Classes utiles et types primitifs

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

Les types primitifs

En java, il existe des types **primitifs** qui ne sont pas des objets :

type	catégorie	taille	valeurs possibles	affichage
byte	entier	8 bits	-128 à 127	0
short	entier	16 bits	-32768 à 32767	0
int	entier	32 bits	-2^{31} à $2^{31}-1$	0
long	entier	64 bits	-2^{63} à $2^{63}-1$	0
float	flottant	32 bits		0.0
double	flottant	64 bits		0.0
char	caractère	16 bits	caractère unicode	'\000'
boolean	booléen	non définie	`false` ou `true`	`false`

Comportement des types primitifs

- Lors d'un appel de méthode les arguments sont passés **par valeur** : une copie de la valeur de l'argument est créé lors de l'appel.
- Pour les objets, cela signifie passer une **copie** de la référence : il est donc possible de modifier l'état de l'objet.
- Pour les types primitifs, cela signifie que l'argument est une **copie uniquement créée pour l'appel** et toute modification de sa valeur n'aura pas d'impact en dehors de l'appel.

Tableaux unidimensionnels

- En Java, les tableaux sont des objets (et donc des références).
- Déclaration d'une variable de type "référence vers un tableau" :

```
int[] arrayOfInt;
double[] arrayOfDouble;
```

- Construction d'un tableau :

```
arrayOfInt = new int[10]
arrayOfDouble = new double[3];
```

- Utilisation d'un tableau :

```
arrayOfInt[0] = 5;
arrayOfInt[9] = 10;
arrayOfDouble[2] = arrayOfInt[0] / arrayOfInt[9];
system.out.println(arrayOfDouble.length) // 3
```

Tableaux multidimensionnels

- Déclaration :

```
int[][] matrixOfInt;
```

- Construction :

```
matrixOfInt = new int[10][];
for(int row = 0; row < matrixOfInt.length; row++)
    matrixOfInt[row] = new int[5];
/* ou directement */
matrix = new int[10][5];
```

- Tableau de tableaux de tailles différentes

```
matrixOfInt = new int[10][];
for(int row = 0; row < matrixOfInt.length; row++)
    matrixOfInt[row] = new int[row + 1];
```

Chaînes de caractères (1/2)

- La classe `String` permet de définir des chaînes de caractères invariables (**immutable**)
- Déclaration et création :

```
String hello = "Hello";
String world = "World";
```

- Concaténation :

```
String helloWorld = hello + " " + world + " ! ";
int integer = 13;
String helloWorld1213 = hello + " " + world + " "
                        + 12 + " " + integer;
```

Chaînes de caractères (2/2)

- Affichage :

```
System.out.print(helloWorld);   // affiche "Hello World !"
System.out.println(helloWorld); // affiche "Hello World !"
                                // avec retour à la ligne
```

- Comparaison :

```
String a1 = "a";
String a2 = "a";
String a3 = new String("a");
System.out.println(a1==a2);          // affiche "true"
System.out.println(a1==a3);          // affiche "false"
System.out.println(a1.equals(a3)); // affiche "true"
```

Listes

- Les classes `ArrayList` et `LinkedList` : permettent de créer des listes en java.

```
List<String> strings = new ArrayList<>();
strings.add("first");
strings.add("second");
System.out.println(strings);
// affiche "[first, second]"
```