TP8 - INTERFACES

RAPPEL INTERFACE

La notion d'interface permet de définir un contrat pour un ensemble de classes.

Par contrat on entend un ensemble d'opérations, incluant des opérations d'accès à des données, que l'on s'attend à avoir dans toutes les classes respectant ce contrat.

RAPPEL INTERFACE

Une interface regroupe :

- des méthodes d'instance

```
public void mamethode();
```

- des méthodes d'instances abstraites
- des méthodes d'instances publiques par défaut

```
public default void mamethode(){...}
```

- des méthodes statiques

```
public static void mamethode(){...}
```

- des attributs de classe constants public

RAPPEL INTERFACE

On ne peut pas mettre :

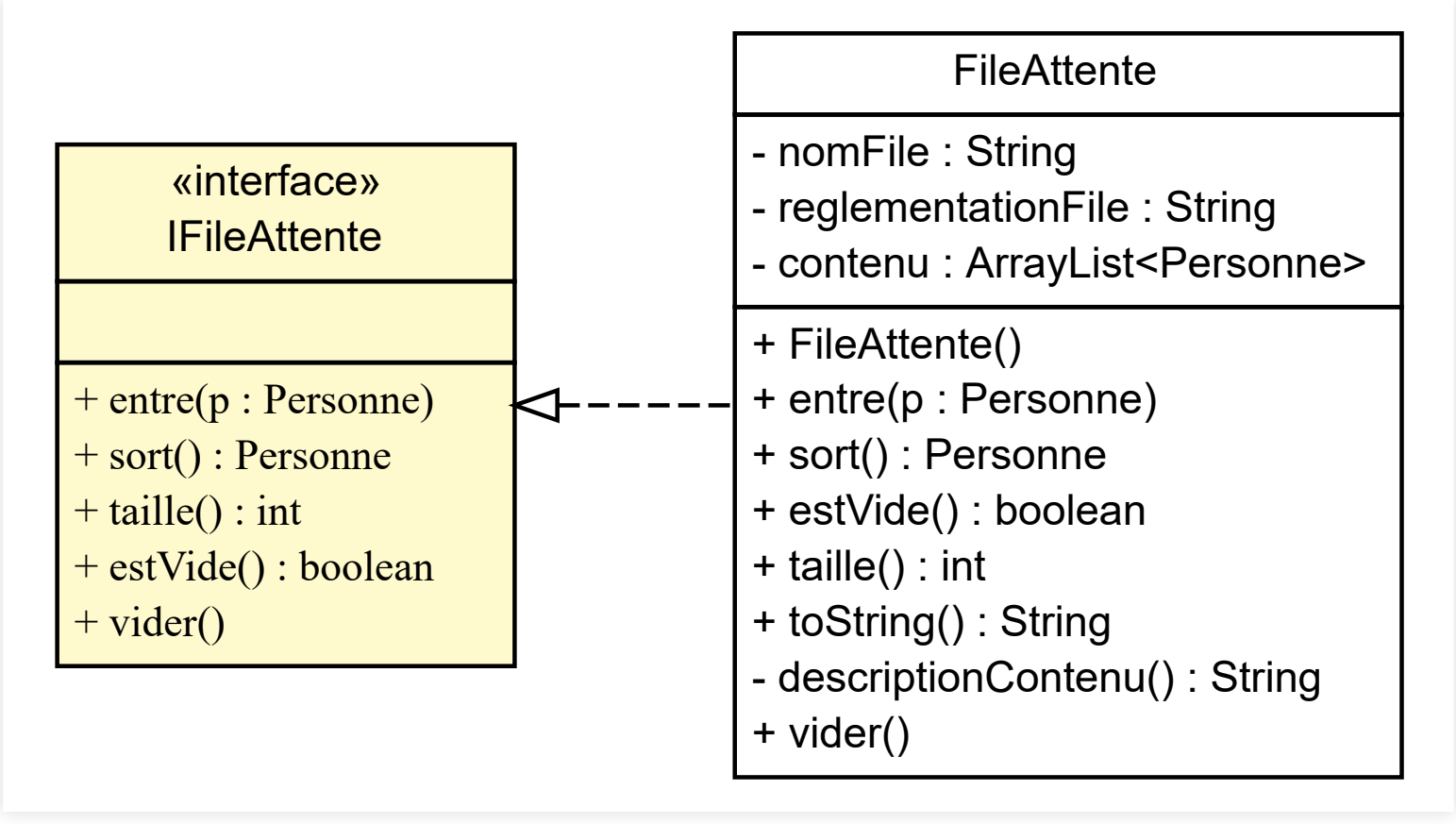
- de constructeurs
- d'attributs d'instances
- de code dans les méthodes
  (en dehors des methodes par défaut et des méthodes statiques)

LE MOT CLÉ "DEFAUT"

- Introduit en Java 8
- Permet d'implementer des méthodes dans l'interface
- Evite de rompre le "contrat" entre l'interface et les classes

```
package tp7;
import tp6.Personne;
public interface IFileAttente {
   ...
   public default String getReglementation(){
       return "sans priorités";
   }
}
```

