Bases de la POO / Java

Types et objets

Variables

- de type primitif
  - int, char, double…
- de type objet
  - Représentent un objet d'une classe donnée
  - Toutes les classes héritent de la super-classe **Object**

Types primitifs

Type	Description
`byte`	entier signé, 8 bits
`boolean`	booléen, 1 bit, `true` / `false`
`char`	caractère, 16 bits, norme Unicode
`short`	entier signé, 16 bits, *big endian*
`int`	entier signé, 32 bits, *big endian*
`long`	entier signé, 64 bits, *big endian*
`float`	flottant, 32 bits, norme IEEE 754
`double`	flottant, 64 bits, norme IEEE 754

Equivalence type primitif/classe

Type primitif	Classe équivalente
`byte`	`Byte`
`boolean`	`Boolean`
`char`	`Character`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`

Equivalence type primitif/classe

- Il existe une classe équivalente à chaque type primitif
  - Exemple: type «**int**» ⇔ classe «**Integer**»
- Création d'un objet à partir d'un primitif
  - `int a = 4;`
  - `Integer i = new Integer(a);`
- Création d'un primitif à partir d'un objet
  - `int j = i.intValue();`
- Création d'un primitif à partir d'une chaîne
  - `int i = Integer.parseInt("4");`

Passage de paramètres

- **Forte analogie C / Java**
  - Les valeurs numériques sont transmises par **valeur**
  - Tous les **objets** sont transmis par **référence**
  - Les tableaux (qui sont des objets !) sont transmis par **référence**

Les tableaux en Java (1)

- Algorithmique identique aux modules M112-M113
- Syntaxe déclarative identique au langage C
- Taille mémoire fixée à la création
- Tableaux fortement typés (comme en langage C)
- Eléments désignés par leur rang (débute à 0)
- Syntaxe de désignation identique au langage C
- Possibilité d'initialisation à la compilation ({ --- })

Les tableaux en Java (2)

- En Java, tout tableau est un **objet**
- Identification de la **classe génitrice** via []
- Attribut public **length**
- Tableaux à 2 dimensions [][]
- Tableaux à 3 dimensions [][][]

- Tableaux à 2 dimensions alloués comme des **tableaux de tableaux**
- Idem pour les tableaux à 3 dimensions
- **Tableaux d'objets**

- http://imss-www.upmf-grenoble.fr/prevert/Prog/Java/CoursJava/tableaux.html

Constantes chaînes

- **Codage des caractères**
  - Codage ASCII pour le langage C (8 bits)
  - Codage **UNICODE** pour le langage Java (16 bits)
- Format identique des constantes chaînes
- Type caractère dans les deux langages
- Possibilité commune de déclarer des tableaux de caractères

Variables chaînes (1)

- classe **String du** package **java.lang**
- En Java, les chaînes de caractères sont des **objets**

- Création d'une chaîne
  - `String s = new String("bonjour");`
  - `String t = "bonjour";`

- Taille d'une chaîne
  - `int taille = s.length();`

- En Java, une chaîne n'est pas un tableau
  - Pas d'opérateur **[ ]**

- Accès à un caractère
  - `char c = s.charAt(3);`

- Transtypage implicite
  - int i= 12;
  - String s= "" + i;

Chaîne de caractères (2)

- Concaténation de chaînes
  - `String s = t + " toi";`
  - `t += " toi";`
- Comparaison de deux chaînes
  - Opérateur == compare les références des objets
  - Méthode «**equals**» compare les chaînes
  - `if (s.equals("bonjour")) ...`
- Recherche d'une sous-chaîne
  - `int position = s.indexOf("toi");`
- Extraction d'une sous-chaîne
  - `String t = s.substring(position,position+3);`
- http://docs.oracle.com/javase/7/docs/api/java/lang/String.html