String, tableaux, comparaisons

Licence professionnelle

UV G5a

Pascal Divoux ; Cedric Wemmert

Les Chaînes de caractères

String

➔ Classe du package java.lang (import inutile)

```
String libelle=new String(''coucou'');
```

ou bien `String libelle=''coucou'';`

➔opérateur + (concaténation)

```
libelle+='' ! '';
```

➔ la méthode length() renvoie le nombre de caractères

```
int lg=libelle.length();
```

➔ la methode « charAt(i) » renvoie le ième caractère

```
char c=str.charAt(i);
```

➔autres méthodes utiles

- *substring : extrait une sous-chaine*
- *toUpperCase : convertit en majuscule*
- *indexOf : retourne l 'indice d 'un caractère*
- *matches : interpretation syntaxique d 'une expr reguliè re*
- *...*

Les Chaînes de caractères

Comparaison de chaînes

➔ Eviter la comparaison avec ==

```
if (ch1==ch2) ...
```

qui compare les références et non les chaînes

```
String ch1=''toto'';
String ch2=ch1;
if (ch1==ch2)... //donnera true
```

mais

```
String ch3=Console.readLine();
if (ch1==ch3)... //donnera false meme si
  l 'utilisateur tape « toto »
```

➔ Utiliser equals

qui compare les contenus

```
if (ch1.equals(ch3)) //donne true
if (ch1.equals(''toto'')) //donne true
```

Les Chaînes de caractères

StringBuffer

➔ La classe String est non modifiable

```
libelle+=prenom;
```

crée en fait un nouveau String de meme nom

➔ Si on veut modifier une chaîne, Il faut utiliser StringBuffer

qui possède des méthodes de modifications

```
append
delete
reverse
...
```

Les Chaînes de caractères

toString()

➔ Chaque classe java possède une méthode « toString() » qui permet d ’afficher ses caractéristiques principales ; c ’est ce qui permet d ’écrire :

```
JOptionPane jop = new JOptionPane();
System.out.println(jop);
```

ou

```
int i = 5;
System.out.println(`` i = ''+i);
```

fait un appel implicite à la méthode « toString » :

```
System.out.println(jop.toString());
```

➔ Prenez l’habitude de créer une telle méthode pour les classes que vous développez

- *Affichage*
- *débuggage*

Les tableaux et les matrices

Déclaration

```
int[] monTableau;
int monTableau[];//equivalent
```

- *La déclaration ne réserve pas de place en mémoire (pas de bornes)*
- *Une matrice est un tableau de tableau*

```
int[][] maMatrice;
```

Création

- *Les bornes du tableau sont déclarées à la création*

```
monTableau=new int[20];
maMatrice=new int[6][4];
char[][] maGrille=new char[12][];
```

- *La 2eme dimension pourra être fixée plus tard*
- *Les lignes n 'ont pas nécessairement toutes la même dimension*

Utilisation

```
System.out.println(monTableau[5]);
```

- *NB : les indices du tableau commencent à 0*
- *l 'attribut length donne la taille du tableau*

```
for (int i=0;i<t.length-1;i++)
  System.out.println(t[i]);
```

- *La tentative d 'accès en dehors des bornes du tableau génère une erreur :* « ArrayIndexOutOfBoundsException »

Collections

- Vector
- Vector<Type> (java 1.5)
- ArrayList<Type>(java 1.5)
  - `add(Object o)`
  - `get (int i)`
  - `size()`
  - `indexOf(Object o)`
  - `remove(int i)`
  - `...cf javadoc...`
- foreach (java 1.5)

```
for (Personne p : Employes) { ....}
```

Références, comparaison, copie

Les références

- *Le nom d'une variable ne représente que la référence vers un objet, pas l'objet lui-même. Les variables références sont passées par copie aux méthodes. Elle peut donc agir directement sur l'objet.*

```
public class UneClasse {
  void meth1 ( int i, StringBuffer s ) {
    i++; s.append("d");
  }
  void meth2 () {
    int i = 0; StringBuffer s = new
  StringBuffer("abc");
    meth1( i, s );
    System.out.println( "i="  + i + ", s="  + s);
    //i=0, s="abcd"
  }
}
```

Reférence, comparaison, copie

Comparaison

➔ Cf comparaison de chaines de caractères

➔La comparaison au moyen de l'opérateur == de 2 variables références vérifie que les 2 variables réfèrent au même objet en mémoire

```
Rectangle r1=new Rectangle(2,4);
Rectangle r2=new Rectangle(2,4);
if (r1==r2) //false car compare les références
```

➔ Pour vérifier que 2 objets différents contiennent les mêmes informations, il faut utiliser « equals »

```
if (r1.equals(r2)) // true : compare les contenus
  des objets
```

Reférence, comparaison, copie

Copie d 'objets

➔ Affecter une variable référence à une autre (opérateur =) ne fait pas une copie de l'objet : les deux variables réfèrent simplement au même objet

```
Rectangle r1=new Rectangle(2,4);
Rectangle r2=r1;
```

– *il n 'y a pas de nouvel objet, seulement 2 noms pour un même objet*

➔Le but d'une opération de copie est de créer un nouvel objet qui soit égal (au sens de equals) à l'objet de départ ; Pour cela, il faut utiliser la méthode **clone()** de l'objet

```
Rectangle r3 = r1.clone();
if ( r3.equals(r1) )     // true
if ( r2 == r1 )          // true
if ( r3 == r1 )          // false
```

Parcours G5a
Parcours G4 (sans objet)