MODÉLISATION UML



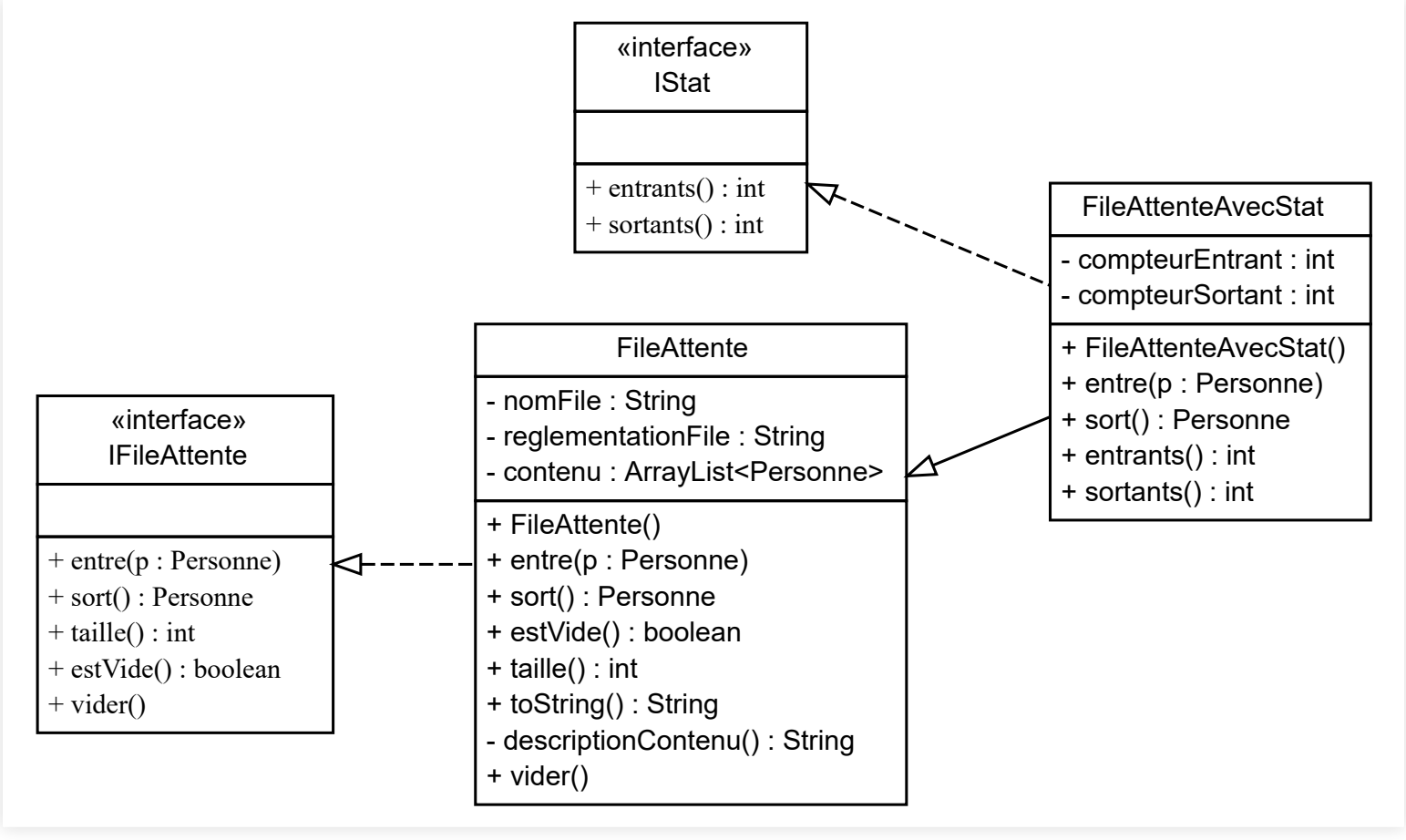
Exemple d'interface de la classe FileAttente

EXEMPLE D'INTERFACE

```
package tp7;
import tp6.Personne;
public interface IFileAttente {
    public void entre(Personne p);
    public Personne sort();
    public int taille();
    public boolean estVide();
    public void vider();
}
```

```
public class FileAttente implements IFileAttente{
...
}
```

MODÉLISATION UML



EXEMPLE D'UTILISATION D'INTERFACE

```
public static <E> extends Comparable<E>> E max(List<E> c) {
    if (c.isEmpty()){ return null;}
    E max = c.get(0);
    for (E e : c){ if (e.compareTo(max) > 0){ max = e; }}
    return max;
}
```

- La méthode max ne manipule pas de classes directement

Si on veut l'utiliser avec la classe Etudiant il suffit que la classe Etudiant implémente l'interface Comparable

INTERFACE COMPARABLE

```
public class Etudiant implements Comparable<Etudiant>{
    ...
    @Override
    public int compareTo(Etudiant o) {
        return this.getNom().compareTo(o.getNom());
    }
 }
```

- L'interface Comparable garantie que les étudiants sont comparables entre eux et permet donc le tri, la recherche d'un max etc...

```
// Tri d'une liste d'étudiant par ordre lexicographique
Collections.sort(listeEtudiant);
// Recherche du max
Collections.max(listeEtudiant);
```

INTERFACE COMPARATOR

- Permet d'implementer d'autres façons de comparer deux éléments
- Exemple : Comparer des personnes en fonction de leur age

```
// Tri d'une liste d'étudiant en fonction de leur age
Collections.sort(listeEtudiant, new ComparateurAge());
```