Introduction à la programmation en JAVA

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Table des matières

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

En Résumé

- Introduite dans **Java 5.0**, la généricité permet de paramétrer du code (classe, interface ou méthode) avec un ou plusieurs types de données

- Exemple
  - Sans génériques

```
List list = new ArrayList();
list.add("Hello");
String message = (String) list.get(0);
```

  - Avec génériques

```
List<String> list = new ArrayList<String>();
list.add("Hello");
String message = list.get(0);
```

Avantages

Plus de **lisibilité** : suppression des transtypages.

```
List list = new ArrayList(new Integer(2));
Integer i = (Integer) list.get(0);
List<Integer> list = new ArrayList<>(new Integer(2));
Integer I = list.get(0);
```

Plus de **sûreté** : empêcher d'ajouter n'importe quoi dans une collection

```
List<String> list = new ArrayList<String>();
list.add(10);
```

Erreur de compilation

Réutilisabilité du code

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Classe générique (1/3)

- Classe avec une ou plusieurs **variables de type**

- Exemple

variable de type

```
public class ArrayList<E>
    extends ArrayList<E>
    implements List<E>, Cloneable, Serializable
{
    public ArrayList();
    public boolean add(E elem);
    public E get(int index);
    public boolean remove(Object o);
    public boolean contains(Object o);
    public Object[] toArray();
    public Iterator<E> iterator();
    public List<E> subList(int from, int to);
    //…
}
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Classe générique (2/3)

- `public class ArrayList<E>`

  `<E>` permet de définir la variable de type, qui peut être utilisée dans la classe générique

  Par convention, on utilise une simple lettre capitale pour les noms de variable de type

- `public E getIndex(int index)`

  Utilisation de la variable de type à la place d'un type ordinaire

Variable de type (1/2)

- Une variable de type peut être utilisée dans une classe générique pour :

1. déclarer un paramètre de méthode

```
public class ArrayList<E>
{
    …
    public void add(E elem);
    …
```

2. déclarer le type de retour d'une méthode

```
public class ArrayList<E>
{
    …
    public E get(int index);
    …
```

Variable de type (2/2)

3. déclarer une variable

```
public class ArrayList<E>
{
    …
    private E data;
    …
```

4. effectuer un transtypage (warning du compilateur)

```
…
data = (E) s.readObject();
…
```

Est interdit `new E()`

- instancier une variable de type
- utiliser une variable de type comme opérande droite de l'opérateur `instanceof` `X instanceof E`

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Héritage et généricité

- Une classe générique peut étendre une autre classe générique si les variables de type sont identiques

```
public class Pair<T>
{
    //…
}

public class Triplet<T> extends Pair<T>
{
    //…
}
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Instancier une classe générique (1/2)

- Une classe générique ne peut pas être directement utilisée

```
List<E> = new LinkedList<E>();
```

Erreur de compilation

- Il faut instancier le type générique en remplaçant les variables de types par les types effectifs

```
List<Integer> = new LinkedList<Integer>();
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Instancier une classe générique (2/2)

- Sont acceptés
  - Classe (concrète ou abstraite) et interface
  - Classe générique

```
List<List<String>> = new ArrayList<List<String>>();
```

  - Au sein d'une classe générique, l'argument de type peut également être une variable de type

```
public class MyList<E> {
    …
    items = new ArrayList<E>();
```

Un argument de type ne peut pas être un type primitif

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Restriction de type

- On veut imposer des limites sur les types effectifs d'une classe générique …

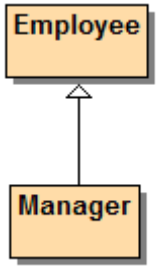
- Les contraintes `extends` apportent une solution

```
public class Pair<K extends Serializable, V extends Serializable>
    implements Serializable
{
    //…
}
```

Wildcard (1/4)

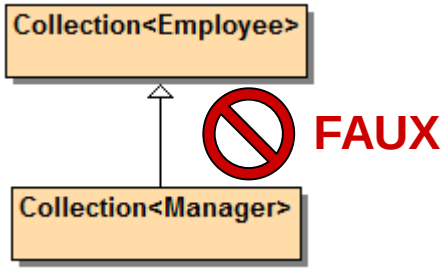
- Soit la hiérarchie de classes suivante

```
public void print(Collection<Employee> employees)
{
    for (Employee e : employees) {
        System.out.println(e.getName());
    }
}
```



- Utiliser une méthode générique avec toute sous-classe de la variable de type ?

