Java
Quelques classes

Saisie des données

- Les données peuvent être saisies
  - par la ligne de commande
    - on récupère les données pat le tableau args de la méthode `main()`
  - par interaction avec la console
  - par interaction via une IHM

Saisie des données

- Récupérer la ligne de commande
  - solution souhaitable uniquement pour paramétrer l'application

```
public class LigneCommande {

    public static void main(String[] args) {
        for(String cde : args)
            System.out.println(cde);
    }

}
```

Saisie de données

- Interaction avec la console
  - utilise l'entrée standard

```
BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Nom : ");
String nom = input.readLine();

System.out.println("Prénom :");
String prenom = input.readLine();

System.out.println("Bonjour "+prenom+" "+nom);
```

Saisie de données

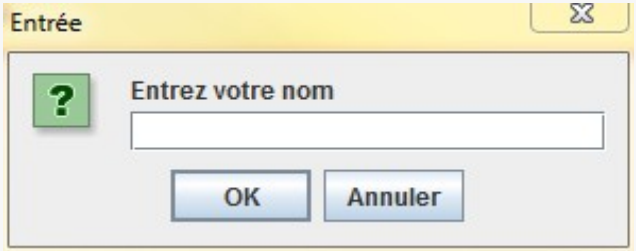
- Interaction avec la console
  - via la classe `Scanner`
    - chaque entrée est confirmée par la touche Return

```
Scanner input = new Scanner(System.in);
System.out.println("Entrez votre prénom et votre age");
String prenom = input.nextLine();
int age = input.nextInt();
System.out.println("Bonjour "+prenom+ "("+age+")");
```

Saisie de données

- Par une boite de dialogue

```
String nom = JOptionPane.showInputDialog(null,"Entrez votre nom","");
System.out.println("Hello, "+nom);
```



Saisie des données

- La plupart des modes de saisie renvoit des chaînes de caractères
  - il est souvent nécessaire de passer du type `String` vers les types `int`, `double`, ...
  - chaque wrapper de type primitif possède une méthode pour interpréter le `String` vers le type cible

```
int i = Integer.parseInt("12");
```

String

- Java possède plusieurs classe pour gérer les chaînes de caractères
  - `String` : chaîne de caractères non modifiables
  - `StringBuffer` : chaîne de caractères modifiables
    - `StringBuilder` : version thread safe
  - `StringTokenizer` : séparation d'une chaîne de caractères

String

- Possède de nombreux constructeurs
- Les méthodes de `String` permettent de comparer, rechercher, extraire, …
  - plus de 40 méthodes
  - toujours utiliser la méthode `equals()` pour comparer les `String`
    - pas de == qui compare les références

String

- Concaténation
  - utilisation de l'opérateur `+`
  - ou utilisation de la méthode `concat()`
    - retourne une nouvelle chaîne

```
String s1 = "Hello";
String s2 =", world";
String s3 = s1.concat(s2);
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
```

```
Hello
, world
Hello, world
```

String

- Extraction de caractères
  - plusieurs méthodes permettent de trouver la position d'un caractère, ou d'une chaîne
    - `indexOf()`, `lastIndexOf()`
  - plusieurs méthodes permettent de retourner une partie de la chaîne
    - `substring()`

```
String s4 = "Hello, world";
int posVirgule = s4.indexOf(',');
String s5 = s4.substring(posVirgule+1);
System.out.println(s5);
```

String

- Quelques méthodes utiles
  - `length()`
  - `toLowerCase()`, `toUpperCase()`
  - `trim()` : supprime les espaces de début et de fin
  - `replace(car oldChar, car newChar)`
  - `split()` : transforme une String en un tableau de String

StringBuffer

- Lorsque de nombreuses concaténations doivent être effectuées il est préférable d'utiliser `StringBuffer`
  - dès que les chaînes manipulées doivent être modifiables il faut privilégier `StringBuffer`
  - `StringBuilder` est l'équivalent thread safe
- Les mêmes types d'opérations sont utilisables sur `String` et `StringBuffer`
  - mais `StringBuffer` modifie l'instance en cours

StringBuffer

- Exemple de concaténation
  - méthode `append()`
  - les appels peuvent être chaînés

```
StringBuffer sb1 = new StringBuffer("Hello");
sb1.append(", ").append("world");
System.out.println(sb1.toString());
```

StringTokenizer

- Permet de casser une chaîne de caractères
  - exemple

```
String s1 = "1;Gaston;LAGAFFE;32";

StringTokenizer st = new StringTokenizer(s1,";");

System.out.println("Nombre de champs : "+st.countTokens());

while(st.hasMoreElements()){
    System.out.println(st.nextToken());
}
```

Manipulation des dates

- Les classes `java.util.Date` et `java.util.Calendar` permettent de manipuler les dates
  - attention il existe aussi un `java.sql.Date`
  - beaucoup de méthodes de `Date` sont obsolètes
  - `Calendar` est une classe abstraite
    - utiliser `GregorianCalendar`

Manipulation des dates

- Formater une date
  - exemple

```
DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
String d1 = df.format(new Date());
System.out.println(d1);
```

Manipulation des dates

- La classe `Calendar` utilise des constantes pour manipuler la structure d'une date
  - exemple

```
Calendar d2 =  new GregorianCalendar();
int jour = d2.get(Calendar.DAY_OF_MONTH);
int mois = d2.get(Calendar.MONTH) + 1;
int an = d2.get(Calendar.YEAR);
System.out.println(String.format("%02d/%02d/%04d", jour,mois,an));
```

les mois sont énumérés de 0 à 11

méthode de formatage des chaînes de caractères

Fonctions mathématiques

- La classe `Math` regroupe
  - un ensemble de constantes mathématiques
  - un ensemble de fonctions mathématiques
- L'ensemble des membres de la classe `Math` est `static`
  - pas besoin d'instancier un `Math`
  - une inclusion statique est possible depuis le JDK 1.5

```
System.out.println(Math.PI);
System.out.println(Math.min(10, 2));
```