Une vraie introduction à Java

Rémi Nicole

1er avril 2014

Une vraie introduction à Java

Rémi Nicole

1er avril 2014

Sommaire

2014-04-01
Une vraie introduction à Java
└Sommaire

Sommaire
Le Java en général
Les bases
L'orienté objet
Notions utiles

Java ?

- Similaire à beaucoup d'autres langages
- Simple pour son domaine
- Beaucoup utilisé en entreprise

2014-04-01
Une vraie introduction à Java
Le Java en général
Introduction
Java ?

Java ?
- Similaire à beaucoup d'autres langages
- Simple pour son domaine
- Beaucoup utilisé en entreprise

Les spécificités de Java

- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK

2014-04-01
Une vraie introduction à Java
└Le Java en général
└Java et les autres
└Les spécificités de Java

Les spécificités de Java
- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK

Les spécificités de Java

- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK
- Les fichiers doivent avoir le même nom que la classe

2014-04-01
Une vraie introduction à Java
└Le Java en général
└Java et les autres
└Les spécificités de Java

Les spécificités de Java
- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK
- Les fichiers doivent avoir le même nom que la classe

Les spécificités de Java

- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK
- Les fichiers doivent avoir le même nom que la classe
- Le code est organisé en packages
- Les dossiers doivent avoir le même nom que le package actuel

2014-04-01
Une vraie introduction à Java
Le Java en général
Java et les autres
Les spécificités de Java

Les spécificités de Java
- Une seule compilation pour tout les systèmes d'exploitation (Windows, Mac OS, GNU/Linux)
- Beaucoup de classes dans le JDK
- Les fichiers doivent avoir le même nom que la classe
- Le code est organisé en packages
- Les dossiers doivent avoir le même nom que le package actuel

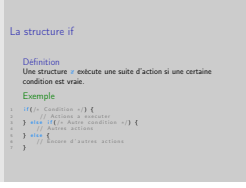
La structure if

Définition

Une structure `if` exécute une suite d'action si une certaine condition est vraie.

Exemple

```
1  if(/* Condition */) {
2      // Actions a executer
3  } else if(/* Autre condition */) {
4      // Autres actions
5  } else {
6      // Encore d'autres actions
7  }
```

- Les conditions peuvent être représentées comme des fonctions qui retourne un boolean
- Les accolades servent à regrouper une liste d'action. (Une action en regroupant plusieurs)

La structure switch I

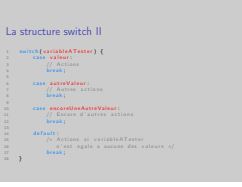
Définition

La structure switch permet de faire une suite de tests d'**égalité** plus légèrement que les if.

- Le switch fait des tests d'**égalité** uniquement.
- Le break sert à sortir de la structure quand il a terminé d'exécuter les instructions.

La structure switch II

```
1   switch(variableATester) {
2       case valeur:
3           // Actions
4           break;
5
6       case autreValeur:
7           // Autres actions
8           break;
9
10      case encoreUneAutreValeur:
11          // Encore d'autres actions
12          break;
13
14      default:
15          /* Actions si variableATester
16             n'est egale a aucune des valeurs */
17          break;
18  }
```

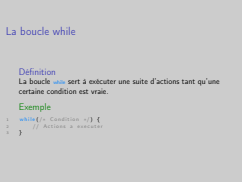
La boucle while

Définition

La boucle while sert à exécuter une suite d'actions tant qu'une certaine condition est vraie.

Exemple

```
1  while(/* Condition */) {
2      // Actions a executer
3  }
```

La boucle for

Définition

Une boucle for est comme une boucle while à ceci près qu'il exécute une action au début, et à chaque *tour* en plus. Ces actions sont séparées par des **points-virgules**.

Exemple

```
1  for(/* Action initiale */ ;
2                  /* Condition */ ;
3                      /* Action recurrente */) {
4      // Actions a executer
5  }
```


La boucle for
Définition
Une boucle for est comme une boucle while à ceci près qu'il exécute une action au début, et à chaque tour en plus. Ces actions sont séparées par des points-virgules.
Exemple

Qu'est-ce qu'une classe ?

Définition

Une classe est un modèle pour créer des objets qui fournit des valeurs initiales pour état (attributs, champs) et implémente un comportement (méthodes, fonctions).

