**Commandes en Java - Annotations**

ATTENTION :

|  |  |  |
| --- | --- | --- |
| *Commentaires :*  //Commentaire 1  /\* Commentaire 2 \*/ | *Instructions JavaDoc :*  /\*\* instruction  \* dites JavaDoc  \*@return  \* qui commencent par une minuscules après @  \*/ | *Annotation Java :*  @Return |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbole | Signification | Code | Utilisation |  |
| Annotations marqueurs | = annotations standards | java.lang.annotation |  |  |
| @Override | indique que la méthode est redéfinie via le principe d'héritage | @Override  public String toString() {  } |  | ATTENTION : doit être utilisée UNIQUEMENT sur des méthodes héritées et redéfinies |
| @Deprecated | indique que l'élément ne devrait plus être utilisé  si l'élément est utilisé, Eclipse le signale par un warning | @Deprecated  public void maMethode() { |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Méta-annotation | servent à annoter nos annotations personnalisées |  |  |  |
| @Documented | inclus l'annotation dans la JavaDoc lors de sa génération | @Documented  public @interface MonAnnotation {  } |  |  |
| @Inherit | les classes dérivant d'une classe mère contenant des annotations hériteront également des annotations qui seront anotée par @Inherit | @Inherited  public @interface MonAnnotation {  } | @MonAnnotation  public class ClasseMere {  }  public class ClasseFille extends ClasseMere {  } | cette classe fille héritera de l'annotation car elle est annotée @Inherited |
|  |  |  |  |  |
|  |  |  |  |  |
| Annotations paramétrées |  |  |  |  |
| @SupressWarning( ) | indique au compilateur d'ignorer les éléments générant le type de warning indiqué en paramètre  ignore :  tous les warning  annotation dépréciée  élément déprécié  oubli d'un break dans bloc switch  bloc finally douteux  serialVersionUID manquant  redéfinition de cstr ss super()  code inutilisé | @SupressWarning("all")  all  boxing  cast  dep-ann  deprecation  fallthrough  finally  serial  super  unused | @SupressWarning("cast", "deprecation") | ATTENTION :   * à utiliser avec parcimonie pour ne pas ignorer des erreurs importantes * le compilateur Java ne prend pas les mêmes paramètres que le compilateur d'Eclipse |
|  |  |  |  |  |
| @Retention( ) | définit une durée de vie pour l'annotation | @Retention(RetentionPolicy.CLASS)  public @interface MonAnnotation {  }  @Retention(RetentionPolicy.RUNTIME)  @Retention(RetentionPolicy.SOURCE) | défaut, annotation enregistrée dans le fichier .class, mais non utilisable par la JVM  enregistrée dans le fichier .class et utilisable lors de l'exécution du prog grâce à l'introspection  non enregistrée dans le fichier .class, utilisable uniquement par les outils sachant lire les fichiers sources (utilitaire JavaDoc, compilateur, …) |  |
| @Target | permet de cibler sur quel type d'élément(s) l'annotation peut être utilisée  tous type de base ou personnalisé  champs d'une classe (= variables)  autres annotations | @Target(ElementType.PACKAGE)  public @interface MonAnnotation {  }  PACKAGE  TYPE  CONSTRUCTOR  METHOD  PARAMETER  FIELD  LOCAL\_VARIABLE  ANNOTATION\_TYPE | *Avec plusieurs paramètres :*  @Target(ElementType.TYPE, ElementType.FIELD)  public @interface MonAnnotation {  }  *Dans le fichier package-info.java :*  @PackAnnotationA  @PackAnnotationB  package mon.package; | ATTENTION :  Les annotations sur le Package se font dans un fichier spécifique nommé ***package-info.java*** à placer dans le package concerné |
|  |  |  |  |  |
|  |  |  |  |  |
| Annotations multiparamétrées |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Annotation Personnalisée |  |  |  |  |
| @interface | créer une annotation personnalisée | public @interface MonAnnotation {  } | public @interface MonAnnotation {  String monChampA();  String monChampB() default "valeur par défaut";  NIVEAU niveau() default NIVEAU.MINEUR;  int[] tableauEntier();  char[] tableauCaractere() default {'A', 'D'};  MonAnnotationB annotation();  Class maClasse() default Double.class;  } | seuls les léléments suivant peuvent être utilisés dans les annotations :   * primitifs (int, double, boolean, char, …) * String * enum * Class * autres annotations |
|  |  |  | @MonAnnotation(  monChampA = "valeur champA",  annotation = @MonAnnotationB,  tableauEntier = { 0 }  )  public void maMethode() { } | ATTENTION : lors de l'utilisation de l'annotation, il faut, à minima, attribuer des valeurs aux éléments qui n'en ont pas par défaut |
|  | annotation monoparamétrée personnalisée | public @interface MonAnnotation1Param {  Type value();  } | @MonAnnotation1Param(valeurParametre); | Rq: si l'annotation ne comporte qu'un paramètre, en le nommant value(), on évite d'avoir à préciser son nom lors de l'appel à l'annotation |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |