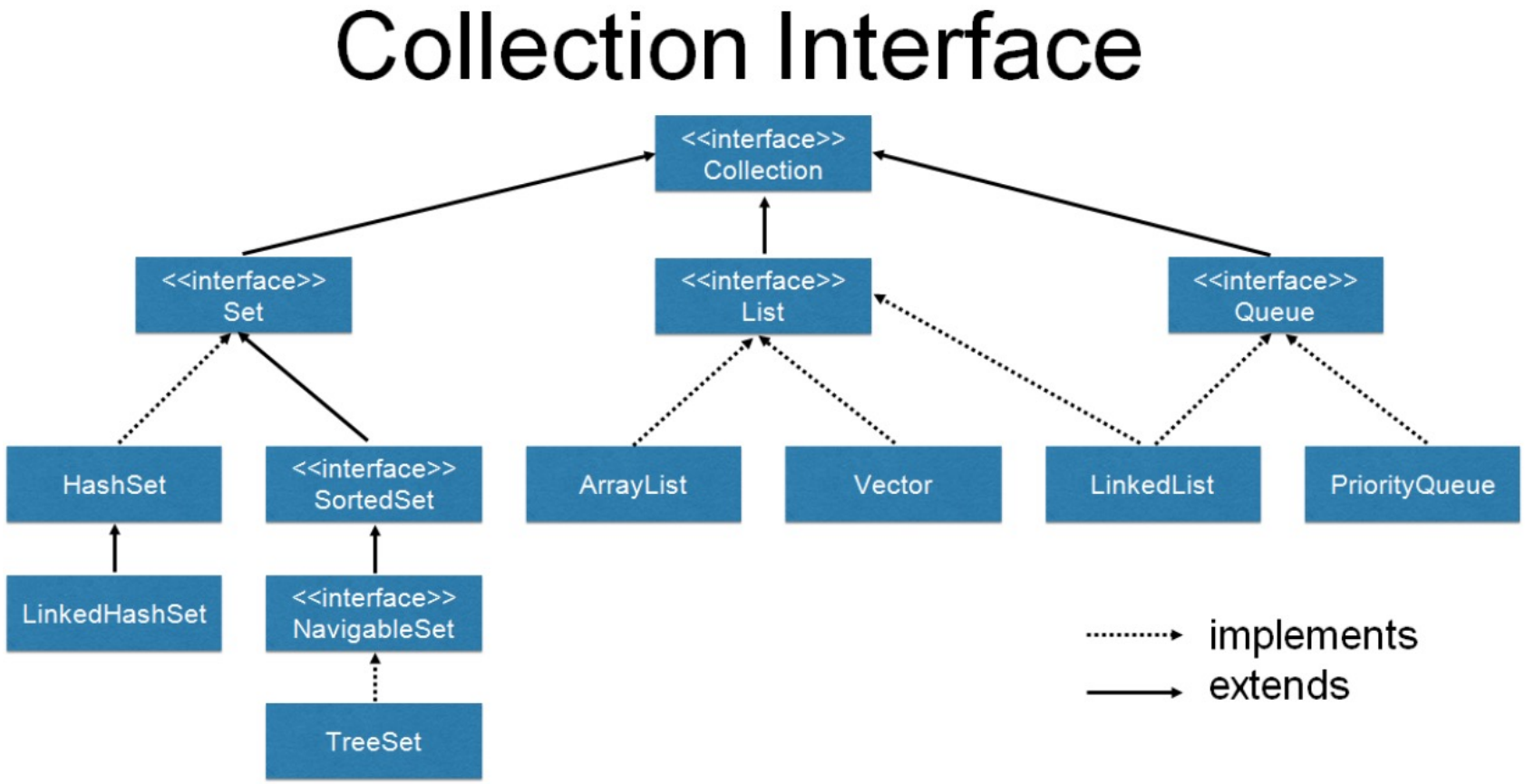
- Ils ont une **taille fixe** ;
- vous ne pouvez modifier que les valeurs **existantes**.

Imaginez une liste des animaux: renard, hibou, koala et loutre

Comment on peut ajouter écureuil entre le renard et le hibou?

Comment on peut supprimer le hibou?

Hiérarchie de collection



Collection ordonnée

Avec une interface List, on peut:

- **accéder** à chaque élément via son index ;
- **ajouter** (*append*) un nouvel élément à la fin ;
- **insérer** un nouvel élément à un index spécifique ;
- **supprimer** un nouvel élément à un index spécifique.

Collection ordonnée

- La Collection le plus connue Arraylist.
- Déclarer une variable dont le type est l'interface List. Cela signifie que vous pouvez assigner n'importe quel objet à la variable qui met en place l'interface List, y compris la classe ArrayList.
- Initialiser la variable avec une expression commençant par le mot clé new qui crée une instance de la classe ArrayList.

```
List<Integer> myList = new ArrayList<Integer> ();
```

double → Double

boolean → Boolean

float → Float

Les classes enveloppantes

~~List<int> myList = new ArrayList<int> ();~~

Collection ordonnée

Ajouter des éléments dans une liste:

```
myList.add(5);

myList.add(6);
```

Java *place* automatiquement la valeur dans un objet *Integer* et l'ajoute à la liste (Autoboxing).

.set()

.remove()

.size()

Exercice

Pour cet exercice, vous pouvez vous aider de la documentation en ligne de la classe ArrayList

- Créez un programme Java qui crée une collection (ArrayList) de noms de pays puis alimenter cette collection avec quelques valeurs et afficher la taille de la collection

- Complétez le programme pour afficher le contenu de la collection.

- Trouvez une méthode pour vider la collection et modifiez votre programme pour afficher un message d'erreur lorqu'elle est vide et afficher le contenu lorsqu'elle n'est pas vide.

- Après avoir de nouveau alimenté votre liste de pays, modifiez le nom d'un pays et affichez de nouveau la liste des pays.

Conseil : Pour modifier le nom, il faut supprimer un élément (remove) et en ajouter un autre (add).

Exercice

- Triez votre collection et ré-affichez la liste des pays. Pour trier notre collection, il faut utiliser la méthode sort .

Si vous regardez la documentation de la classe ArrayList , vous ne trouvez pas cette méthode. Mais allez voir la documentation de la classe Collections :

cette méthode y est présente. Et par chance, une ArrayList est aussi une collection . Comme Médor qui est un Chien est aussi un Animal . Ces notions seront vus ultérieurement. . .

Pour le moment, utilisons la syntaxe suivante Collections.sort(uneArrayList) pour trier une ArrayList.

Collection non ordonée

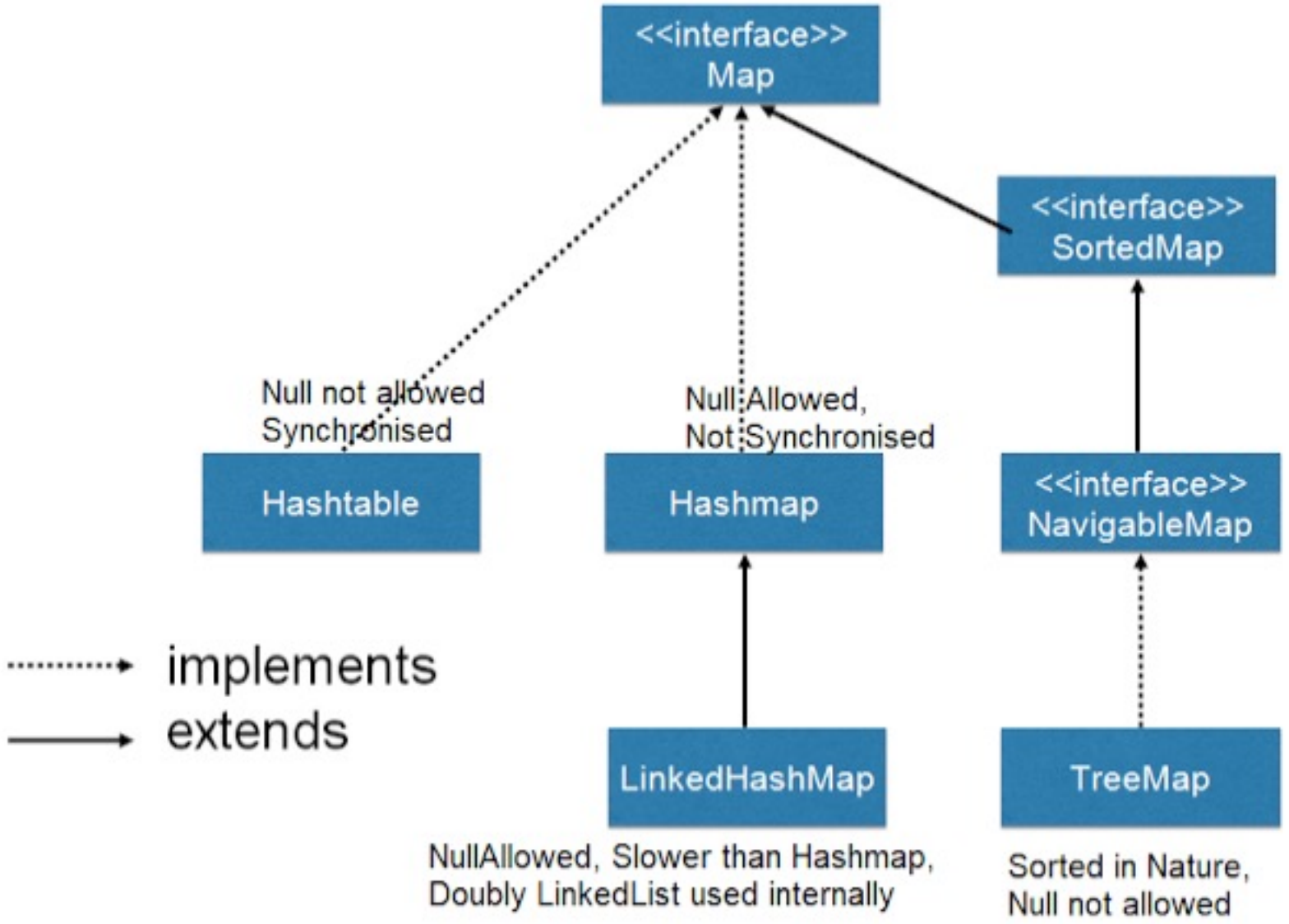
Avec un Interface set, on peut simuler le concept d'ensemble des objets.

Déclaration:

```
Set<String> ingredients = new HashSet<String>();
```

- **ajouter** un nouvel élément avec une nouvelle clé. add()
- **supprimer** un élément pour une clé spécifique. remove()
- **compter** le nombre d'éléments de l'ensemble. size()

Map Interface



Map

- Un **dictionnaire** est une **liste d'éléments organisés en fonction d'une clé**.

- Le type de clé utilisé de la façon la plus courante est **String**.

Déclaration:

```
Map<String, Integer> myMap = new HashMap<String, Integer>()
```

Map

- **accéder** à la valeur d'un élément par sa clé ;
- **ajouter** un nouvel élément avec une nouvelle clé et une valeur ;
- **supprimer** un élément pour une clé spécifique.

Exercice

```
public class Etudiant {
    private String nom, prenom;
    private int age;
    private int [] notes;

    public Etudiant(String _nom, String _prenom, int _age) {
    nom = _nom;
    prenom = _prenom;
    age = _age;
    }
    public String getNom() {
    return nom;
    }
    void setAge(int _age) {
    age = _age;
    }
}
```

modifiez la classe pour pouvoir ajouter une note à un étudiant.

Dans la méthode main(), créez plusieurs étudiants, affichez maintenant pour chaque étudiant la liste de ses notes; affichez aussi la moyennes de ses notes; enfin, affichez la moyenne de tous les étudiants.

Exercice

On veut intégrer la gestion des étudiants dans celle d'une scolarité. Pour une année particulière, une scolarité gère une collection de promotions. Chaque promotion est nommée et comprend une liste d'étudiants.

- Une promotion d'étudiants représente tous les étudiants d'une même classe (IUP1, IUP2 ou IUP3 par exemple)

- La scolarité gère un ensemble de promotions. Développez les classes Promotion et Scolarité : outre les fonctions classiques :

  - ajout et retrait d'une promotion,
  - ajout et retrait d'un étudiant dans une promotion, la scolarité doit pouvoir :

    - retrouver un étudiants par son nom et
    - retrouver la promotion d'un étudiant.

Exercice

Quand un étudiant est ajouté dans une promotion, il faut tester que cet étudiant n'existe pas déjà dans la promotion.

Utilisez un Map pour programmer les recherches et le test d'unicité.

De plus, on a besoin d'une fonction d'affichage pour chacune des classes :

- l'affichage d'une promotion doit afficher le nom de la promotion, la liste des étudiants et la moyenne des notes,

- l'affichage de la scolarite doit afficher l'année et toutes ses promotions. Pour chacune des classes développées, il est demandé de mettre en oeuvre et d'utiliser equals et toString.

Généricité

ancien style:

```
import java.util.ArrayList;
import java.util.List;

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

        List list = new ArrayList();
        list.add("apple");
        list.add("orange");
```

Comment enregistrer un élément dans une variable?

Généricité

List<Integer> myList = new ArrayList<Integer> ();

- myList est une liste avec les éléments de type Integer.

Set<String> ingredients = new HashSet<String>();

- ingredients est une Set avec les éléments de type String.

Généralement: On peux créer une classe paramétrée avec un type spécifique.

Généticité

```java
public class Machine {
    String name;

    public void display() {
        System.out.println("This is a machine");
    }
}
```

```java
import java.util.ArrayList;
import java.util.List;

public class Box<Machine> {
    private List<Machine> list = new ArrayList<>();

    public void ajoute(Machine machine) {
        list.add(machine);
    }

    public Machine enlevePremier() {
        return list.get(0);
    }
}
```

Exercice

On souhaite écrire un programme permettant à l'utilisateur de manipuler des ensembles au sens mathématique :

- Les valeurs contenues dans un ensemble ne sont pas ordonnées,
- Les valeurs ne peuvent être présentes qu'une seule et unique fois dans l'ensemble.

On souhaite pouvoir manipuler des ensembles d'entiers, des ensembles de réels, des ensembles de . . . et écrire le moins de code possible ! Les étapes à suivre sont les suivantes :

Exercice

Ecrivez la déclaration de la classe Ensemble qui contient :
– Une variable d'instance, liste de type ArrayList, qui contiendra les différentes valeurs de l'ensemble (type générique).

– Une variable d'instance, card de type entier, qui contiendra le cardinal de l'ensemble.

– Le constructeur qui initialise un ensemble à l'ensemble vide.

– Une méthode, ajoute, qui permet d'ajouter un élément à un ensemble. Si l'élément n'est pas déjà dans l'ensemble, la fonction retournera VRAI. Elle retournera FAUX dans le cas contraire.

– Une méthode, enleve, qui permet d'enlever un élément à un ensemble. Si l'´elément est présent dans l'ensemble, la fonction retournera VRAI. Elle retournera FAUX dans le cas contraire.

– Une surcharge de la méthode, toString, qui indique le cardinal de l'ensemble ainsi que la valeur de tous les éléments qui le composent.

Exercice

- Ecrivez la déclaration de la classe Principal qui:

– Affiche un petit menu :
1- Ajouter un élément,
2- Enlever un élément,

– En fonction du choix de l'utilisateur :
Ajoute un élément : si l'ajout a été possible affiche le nouvel état de l'ensemble, puis affiche à nouveau le menu ;

si l'ajout n'a pu se faire affiche simplement un message.

– Enléve un élément : si la suppression a été possible affiche le nouvel état de l'ensemble, puis affiche à nouveau le menu ;

si la suppression n'a pu se faire affiche simplement un message.