

# Java Intro

Mathias DIDIER

3 mai 2015

## Organisation en packages

- Entrées/sorties(java.io)
- Reseaux (java.net)
- UI (java.awt/javafx.swing)
- BDD (java.sql)
- Methode d'objet distant (java.rmi)
- Mathematique (java.math)

# Plateforme

# Language Objet

## Concept

Modularité/Classes

Extensibilité/Heritage

## Technique

- Classes, interfaces, paquets
- Tout est en classe (Enveloppe pour les génériques)
- Objets accessibles par références.
- Heritage simple des classes
- Heritage multiple des interfaces
- Polymorphisme

# Rappel

## Encapsulation

Implémente le masquage d'informations et la modularité

## Polymorphisme

Permet a 2 objets d'obtenir un comportement different a un meme signal.

## Heritage

On peut définir de nouvelles classes basées sur des classes existantes, pour obtenir du code réutilisable et de l'organisation.

envoyer des messages à des objets sans connaître au moment du codage leur type spécifique.

Liaisons dynamiques (def du dessus)

# Objet

## Defini par

Ses variables d'instance

Son comportement (methodes)

## Classe

Constructeur qui defini l'objet (moule)

## This

"this" fais reference a l'objet dans lequel le code ce situe

# Garbage Collector

## Rammasse miettes

Permet la desallocation memoire des objets

C'est automatique

on peut definir un comportement avant son execution en

implémentant la fonction

```
public void finalise()
```

# Casting

## Upcasting

```
List list = new LinkedList();
```

## Downcasting

```
...(List) list; (cast classique)
```

## Nom

Transtypage



# Controle d'accès

## Static

Une variable static est commune a tout les objets de cette classe

+ - locales

Les methodes aussi peuvent etre static

## Abstract

## Utilisation

Les indices des tableaux commencent à 0

Taille : `tableau.length`

lève une `ArrayIndexOutOfBoundsException` si hors limites

# Chaines

## Intro

La classe String ne permet pas de modifier la chaîne

La classe StringBuffer elle le permet

la méthode length retourne le nombre de caractères dans la chaîne

la méthode charAt retourne le caractère situé à la position donnée

indexOf(char ch) -> trouver la position de la première occurrence de ch

indexOf(char ch, int start) : première position de ch  $\geq$  start

indexOf(String str) : première position de str  
String substring(int beginIndex, int endIndex)

# Interfaces

## Classe

Classe abstraite, ou tout les methodes sont abstraites

Une classe peut implementer plusieurs interfaces

Une interface peut heriter d'une autre interface

# Exceptions

## Intro

Lancer, ou lever ou exception signifie envoyer un signal particulier au bloc superieur  
qui pourra la traiter, ou bien la renvoyer au bloc superieur  
une classe succesptible de lever une exception doit le preciser dans son entete grace au mot throws

## Quelques exceptions

ArithmeticException -> division par zero

NullPointerException->reference nulle

ClassCastException -> tentative de cast illegal

NegativeArraySizeException -> creation tableau de taille negative

ArrayIndexOutOfBoundsException-> depassement limite d'un tableau

# Capturer/Lever

## Capturer

on execute le code dans un bloc try  
on declare un catch(Exception e)  
qui traite l'exception reçu

## Lever

On utilise throw(Exception) pour lever une exception

## Finaly

On peut declarer plusieurs catch, et si aucun correspond, on declare un finally



# Creation exception

## Type

Error sont les throwable grave, il ne faut pas les capturer  
Exceptions sont les exception courantes

## Redefinition

On fait heriter de Throwable un objet qui doit se comporter comme une exception

On utilise la fonction getMessage pour modifier le message qui apparait si l'exception n'est pas capturée

# Collections

## Intro

Groupe d'elements uniques -> Collection  
ou Set

Ensemble de paires clés-valeurs -> Map  
(tableaux de hachages)

# List

## LinkedList

liste lié de voisins en voisins  
optimisé pour des parcours complets

## ArrayList

Optimisé pour les acces aléatoires

## List

defini les methodes add, et remove

# Iterateur

## Iterator

On peut accéder aux éléments grâce à un itérateur

on le récupère grâce à `"list.iterator()"`

on vérifie qu'il y ait un élément suivant avec

`"itérateur.hasNext()"`

on saisit l'élément suivant grâce à `"(Cast vers le type presque obligatoire)itérateur.next()"`

enlève le dernier objet saisi avec `.remove`

# Comparator

# Comparator

## Comparer 2 objets

Pour pouvoir comparer 2 objets, il faut un comparateur approprié

Ainsi la classe doit implémenter l'interface comparable

## Fonctions

`int compare(Object o1, Object o2)`

`boolean equals(Object oth)`

On a ainsi accès au fonction de base pour comparer, trier, etc  
(Définition d'un ordre naturel)

## Amelioration des boucles

```
for(Object element : liste)
```

## Methodes

Ajout des methodes a arguments variables  
en fonction du type, et du nombre  
pour nombre variable d'arg on utilise les "..."  
ex "int ... valeurs"

# Scanner

## Entree formattes

```
Scanner s = new Scanner(System.in)  
s.next() s.nextInt etc  
s.close()
```



## Liste D'elements avec type

### Definition

```
List<String> ls = new ArrayList<String>();
```

Ainsi plus besoin de caster lors de la recuperation d'un objet