Université Nice Sophia Antipolis

Les annotations en Java

F. Mallet

Université Nice Sophia Antipolis

Les annotations

F. Mallet

miage.m1@gmail.com

http://deptinfo.unice.fr/~fmallet/

Les annotations

- Apparues avec Java2 1.5 (Tiger)
- Annoter des éléments du langage pour
  - Documenter le code source
  - Favoriser l'introspection
  - Ajouter des opérations avant la compilation
    - Nouvel outil: **A**nnotation **P**rocessing **T**ool
- Utilisables à la compilation ou à l'exécution
  - Différent des tag javadoc `@author, @return, …`

Utilisation

- Devant l'élément marqué: **`@Annotation`**
  - Package
  - Class, Interface, Enum
  - Annotation
  - Constructeur, Méthode, Paramètre
  - Champ, Variable

- Exemple

```
@Annotation
class MaClasse { … }
```

Les annotations standards (1/3)

- Déprécié : **`@Deprecated`**
  - Eléments qui ne devraient plus être utilisés: classes, méthodes, constructeur, champ, …
  - Remplace le tag javadoc @deprecated
    - Mais sans expliquer la raison
    - Possible d'utiliser conjointement les deux

```
@Deprecated
public Date() { … }
```

Les annotations standards (2/3)

- Méthode redéfinie : **`@Override`**
  - Si la méthode ne redéfinit pas effectivement une méthode, une erreur est levée par le compilateur ou IDE (Eclipse)

```
class A {
  void m() { … }
}
class B extends A {
  @Override
  void m() { … }
}
```

Les annotations standards (2/3)

- Méthode redéfinie : **`@Override`**
  - Si la méthode ne redéfinit pas effectivement une méthode, une erreur est levée par le compilateur ou IDE (Eclipse)

```
class A {
  void m() { … }
}
class B extends A {
  @Override
  void M() { … }
}
```

Les annotations standards (2/3)

- Méthode redéfinie : **`@Override`**
  - Si la méthode ne redéfinit pas effectivement une méthode, une erreur est levée par le compilateur ou IDE (Eclipse)

```
class A {
  void m() { … }
}
class B extends A {
  @Override
  void m(int a) { … }
}
```

Les annotations standards (3/3)

- Cache les avertissements: **`@SuppressWarnings`**
  - Par exemple, si on utilise une classe générique sans définir le type.

```
class A {
   void m(ArrayList liste) { … }
}
```

  - ArrayList is a raw type. References to generic type ArrayList<E> should be parameterized

Les annotations standards (3/3)

- Cache les avertissements: **`@SuppressWarnings`**
  - Par exemple, si on utilise une classe générique sans définir le type.

```
class A {
   @SuppressWarnings("unchecked")
   void m(ArrayList liste) { … }
}
```

La première annotation (1/2)

- L'annotation **`@interface`**

- Exemple de déclaration

  public @interface PremiereAnnotation { }

- Exemple d'utilisation

```
public class Foo {
  @PremiereAnnotation
  public void method() { … }
}
```

La première annotation (2/2)

- L'annotation **`@interface`**

- Exemple de déclaration

  public @interface PremiereAnnotation { }

- Exemple d'utilisation

```
@PremiereAnnotation
public class Foo {
   public void method() { … }
}
```

Annotation @TODO

- Déclaration simple

```
public @interface TODO { }
```

- Déclaration avec un paramètre String

```
public @interface TODO {
  String value();
}
```

- Exemple d'utilisation

```
@TODO(value="ajouter un constructeur")
public class Foo {
}
```

Annotation @TODO

- Value est optionnel
  - Si value est le seul paramètre, alors il est **optionnel**
- Déclaration avec un paramètre String

```
public @interface TODO {
  String value();
}
```

- Exemple d'utilisation

```
@TODO("ajouter un constructeur")
public class Foo {
}
```

Annotation @TODO

- Déclaration simple

```
public @interface TODO { }
```

- Déclaration avec un paramètre String

```
public @interface TODO {
  String value();
}
```

- Types de retour autorisés pour value()
  - Type primitif, String, Class, Enum
  - java.lang.annotation.Annotation
  - Tableau d'un type autorisé

Annotation @TODO

- Déclaration avec plusieurs champs

```
public @interface TODO {
  String value();
  Level level();

  static enum Level {MINOR,NORMAL,URGENT};
}
```

