



# Le langage Java

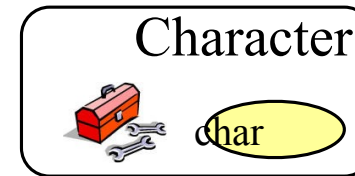
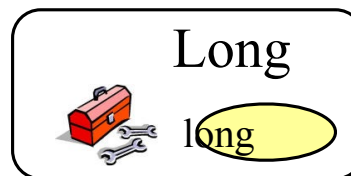
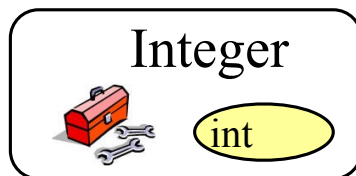
Classes de base

# Les Wrappers (1/3)

- Les types primitifs permettent d'effectuer des opérations simples et rapide

```
int i; int j;  
i = 2 + 3; j = i + 2;  
j++;
```

- Pour chaque type primitif il existe une classe dite Wrapper (emballage).
  - Une classe Wrapper contient une variable d'un type primitif donné et des outils facilitant la manipulation de cette variable.



...

# Les Wrappers (2/3)

- Classes Wrapper existantes : Boolean, Byte, Short, Integer, Long, Float, Double, Character
- méthodes proposées par les classes Wrapper :
  - toString() convertit en chaîne de caractères la variable contenue dans le Wrapper
  - intValue(), doubleValue(), longValue() ... pour des conversions numériques
- Les Wrappers de types numériques contiennent des constantes MAX\_VALUE et MIN\_VALUE

# Les Wrappers (3/3)

- Exemples d'utilisation
  - Méthodes d'instance

```
Integer i = new Integer(12);  
String s = i.toString();  
double d = i.doubleValue();  
Boolean b = new Boolean(true);  
boolean b2 = b.booleanValue();
```

- Méthodes statiques

```
String s = "12";  
int i = Integer.parseInt(s);  
boolean b = Boolean.getBoolean("true");
```



la plupart de ces méthodes ne s'appliquent pas au type Character qui n'enveloppe pas un nombre.

# String (1/4)

- Les String représentent les chaînes de caractères en Java
  - Les Strings sont des objets
  - Initialisation facilitée sans le mot-clé 'new'
  - La taille d'une String n'est pas limitée

```
String s1 = new String("hello world");  
String s2 = "hello world";
```

*Deux modes d'initialisation d'une String*

# String (2/4)

- Quelques méthodes utiles

- Longueur de la chaîne

- `int length()`

```
String s1 = new String("hello");  
int longueurChaine = s1.length();  
System.out.println(longueurChaine);
```

*Affiche 5*

- Comparaisons

- `int compareTo(String)`, `int compareToIgnoreCase(String)`

```
String s1 = new String("hello");  
String s2 = new String("hello");  
System.out.println(s1.compareTo(s2));
```

*Affiche 0*

# String (3/4)

- Manipulation de chaînes
  - trim() : supprime les espaces superflus

```
String s1 = new String("    hello  
world ");  
System.out.println(s1);  
System.out.println(s1.trim());
```

*Affiche :*  
*hello world*  
*hello world*

- toUpperCase(), toLowerCase() ...

```
String s1 = new String("hello world");  
System.out.println(s1.toUpperCase());
```

*Affiche « HELLO WORLD »*

# String (4/4)

- Concaténation : `concat(...)` ou opérateur `+`

```
String s1 = new String("hello");  
String s2 = s1 + (" world");  
String s3 = s1.concat(" world");
```



**Utilisée à grande échelle, la concaténation de String est une opération peu performante**

- Les instances de String sont dites *immuables*

```
String s2 = "hello world";  
s2 = "bye bye world";
```

*De manière sous-jacente,  
l'objet référencé par s2 est  
détruit puis recréé*



# StringBuffer / StringBuilder

- La concaténation d'objets String est peu performante
- Alternative pour construire des chaînes : StringBuffer
  - chaîne modifiable
  - meilleures performances
- Création
  - StringBuffer(), StringBuffer(String)
- Construction dynamique
  - append(valeur)- valeur : type primitif ou objet
  - insert(int position, valeur)- même remarque
- Conversion en String
  - toString(), ou new String(StringBuffer)

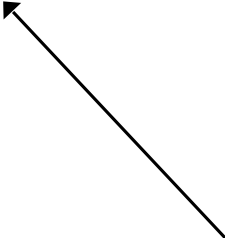
# Flux standards

- Sortie standard : `System.out`
- Erreur standard : `System.err`
- Écrire : `print(valeur)`,  
`println(valeur)`
  - valeur : type primitif ou objet

- `java.util.Date`
  - plupart des méthodes dépréciées : existent mais ne doivent pas être utilisées car elles peuvent disparaître dans une future version du JDK
  - pas de gestion des zones (Locale)
- `java.util.Calendar` et `java.util.GregorianCalendar`
  - tiennent compte de la zone (fuseaux horaires, etc)
  - référencent une date/heure avec accès au jour, heure, secondes...
  - instance : constructeurs ou `Calendar.getInstance()`
  - `Date getTime()` : retourne la Date associée à ce calendrier
- `java.text.DateFormat` et `java.text.SimpleDateFormat`
  - conversion Date/String
  - constructeur avec un motif pour la conversion
  - String  $\Rightarrow$  Date : `Date parse(String)`
  - Date  $\Rightarrow$  String : `String format(Date)`

# Exemple d'affichage d'une date

```
String s1 = new String("hello");  
Calendar calendrierMaintenant = Calendar.getInstance();  
Date maintenant = calendrierMaintenant.getTime();  
SimpleDateFormat formattage = new SimpleDateFormat("EEEE dd MMMM yyyy");  
System.out.println(formattage.format(maintenant));
```

An arrow pointing from the output box to the code box.

*Affiche :*  
*lundi 30 janvier 2006*