```
List<Manager> managers = new ArrayList<Manager>();
//…
print(managers);
```



Erreur de compilation

Wildcard (2/4)

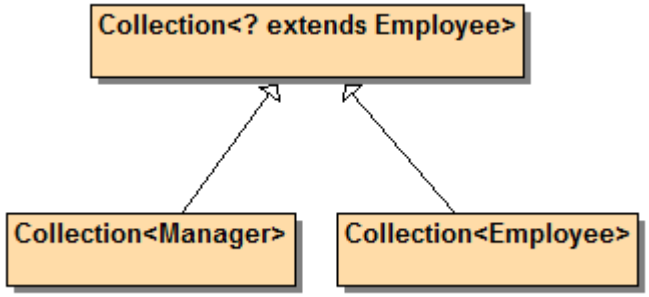
- … L'instanciation du type générique avec wildcard (« ? ») permet de résoudre ce problème

```
List<? extends Employee>
```

- Il est possible de spécifier qu'un paramètre de type est toute sous-classe ou sur-classe d'une classe donnée
  - `<?>` désigne un type inconnu
  - `<? extends C>` désigne un type inconnu qui est C ou un sous-type de C
  - `<? super C>` désigne un type inconnu qui est C ou un sur-type de C

Wildcard (3/4)

- Exemple 1




```
public class Joker
{
    public static void print(Collection<? extends Employee> employees)
    {
         for (Employee e : employees)
              System.out.println(e.getName());
    }

    public static void main(String[] args)
    {
        List<Manager> managers = new ArrayList<Manager>();
        //…
        print(managers);
    }
}
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Wildcard (4/4)

- Exemple 2

```
public class Generic<T extends Person>
{

    public void doSomething(List<? extends T> persons)
    {
        // …
    }

}
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Interfaces génériques

- Déclaration

```
interface Stack<T>
{
    boolean isEmpty();
    void push(T t);
    T pop();
    Integer size();
    T peek();
}
```

Méthode générique

- Déclaration de la méthode générique

```
class ArrayHelper {
    public static <T> Collection<T> toCollection(T[] a) {
        List<T> c = new LinkedList<T>();
        for (T o : a)
            c.add(o);
        return c;
    }
}
```

- Appel de la méthode

```
Collection<String> n = ArrayHelper.<String>toCollection(names);
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions

Legacy code

- Il est permis d'utiliser une classe générique sans argument de types. Le type retourné est alors Object.

```
LinkedList list = new LinkedList();
List.addItem("Hello");
String hello = (String) list.getFirstItem();
```

Exercices

1. Créer une classe générique nommée *GenericContainer* qui possède un attribut de classe générique passé en paramètre dans le constructeur et récupéré via son getter.

2. Représenter la clef composite d'une DB avec une classe nommée *CompositeKey* qui contient deux attributs génériques de type non obligatoirement identiques *firstKey* et *secondKey.* Réécrivez la méthode toString et equals.

3. Ecrire une méthode qui échange la position de deux éléments dans un tableau générique reçu en paramètre.

Exercices

4. Soit une classe non générique qui a le rôle d'une librairie pour les medias suivants: livre, video, journal. Transformez la pour qu'elle devienne générique.

```
public class Library {
    private List resources = new ArrayList();
    public void addMedia(Media x) {
        resources.add(x);
    }
    public Media retrieveLast() {
        int size = resources.size();
        if (size > 0) {
            return (Media)resources.get(size - 1);
        }
        return null;
    }
}

interface Media {
}

interface Book extends Media {
}

interface Video extends Media {
}

interface Newspaper extends Media {
}
```

technocité
Centre de compétence

WAVENET
Intelligent IT Solutions