- Exemple d'utilisation

```
@TODO(value="à modifier", level=TODO.Level.NORMAL)
void bar() { … }
```

Annotation @TODO

- Déclaration avec plusieurs champs

```
public @interface TODO {
  String value();
  Level level();

  static enum Level {MINOR,NORMAL,URGENT};
}
```

- Exemple d'utilisation

```
import static TODO.Level.NORMAL;
…
@TODO(value="à modifier", level=NORMAL)
void bar() {… }
```

Annotation @TODO

- Déclaration avec plusieurs champs

```
public @interface TODO {
  String value();
  String[] responsables();
}
```

- Exemple d'utilisation de tableaux

```
@TODO(value="à modifier",
  responsables={"Michel","Philippe"})
void bar() {… }
```

Annotation @TODO

- Déclaration avec plusieurs champs

```
public @interface TODO {
  String value();
  String[] responsables();
}
```

- Exemple d'utilisation de tableaux

```
@TODO(value="à modifier",
  responsables="Michel")
void bar() {… }
```

Annotation @TODO

```
public @interface TODO {
  String value();
}
```

- Interdit d'appliquer plusieurs fois la même annotation

```
@TODO("à modifier")
@TODO("changer le nom")
void bar() {… }
```

Annotation @TODOs

```
@interface TODO { String value(); }
@interface TODOs { TODO[] value(); }
```

- Mais on peut faire un tableau d'annotations

```
@TODOs({
   @TODO("changer le nom"),
   @TODO("à modifier")
})
void bar() {… }
```

Annotation @TODOs

```
@interface TODO { String value(); }
@interface TODOs { TODO[] value(); }
```

- Mais on peut faire un tableau d'annotations

```
@TODOs(value = {
   @TODO(value = "changer le nom"),
   @TODO(value = "à modifier")
})
void bar() {… }
```

Les méta-annotations

- Paquetage **`java.lang.annotation`**
  - @Documented
    - Indique que l'annotation doit être documentée par javadoc et les outils compatibles
  - @Inherited
    - L'annotation est hérité par l'élément qui hérite de l'élément annoté
  - @Retention
    - Où l'annotation doit être maintenue (source, .class, JVM)
  - @Target
    - Limite le type des éléments sur lequel l'annotation peut s'appliquer

Annotation @Documented

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Documented {
}
```

- Usage

```
@Documented
@interface Foo { }
```

Annotation @Inherited

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Inherited {
}
```

- Application

```
@Inherited
@interface Foo { }
```

Annotation @Inherited

- Application

```
@Inherited
@interface Foo { }
```

- Usage
  - Si une classe A est annotée

```
@Foo
class A { … }
```

  - Alors, les sous-classes de A héritent automatiquement de l'annotation

```
class B extends A { … }
```

Annotation @Retention

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Retention {
  RetentionPolicy[] value();
}

public enum RetentionPolicy {
 SOURCE,     // pas incluse dans le .class
 CLASS,      // dans le .class, mais pas JVM
 RUNTIME     // disponible par introspection
}
```

Annotation @Target

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Target {
  ElementType[] value();
}

public enum ElementType {
 ANNOTATION_TYPE, CONSTRUCTOR, FIELD,
 LOCAL_VARIABLE, METHOD, PACKAGE,
 PARAMETER, TYPE
}
```

Annotation @Target

```
@Target(ElementType.ANNOTATION_TYPE)
public @interface Target {
  ElementType[] value();
}

@Target({FIELD, METHOD})
public @interface Foo {
  String value();
}
```

Annotation et introspection

- Méthodes de la classe **`Class`**
  - <A extends Annotation>
    getAnnotation(Class<A> annotation);
  - Annotation[ ] getAnnotations();
  - Annotation[ ] getDeclaredAnnotations();
  - boolean isAnnotation();
  - boolean
    isAnnotationPresent(Class<? extends Annotation> c);

Annotation et introspection

- Exemple

```
@Deprecated
@TODO{"A Modifier"}
class A { … }
```

@TODO(value=A Modifier)
@java.lang.Deprecated()

- Introspection

```
class<A> c = A.class;
Annotation[] as =
                    c.getAnnotations();
for(Annotation a : as)
  System.out.println(a);
```

