Chapitre 12
Annotations

Meta données (syntaxiques) ajoutées au code source java qui seront exploitées par le compilateur ou la machine virtuelle

Annotation

- Annotation = meta-tag

- ***Programmation déclarative***
  - Le programmeur dit ce qu'il faut faire
  - et des outils génèrent le code pour le faire
- Le compilateur ou la machine virtuelle
  - extraient, à partir des annotations,
  - des informations sur le comportement du programme

Déclaration d'un type d'annotation

```
public @interface AnnotationName
        {  … method1( ) … ;
           … method2( ) … ;
           … method3( ) … ;
           …
        }
```

Déclarations de méthodes

Utilisation d'annotations

Pour annoter un code

```
@AnnotationName (  method1 = value1,
                   method2 = value2,
                   method3 = value3,   …  )
```

Code qui peut être annoté

- Package
- Classe, interface, annotation, …
- Variable d'instance
- Constructeur
- Méthodes
- Variables
- Arguments

Catégories d'annotations

- **Marker : sans élément**
- **Single-element annotation**
- **Multi-value annotation**

Annotation de type Marker

Marker : sans élément

Exemple : public *@interface* Marker {
}

Annotation de type Marker

Exemples d'utilisation

@Marker → *Annote une classe*

```
public class ClassX {
```

@Marker → *Annote une variable d'instance*

```
    private int number;
```

@Marker → *Annote un constructeur*

```
    public ClassX (int number)      { setNumber(number) ;}
```

@Marker → *Annote une méthode*

```
    public void methodY(int number)  { … }

}
```

Annotation single element

Single element annotation

Exemple :

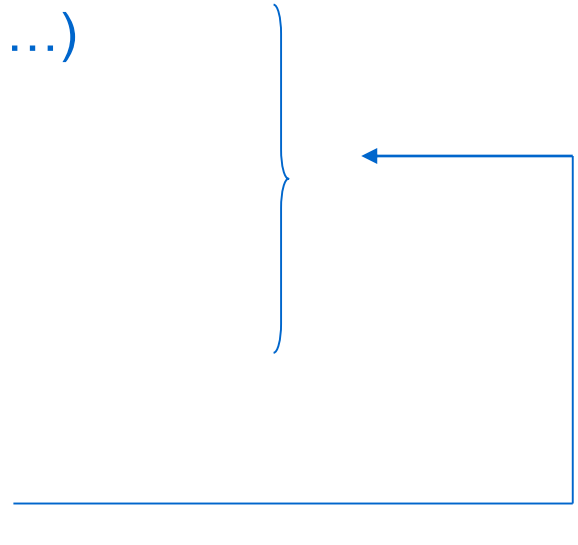
```
public @interface SingleValueAnnotation

{
        String  value( )   default   "bidon" ;
}
```

facultatif

Règles de déclaration des méthodes

- Pas de paramètre
- Pas de clause throws
- Types de retour : uniquement
  - Types primitifs (int, float, char, …)
  - String
  - Class
  - Enumeration

  - + Tableau de types ci-dessus



Annotation single element

Exemples d'utilisation

```
@SingleValueAnnotation  (value  =  "test" )
public class ClassX {

        @SingleValueAnnotation ("test")
        private int number;

        @SingleValueAnnotation ("test")
        public ClassX (int number) { … }

        @SingleValueAnnotation ("test")
        public void methodY (int number)  { … }
}
```

Pas obligatoire si annotation "single-element"

Annotation multi-value

Multi-value annotation

Exemple :

Soit l'énumération:

```
public enum MonthEnum
        { JANUARY, MARCH, MAY, JULY, SEPTEMBER, NOVEMBER }


public @interface MultiValueAnnotation {
        public int hourValue( ) default 24;
        public String whenValue( ) default "now";
        public String[ ] dayValues( ) default { "Monday,Tuesday" };
        public MonthEnum monthValue( ) default MonthEnum.JULY;
}
```

Annotation multi-value

Exemples d'utilisation

```
@MultiValueAnnotation
public class ClassX {

        @MultiValueAnnotation ( hourValue = 12 )
        private  int  number;

        @MultiValueAnnotation ( dayValues = { "friday", "saturday", "sunday" } )
        public ClassX (int number) { … }

        @MultiValueAnnotation ( dayValues = { "friday", "saturday", "sunday" },
                                monthValue =  MonthEnum.MAY)
        public void methodY(int number) { … }
}
```

Annotations build-in

- **Annotations simples**
- **Meta-annotations**

Annotations simples

Annotations simples

- **Override**
- **Deprecated**
- **SuppressWarnings**

Annotation Override

Une méthode annotée *Override* doit redéfinir une méthode de la super-classe

Exemple :

```
public class ClassX {
        @Override public String toString( )
                                { return … ; }
}
```