Qu'est-ce qu'une classe ?
Définition
Une classe est un modèle pour créer des objets qui fournit des valeurs initiales pour état (attributs, champs) et implémente un comportement (méthodes, fonctions).

Qu'est-ce qu'une classe ?

Définition

Une classe est un modèle pour créer des objets qui fournit des valeurs initiales pour état (attributs, champs) et implémente un comportement (méthodes, fonctions).

Exemple

Un voiture peut être représentée comme un objet car elle possède un état initial (neuf), et des comportements (rouler, changer de vitesse, klaxonner, allumer les phares) qui vont ou non changer son état.

Qu'est-ce qu'une classe ?
Définition
Une classe est un modèle pour créer des objets qui fournit des valeurs initiales pour état (attributs, champs) et implémente un comportement (méthodes, fonctions).
Exemple
Un voiture peut être représentée comme un objet car elle possède un état initial (neuf), et des comportements (rouler, changer de vitesse, klaxonner, allumer les phares) qui vont ou non changer son état.

Créer une classe

Une classe est représentée par le mot-clef `class`.

Créer une classe
Une classe est représentée par le mot-clef class.

Créer une classe

Une classe est représentée par le mot-clef `class`.
Le mot -clef `public` implique qu'elle soit utilisable autre part que dans son propre paquet. L'autre possibilité est de ne **rien** mettre.

Créer une classe

Une classe est représentée par le mot-clef class.
Le mot -clef public implique qu'elle soit utilisable autre part que dans son propre paquet. L'autre possibilité est de ne rien mettre.

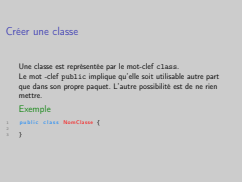
Pas possible de mettre `private` ou `protected`

Créer une classe

Une classe est représentée par le mot-clef `class`.
Le mot -clef `public` implique qu'elle soit utilisable autre part que dans son propre paquet. L'autre possibilité est de ne rien mettre.

Exemple

```
1  public class NomClasse {
2
3  }
```

Les attributs

Par sécurité (le principe de l'*encapsulation*), **tout** les attributs de la classe doivent être mis en private ou protected pour empêcher l'accession des attributs en dehors de la classe. Cela évite une mauvaise utilisation d'une classe.

Exemple

```
1  public class MaClasse {
2      private int unAttribut;
3      protected static boolean unAutreAttribut;
4      // public est deconseille
5      public static String encoreUnAttribut;
6  }
```


Les personne utilisant classe obligé de passer par méthodes qui vérifient si l'utilisateur fait pas de conneries

Les méthodes I

Parmi les méthodes particulières, il y a les *getters*, qui ont pour utilité de récupérer un attribut à partir d'une classe extérieure, les *setters*, qui ont pour utilité de changer la valeur d'un attribut à partir d'une classe extérieure.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└Les attributs et les méthodes
└Les méthodes

Les méthodes I

Parmi les méthodes particulières, il y a les *getters*, qui ont pour utilité de récupérer un attribut à partir d'une classe extérieure, les *setters*, qui ont pour utilité de changer la valeur d'un attribut à partir d'une classe extérieure.

Les méthodes II

Il existe aussi les *tests* qui sont des méthodes qui renvoie une valeurs booléenne. (Ex : isEmpty() de la classe String)

Les méthodes II

Il existe aussi les *tests* qui sont des méthodes qui renvoie une valeurs booléenne. (Ex : isEmpty() de la classe String)

Les méthodes III

Exemple

```
1  public class maClasse {
2      public boolean estNulle() {
3          return true;
4      }
5
6      public static void ditBonjour() {
7          System.out.println("Bonjour !");
8      }
9  }
```


Les tableaux à une dimension I

Définition

Les tableaux correspondent à une suite d'éléments d'un type particulier stockés côte à côte dans la mémoire. Un tableaux doit avoir une longueur définie au départ et ne peux **pas** changer. On accède à ses éléments via un indice, commençant par **0**.

Les tableaux à une dimension I

Définition
Les tableaux correspondent à une suite d'éléments d'un type particulier stockés côte à côte dans la mémoire. Un tableaux doit avoir une longueur définie au départ et ne peux pas changer. On accède à ses éléments via un indice, commençant par 0.

- Dans la boucle for, c'est un < car i ne doit pas atteindre length
- System.out.print non pas println pour afficher les mots sur une seule ligne

Les tableaux à une dimension II

Exemple

```
1  // Un tableau de 6 entiers
2  int[] monTableau = new int[6]
3  // On change son troisieme element
4  monTableau[2] = 3;
5
6  // Un tableaux de Strings deja definis
7  String[] unAutreTableau = {"Bonjour", "tout", "le", "monde"};
8  // On change le premier element
9  unAutreTableau[0] = "Salut";
10 // On affiche la chaine
11 for(int i = 0 ; i < unAutreTableau.length ; ++i) {
12     System.out.print(unAutreTableau[i] + "␣");
13 }
```


Les tableaux à plusieurs dimensions

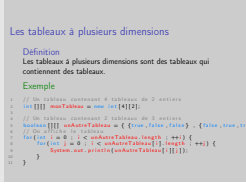
Définition

Les tableaux à plusieurs dimensions sont des tableaux qui contiennent des tableaux.

Exemple

```java
// Un tableau contenant 4 tableaux de 2 entiers
int[][] monTableau = new int[4][2];

// Un tableau contenant 2 tableaux de 3 entiers
boolean[][] unAutreTableau = { {true,false,false} , {false,true,true}};
// On affiche le tableau
for(int i = 0 ; i < unAutreTableau.length ; ++i) {
    for(int j = 0 ; i < unAutreTableau[i].length ; ++j) {
        System.out.println(unAutreTableau[i][j]);
    }
}
```

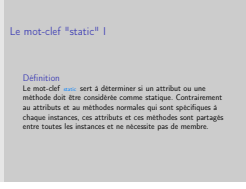



Le mot-clef "static" I

Définition

Le mot-clef static sert à déterminer si un attribut ou une méthode doit être considérée comme statique. Contrairement au attributs et au méthodes normales qui sont spécifiques à chaque instances, ces attributs et ces méthodes sont partagés entre toutes les instances et ne nécessite pas de membre.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└Le mot-clef "static"
└Le mot-clef "static"



Le mot-clef "static" II

Exemple

Si dans la classe Voiture on veut créer un attribut *nombresDeVoitures*, il ne doit pas être particulier à chaque voitures mais partagé. Si aucune voiture n'existe, le champ *nombreDeVoitures* existe et vaut 0.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└Le mot-clef "static"
└Le mot-clef "static"
Le mot-clef "static" II
Exemple
Si dans la classe Voiture on veut créer un attribut *nombresDeVoitures*, il ne doit pas être particulier à chaque voitures mais partagé. Si aucune voiture n'existe, le champ *nombreDeVoitures* existe et vaut 0.

Le mot-clef "static" III

Exemple

```java
public class Voiture {
    // Les champs statiques respectent aussi l'encapsulation
    protected static int nombreDeVoitures;

    public static int getNombreVoitures() {
        return nombreDeVoitures;
    }

    public Voiture() {
        // A chaque creation de voitures
        // le champ nombreDeVoitures augmente
        ++nombreDeVoitures;
    }
}
```


L'héritage I

Définition

Une classe A elle hérite d'une classe B ou A est une classe fille de B, implique que A possède tout les attributs et les méthodes de la classe B. La classe fille peut accéder directement aux éléments public et protected de la classe mère (mais pas les éléments private) et peu changer le comportement des méthodes héritées.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└L'héritage
└L'héritage

L'héritage I
Définition
Une classe A elle hérite d'une classe B ou A est une classe fille de B, implique que A possède tout les attributs et les méthodes de la classe B. La classe fille peut accéder directement aux éléments public et protected de la classe mère (mais pas les éléments private) et peu changer le comportement des méthodes héritées.

L'héritage II

Exemple

Une voiture est un véhicule. Cela implique qu'une voiture peut faire tout ce que peut faire un véhicule. Cela implique que toutes les caractéristiques d'un véhicules sont aussi des caractéristiques d'une voiture. Dans la programmation orienté objet, la classe Voiture hérite de la classe Véhicule.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└L'héritage
└L'héritage
L'héritage II
Exemple
Une voiture est un véhicule. Cela implique qu'une voiture peut faire tout ce que peut faire un véhicule. Cela implique que toutes les caractéristiques d'un véhicules sont aussi des caractéristiques d'une voiture. Dans la programmation orienté objet, la classe Voiture hérite de la classe Véhicule.