Annotation et introspection

- Exemple

A Modifier

```
@Deprecated
@TODO{"A Modifier"}
class A { … }
```

- Introspection

```
class<A> c = A.class;
TODO todo =
        c.getAnnotation(TODO.class);
System.out.println(todo.value());
```

ANNOTATION PROCESSING TOOL

Annotation Processing Tool

- Lors de la compilation ou à la volée !
- Deux utilisations complémentaires
  - Aide à l'édition/compilation
    - Règles de développement (commentaires…)
    - Affiche des messages (Warning, Error) à partir du source Java
  - Génération de code
    - Ne peut PAS modifier le code source
    - Peut générer des classes pour compléter le code initial (Adaptateur…)

Exemple: `@Override`

- Méthode redéfinie : **`@Override`**
  - Si la méthode ne redéfinit pas effectivement une méthode, une erreur est levée par le compilateur ou IDE (Eclipse)

```
class A {
  void m() { … }
}
class B extends A {
  @Override
  void M() { … }
}
```

com.sun.mirror.apt.**AnnotationProcessor**

- Documentation
  - http://download.oracle.com/javase/1.5.0/docs/guide/apt/ (JSR 175)
- Compilation
  - Implémentation livrée avec le JDK mais pas dans le CLASSPATH par défaut
  - $JAVA_HOME/lib/tools.jar
  - IDE-specific: plugin org.eclipse.jdt.apt.core
- API
  - Interface avec une seule méthode (à redéfinir)
  - **`void process()`**
- Quel processeur pour quelle annotation ?

com.sun.mirror.apt.AnnotationProcessorFactory

- Déclare les annotations supportées

```
public Collection<String> supportedAnnotationTypes() {
   return Collections.singleton("miage.m1.MyAnnotation");
}
```

- Déclare les options supportées

```
public Collection<String> supportedOptions() { return null;}
```

- Créé un AnnotationProcessor adapté en fonction de l'annotation rencontrée

```
public AnnotationProcessor getProcessorFor(
                    Set<AnnotationTypeDeclaration> atDecls,
                   AnnotationProcessorEnvironment env) {
  return miage.m1.MyAnnotation(env);
}
```

Exemple d'AnnotationProcessor

```
class MyAnnotationProcessor implements AnnotationProcessor {
  private AnnotationProcessorEnvironment _env;

  MyAnnotationProcessor(AnnotationProcessorEnvironment env) {
    super();
    _env = env;
  }

  @Override
  public void process() {
    // traite l'annotation
  }
}
```

