Types paramétrés

Alexis Nasr (d'après les slides de Arnaud Labourel)

La classe `Object`

Par défaut, les classes étendent la classe `Object`, qui possède les méthodes suivantes, que l'on peut redéfinir :

- `boolean equals(Object obj)`: Indicates whether some other object is "equal to" this one.
- `String toString()`: Returns a string representation of the object.
- ...

Test d'égalité

- `o1 == o2` : vrai si `o1` et `o2` sont le même objet et faux sinon
- `o1.equals(o2)` : vrai si `o1` et `o2` correspondent à deux objets considérés égaux (exemple : étudiant ayant le même id, chaîne de caractères ayant les mêmes caractères, ...)

Stack d'Object

- Supposons que nous ayons la classe suivante :

```
public class Stack {
    private Object[] stack = new Object[100];
    private int size = 0;
    public void push(Object object) {
        stack[size] = object; size++;
}
    public Object pop() {
        size--;
        Object object = stack[size];
        stack[size]=null; // Pour le Garbage Collector.
        return object;
    }
}
```

Problème de Stack d'Object

- Nous rencontrons le problème suivant :

```
Stack stack = new Stack();
String string = "truc";
stack.push(string);
string = (String)stack.pop();
// Transtypage obligatoire !
```

- Nous avons également le problème suivant :

```
Stack stack = new Stack();
Integer intValue = new Integer(2);
stack.push(intValue);
String string = (String)stack.pop();
// Erreur à l'exécution
```

La solution : types paramétrés

- Par conséquent, on souhaiterait pouvoir préciser le type des éléments :

```
Stack<String> stack = new Stack<String>();
String string = "truc";
stack.push(string); // Le paramètre doit être un String.
String string = stack.pop(); // retourne un String.
```

- Java nous permet de définir une classe Stack qui prend en paramètre un type. Ce type paramétré va pouvoir être utilisé dans les signatures des méthodes et lors de la définition des champs de la classe.
- Lors de la compilation, Java va utiliser le type paramétré pour effectuer :
  - des vérifications de type ;
  - des transtypages automatiques ;
  - des opérations d'emballage ou de déballage de valeurs.

Définition de classes paramétrées

- La nouvelle version de la classe Stack :

```
public class Stack<T> {
    private Object[] stack = new Object[100];
    private int size = 0;
    public void push(T element) {
        stack[size] = element;
        size++;
    }
    public T pop() {
        size--;
        T element = (T)stack[size];
        stack[size] = null;
        return element;
    }
}
```

Emballage et déballage

- Les types primitifs ne sont pas des classes :
- Dans le cas d'un `int`, on doit utiliser la classe d'emballage (wrapper class) Integer :
  - Interdit : ~~`Stack<int> stack = new Stack<int>();`~~
  - Autorisé :

```
Stack<Integer> stack = new Stack<Integer>();
int intValue = 2;
Integer integer = new Integer(intValue);
// → emballage du int dans un Integer.
stack.push(integer);
Integer otherInteger = stack.pop();
int ohterIntValue = otherInteger.intValue();
// → déballage du int présent dans le Integer.
```

Types primitifs

type	classe d'emballage	taille	valeurs possibles
byte	Byte	8 bits	-128 à 127
short	Short	16 bits	-32768 à 32767
int	Integer	32 bits	-2^{31} à $2^{31}-1$
long	Long	64 bits	-2^{63} à $2^{63}-1$
float	Float	32 bits	
double	Double	64 bits	
char	Character	16 bits	caractère unicode
boolean	Boolean	non définie	`false` ou `true`

Emballage et déballage automatique

Depuis Java 5, il existe l'emballage et le déballage automatique :

```
Stack<Integer> stack = new Stack<Integer>();
int intValue = 2;
stack.push(intValue);
// → emballage automatique du int dans un Integer.
int otherIntValue = stack.pop();
// → déballage automatique du int.
```

Attention

Il est important de noter que des allocations sont effectuées lors des emballages sans que des `new` soient présents dans le code.

Exemple : liste chaînée générique

On considère une liste chaînée de `String`

```
public class LinkedList {
    private class Node {
        private String data;
        private Node next;
        public Node(String data, Node next) {
            this.data = data;
            this.next = next;
        }
    }
    private Node first = null;
    public void add(String data) {
        first = new Node(data, first);
    }
}
```

Exemple : liste chaînée générique

Nous la transformons en classe paramétrée de la façon suivante :

```
public class LinkedList<T> {
    private class Node {
        private T data;
        private Node next;
        public Node(T data, Node next) {
            this.data = data;
            this.next = next;
        }
    }
    private Node first = null;
    public void add(T data) {
        first = new Node(data, first);
    }
}
```