L'héritage III

Exemple

```
1  public class Vehicule {
2      protected int nombreDeRoues;
3
4      protected int coordoneesX;
5      protected int coordoneesY;
6
7      public void allerA(int x, int y) {
8          coordoneesX = x;
9          coordoneesY = y;
10     }
11 }
12
13 public class Voiture extends Vehicule {
14     protected int nombreDeRoues = 4;
15 }
```


Les collections

Définition

Une collection est un objet qui a pour but de stocker d'autres objets. Il peuvent faire office de tableaux mais peuvent aussi augmenter et réduire en taille et contenir des objets de type différents.

2014-04-01
Une vraie introduction à Java
└L'orienté objet
└Les collections
└Les collections

Les collections
Définition
Une collection est un objet qui a pour but de stocker d'autres objets. Il peuvent faire office de tableaux mais peuvent aussi augmenter et réduire en taille et contenir des objets de type différents.

Les collections

Définition

Une collection est un objet qui a pour but de stocker d'autres objets. Il peuvent faire office de tableaux mais peuvent aussi augmenter et réduire en taille et contenir des objets de type différents.

Il y a globalement trois catégories de collections :

- Les List
- Les Map
- Les Set

Les collections
Définition
Une collection est un objet qui a pour but de stocker d'autres objets. Il peuvent faire office de tableaux mais peuvent aussi augmenter et réduire en taille et contenir des objets de type différents.
Il y a globalement trois catégories de collections :
- Les List
- Les Map
- Les Set

Les List I

Définition

Les List opèrent comme des tableaux extensibles, rétractables et pouvant contenir plusieurs. Ils ont un index entier. Il y a par exemple les *Vector*, les *LinkedList* et les *ArrayList*.

2014-04-01
Une vraie introduction à Java
L'orienté objet
Les collections
Les List
Les List I
Définition
Les List opèrent comme des tableaux extensibles, rétractables et pouvant contenir plusieurs. Ils ont un index entier. Il y a par exemple les *Vector*, les *LinkedList* et les *ArrayList*.

Les List II

Exemple

```
1  import java.util.LinkedList;
2  import java.util.List;
3  import java.util.ListIterator;
4  // ...
5
6  List maListe = new LinkedList();
7  maListe.add(true);
8  maListe.add(42);
9  maListe.add("souhaitabilite");
10
11 // On parcourt la liste
12 for(int i = 0 ; i < maListe.size() ; ++ i)
13     System.out.println(maListe.get(i));
14
15 // Ou
16 ListIterator li = maListe.listIterator();
17 while(li.hasNext());
18     System.out.println(li.next());
```


Les Map I

Définition

Les Map sont des conteneurs à système clef/valeurs. Chaque élément possède sa clef qui doit être unique, mais pas forcément ordonnée. Il y a parmi eux les objets *Hashtable*, *HashMap*, *TreeMap*, etc. . .

Les Map I
Définition
Les Map sont des conteneurs à système clef/valeurs. Chaque élément possède sa clef qui doit être unique, mais pas forcément ordonnée. Il y a parmi eux les objets *Hashtable*, *HashMap*, *TreeMap*, etc. . .

Les Map II

Exemple

```java
import java.util.Enumeration;
import java.util.Hashtable;
// ...

// Une map qui a un nombre a virgule associe un String
Hashtable<float,String> monConteneur = new Hashtable<float,String>();
monConteneur.put(2.3,"deux virgule trois");
monConteneur.put(4.2,"quatre virgule deux");
monConteneur.put(1.337,"un virgule trois-cent trente sept");

// On le parcourt
Enumeration e = monConteneur.elements();
while(e.hasMoreElements())
    System.out.println(e.nextElement());
```


Les Map II
Exemple

Les Set I

Définition

Les Set sont des conteneurs dont les éléments n'ont pas d'index associés. Ils n'acceptent pas de doublons. Il y a parmi eux les objets *HashSet*, *TreeSet*, *LinkedHashSet*, etc. . .

Les Set I

Définition
Les Set sont des conteneurs dont les éléments n'ont pas d'index associés. Ils n'acceptent pas de doublons. Il y a parmi eux les objets *HashSet*, *TreeSet*, *LinkedHashSet*, etc. . .