Si la méthode annotée *Override* n'existe pas dans la super-classe

⇨ il y a erreur à la compilation

Annotation Deprecated

Exemple :

```
public class ClassX {
        @Deprecated   public  int  doSomething( )
                { return … }
}
```

Si appel de la méthode doSomething( ) :

à la **compilation** :

avertissement signalant que la méthode est déconseillée car obsolète

Annotation SupressWarnings

Permet de supprimer des avertissements à la compilation

Exemple :

```
public class Principal {
        @SuppressWarnings ( { "deprecation" } )
        public static void main(String[ ] args)
                { ClassX classX = new ClassX( );
                  … classX.doSomething( ) ;
                }
}
```

Deprecated

Si la méthode doSomething( ) de la classe ClassX est "deprecated"

⇨ compilateur n'affichera pas d'avertissement

Meta-Annotations

Meta-Annotations

= Annotation de type d'annotation

- **Target**
- **Retention**
- **Documented**
- **Inherited**

Meta-annotation Target

Pour préciser sur quel type d'élément l'annotation peut porter

Types d'éléments :

- ElementType.**TYPE**
- ElementType.**FIELD**
- ElementType.**METHOD**
- ElementType.**PARAMETER**
- ElementType.**CONSTRUCTOR**
- ElementType.**LOCAL_VARIABLE**
- ElementType.**ANNOTATION_TYPE**

Meta-annotation Target

Exemple :

import java.lang.annotation.*;

```
@Target ( { ElementType.CONSTRUCTOR, ElementType.METHOD } )
public @interface SingleValueAnnotation {
        public String value( ) default "bidon";
        }
```

Si on tente d'appliquer cette annotation à d'autres éléments qu'un constructeur ou une méthode, il y a *erreur à la compilation*

Meta-annotation Target

Exemples d'utilisation

```
@SingleValueAnnotation
public class ClassX {
        @SingleValueAnnotation ("test")
        private int number;

        @SingleValueAnnotation ("test")
        public ClassX (int number) { … }

        @SingleValueAnnotation ("test")
        public void methodY (int number)  { … }
}
```

→ *Erreur à la compilation*

→ *Erreur à la compilation*

OK

Meta-annotation Retention

Permet de préciser le niveau auquel est "retenue" l'annotation

Trois niveaux

RetentionPolicy.**SOURCE**

*L'annotation sera retenue uniquement au niveau **code source** et ignorée par le compilateur*

RetentionPolicy.**CLASS**

*L'annotation sera retenue par le compilateur à la **compilation** mais ignorée par la machine virtuelle*

RetentionPolicy.**RUNTIME**

*L'annotation sera retenue par la **machine virtuelle** et ne pourra être lue qu'à l'exécution*

Meta-annotation Retention

Exemple:

```
import java.lang.annotation.*;


@Retention ( RetentionPolicy.RUNTIME )
public @interface SingleValueAnnotation {
        public String value( ) default "bidon";
        }
```

Meta-annotation Documented

Permet d'inclure l'annotation dans la javadoc

Exemple :

```
@Documented
public @interface SingleValueAnnotation {
        public String value( ) default "bidon";
        }
```

Meta-annotation Inherited

Seulement pour les annotations sur des **classes**

Si une annotation A annotée @Inherited porte sur une classe C,

alors toute sous-classe de la classe C en héritera,

c'est-à-dire, sera également annotée par l'annotation A

Meta-annotation Inherited

Exemple :

```
@Inherited
public @interface SingleValueAnnotation {
        public String value( ) default "bidon";}
```

Exemple d'utilisation

```
@SingleValueAnnotation
public class ClassX { … }
```

⇨ Pour toute sous-classe de ClassX :

```
@SingleValueAnnotation
public class SubClassX extends ClassX { … }
```

JavaDoc

Les commentaires placés entre

```
/**
*
*
*/
```

seront repris dans la JavaDoc (générée automatiquement)

Annotations utilisables dans la JavaDoc

Documentation d'une classe

@author

@version

@since

@see *Suggestion de consulter d'autres sources, …*

ex: *@see #field* *(si dans même classe)*

@see #Constructor(type, type, …)

@see #method(type, type, …)

@see Class *(si dans même package)*

@see package.Class

@see package.Class#field

@see package.Class#method(type, type, …)

Annotations utilisables dans la JavaDoc

Documentation d'une méthode/constructeur

@param *paramName*	*Informations sur les paramètres*
@return	*Informations sur la valeur de retour*
@throws	*Informations sur les exceptions détectées*
@see	
…	