JAVA 11

JÉRÉMY PERROUAULT

INTRODUCTION

INTRODUCTION

INTRODUCTION

- Java11 a apporté les changements suivants
  - Dernier Oracle JDK gratuit (commercial sans licence)
  - Simplification de « run » des programmes
  - Inférence de type pour les expressions lambda
  - Amélioration des **`String`**
  - Amélioration des **`Predicate`**
  - Amélioration des **`Optional`**
  - Classes imbriquées et visibilité des attributs
  - Suppression des modules JEE, JavaFX et CORBA

EXECUTION

SIMPLIFICATION

EXECUTION

- Avec Java11, il est possible d'utiliser la commande java directement sur un fichier .java
  - Avant
    - javac Demo.java
    - java Demo
  - Maintenant
    - java Demo.java

INFERENCE

VAR DANS LES LAMBDA

INFERENCE

- Possibilité d'utiliser le mot-clé « var » dans l'expression lambda
  - On pouvait utiliser le typage dans les versions précédentes
  - Sans typage, impossible de préfixer le paramètre par une annotation par exemple
  - On ne peut pas mixer inférence et type, lorsqu'on utilise l'inférence il faut le faire pour tous les paramètres

STRING

NOUVEAUTÉS JAVA 11

STRING

- De nouvelles fonctionnalités sont ajoutées
  - `isBlank` Vérifie si la chaine est vide (espaces compris)
  - `lines` Permet de couper la chaine sur les sauts de ligne en **`List<String>`**
  - `strip` Supprime les « espaces » avant et après la chaine
    - Meilleur que `trim` puisqu'il prend en charge les caractères de retour, l'Unicode
  - `stripLeading` Supprime les « espaces » avant la chaine
  - `stripTrailing` Supprime les « espaces » après la chaine
  - `repeat` Concatène la chaine avec n répétitions

```
String demo = " dé mo ";

System.out.println(demo.strip().replace(" ", "@"));
System.out.println(demo.stripLeading().replace(" ", "@"));
System.out.println(demo.stripTrailing().replace(" ", "@"));
```

PREDICATE

NOUVEAUTÉS JAVA 11

PREDICATE

- De nouvelles fonctionnalités sont ajoutées
  - `not` Méthode statique pour inverser le résultat d'un **`Predicate`**

```java
List<String> fruits = Arrays.asList("Pomme", "\n \n", "Banane", " ");

fruits.stream()
  .filter(f -> !f.isBlank())
  .forEach(System.out::println);
```

```java
List<String> fruits = Arrays.asList("Pomme", "\n \n", "Banane", " ");

fruits.stream()
  .filter(Predicate.not(String::isBlank))
  .forEach(System.out::println);
```

OPTIONAL

NOUVEAUTÉS JAVA 11

OPTIONAL

- De nouvelles fonctionnalités sont ajoutées
  - `isEmpty` Vérifie si l'élément est null (inverse de `isPresent`)

CLASSES

CLASSES IMBRIQUÉES

CLASSES IMBRIQUÉES

- Les classes imbriquées peuvent maintenant accéder aux attributs de classe « principale »
  - Sans artifices générés

```
public class NestedClass {
  private static int x = 5;

  public static void doPrint() {
    System.out.println(x);
  }

  public static class NestedTest {
    public static void nestedDoPrint() {
      System.out.println(x);
    }
  }
}
```

CLASSES IMBRIQUÉES

- En compilant le code précédent avec Java 9 ou 10, puis avec Java 11
  - javac NestedClass.java
- Puis en analysant le bytecode avec l'outil de désassemblage javap
  - javap –v NestedClass.class
  - javap –v NestedClass$NestedTest. class

- On se rend compte qu'avec Java 11, l'attribut « x » est considéré comme un attribut
  - Accessible dans la classe « principale »
  - Contrairement aux versions précédentes qui génèrent un artifice d'accès « access$000 »

CLASSES IMBRIQUÉES – JAVA 9

- NestedClass.class

```
static int access$000();
  descriptor: ()I
  flags: (0x1008) ACC_STATIC, ACC_SYNTHETIC
  Code:
    stack=1, locals=0, args_size=0
       0: getstatic     #1                  // Field x:I
       3: ireturn
    LineNumberTable:
      line 3: 0
```

- NestedClass$NestedTest.class

```
public static void nestedDoPrint();
  descriptor: ()V
  flags: (0x0009) ACC_PUBLIC, ACC_STATIC
  Code:
    stack=2, locals=0, args_size=0
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: invokestatic  #3                  // Method fr/formation/NestedClass.access$000:()I
       6: invokevirtual #4                  // Method java/io/PrintStream.println:(I)V
       9: return
    LineNumberTable:
      line 12: 0
      line 13: 9
```

CLASSES IMBRIQUÉES – JAVA 11

- NestedClass.class
  - Rien de particulier, pas de méthode « access$000 »

- NestedClass$NestedTest.class

```
public static void nestedDoPrint();
  descriptor: ()V
  flags: (0x0009) ACC_PUBLIC, ACC_STATIC
  Code:
    stack=2, locals=0, args_size=0
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: getstatic     #3                  // Field fr/formation/NestedClass.x:I
       6: invokevirtual #4                  // Method java/io/PrintStream.println:(I)V
       9: return
    LineNumberTable:
      line 12: 0
      line 13: 9
```