Les Set II

Exemple

```java
import java.util.HashSet;
import java.util.Iterator;
// ...

HashSet monConteneur = new HashSet();
monConteneur.add("A");
monConteneur.add(2);
monConteneur.add("mains");

// On le parcourt
Iterator it = monConteneur.iterator();
while(it.hasNext())
    System.out.println(it.next());
```


Les packages

Définition

Les packages Java sont une manière d'organiser les classes d'un programme qui appartiennent à une même catégorie ou ont des même fonctionnalités. On les utilise avec des dossiers et en les spécifiant au début du fichier avec le mot-clef package. Les classes **publiques** d'un package extérieur nécessitent d'être importées.

Les packages

Définition
Les packages Java sont une manière d'organiser les classes d'un programme qui appartiennent à une même catégorie ou ont des même fonctionnalités. On les utilise avec des dossiers et en les spécifiant au début du fichier avec le mot-clef package. Les classes **publiques** d'un package extérieur nécessitent d'être importées.

Les packages

Définition

Les packages Java sont une manière d'organiser les classes d'un programme qui appartiennent à une même catégorie ou ont des même fonctionnalités. On les utilise avec des dossiers et en les spécifiant au début du fichier avec le mot-clef package. Les classes **publiques** d'un package extérieur nécessitent d'être importées.

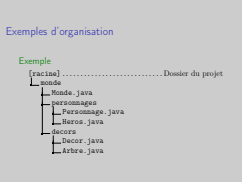
Les packages

Définition
Les packages Java sont une manière d'organiser les classes d'un programme qui appartiennent à une même catégorie ou ont des même fonctionnalités. On les utilise avec des dossiers et en les spécifiant au début du fichier avec le mot-clef package. Les classes **publiques** d'un package extérieur nécessitent d'être importées.

Exemples d'organisation

Exemple

```
[racine] ..... Dossier du projet
└── monde
    ├── Monde.java
    ├── personnages
    │   ├── Personnage.java
    │   └── Heros.java
    └── decors
        ├── Decor.java
        └── Arbre.java
```

Convention de Java veut que toutes les classes soient dans un package => aucune classe dans le dossier racine

Exemple de fichiers

Exemple

Heros.java

```java
package monde.personnages;

public class Heros extends Personnage {
    // ...
```

Monde.java

```java
package monde;
import monde.personnages.Heros;
import monde.decors.Arbre;

public class Monde {
    // ...
```


La Javadoc I

Définition

La Javadoc est un outil utilisé dans les commentaires des fichiers Java. Ils commencent par `/**` et terminent comme un commentaire avec `*/`.

La Javadoc I
Définition
La Javadoc est un outil utilisé dans les commentaires des fichiers Java. Ils commencent par /** et terminent comme un commentaire avec */.

La Javadoc II

Il existe 9 tags javadoc (*@qqch*)

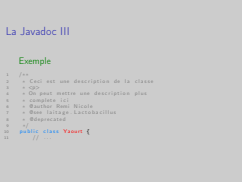
- @param
- @return
- @throws
- @author
- @version
- @see
- @since
- @serial
- @deprecated

2014-04-01
Une vraie introduction à Java
Notions utiles
La Javadoc
La Javadoc
La Javadoc II
Il existe 9 tags javadoc (@qqch)
@param
@return
@throws
@author
@version
@see
@since
@serial
@deprecated

La Javadoc III

Exemple

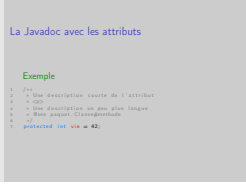
```
1  /**
2   * Ceci est une description de la classe
3   * <p>
4   * On peut mettre une description plus
5   * complete ici
6   * @author Remi Nicole
7   * @see laitage.Lactobacillus
8   * @deprecated
9   */
10 public class Yaourt {
11     // ...
```

La Javadoc avec les attributs

Exemple

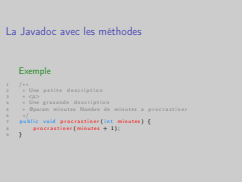
```
1  /**
2   * Une description courte de l'attribut
3   * <p>
4   * Une description un peu plus longue
5   * @see paquet.Classe#methode
6   */
7  protected int vie = 42;
```

La Javadoc avec les méthodes

Exemple

```
1  /**
2   * Une petite description
3   * <p>
4   * Une graaande description
5   * @param minutes Nombre de minutes a procrastiner
6   */
7  public void procrastiner(int minutes) {
8      procrastiner(minutes + 1);
9  }
```

Liens utiles

- http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java
- http://docs.oracle.com/javase/search.html

Liens utiles
- http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java
- http://docs.oracle.com/javase/search.html