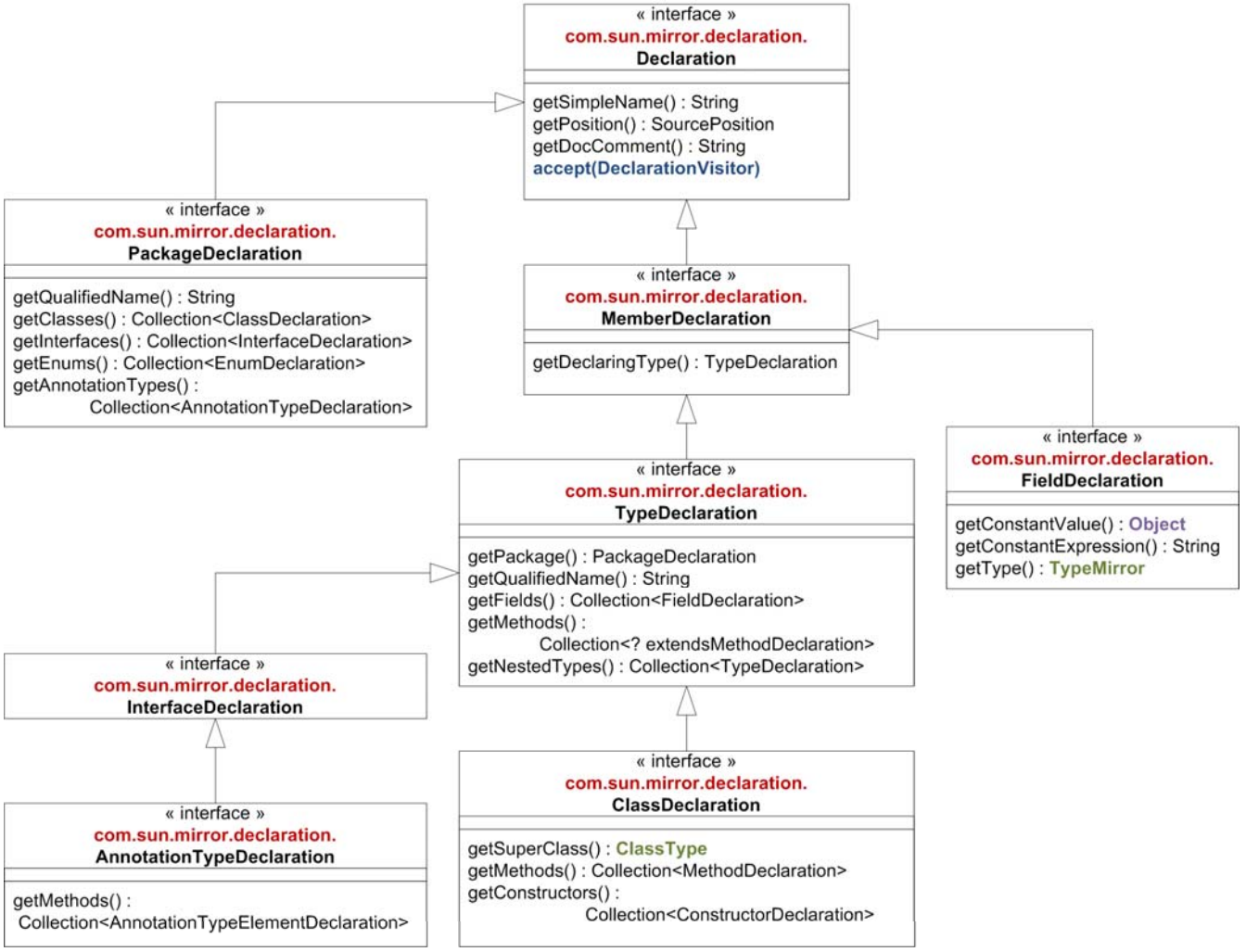
Exemple d'AnnotationProcessor

```
class MyAnnotationProcessor implements AnnotationProcessor {
  // constructeur, champs, …
  @Override
  public void process() {
    AnnotationTypeDeclaration annotDecl =
    (AnnotationTypeDeclaration)_env.getTypeDeclaration(
                                          "miage.m1.MyAnnotation");
    if (annotDecl==null) return;

    Collection<Declaration> decls =
                    _env.getDeclarationsAnnotatedWith(annotDecl);
    if (decls == null) return;

    for(Declaration declaration : decls) {
      // traite la déclaration annotée
    }
}
```

Declaration vs. Type



com.sun.mirror.util.DeclarationVisitor

- Tout visiteur doit implémenter l'interface

```
interface DeclarationVisitor {
  visitDeclaration(Declaration);
  visitMemberDeclaration(MemberDeclaration);
  visitTypeDeclaration(TypeDeclaration);
  visitClassDeclaration(ClassDeclaration);
  visitFieldDeclaration(FieldDeclaration);
  …
}
```

- Adaptateur standard vide
  - com.sun.mirror.util.**SimpleDeclarationVisitor**

Pour afficher des messages

```
class MyAnnotationProcessor implements AnnotationProcessor {
  // constructeur, champs, …
  @Override
  public void process() {
    AnnotationTypeDeclaration annotDecl =
    (AnnotationTypeDeclaration)_env.getTypeDeclaration(
                                          "miage.m1.MyAnnotation");
    if (annotDecl==null) return;

    Collection<Declaration> decls =
                    _env.getDeclarationsAnnotatedWith(annotDecl);
    if (decls == null) return;

    for(Declaration declaration : decls) {
      declaration.accept(new MyVisitor(_env.getMessenger()));
    }
}
```

Exemples de Visiteur

- Affiche simplement le nom des déclarations

```
class MyVisitor extends SimpleDeclarationVisitor{
  public void visitDeclaration(Declaration d) {
    System.out.println("Decl:" + d.getSimpleName());
  }
}
```

Exemples de Visiteur

- Affiche un warning si le nom d'une classe commence par une minuscule

```
import com.sun.mirror.apt.Messager;
class MyVisitor extends SimpleDeclarationVisitor{
  Messager messager;
  MyVisitor(Messager messager) {
    this.messager = messager;
  }
  public void visitClassDeclaration(ClassDeclaration d) {
    if (Character.isLowerCase(d.getSimpleName().getChar(0)))
      messager.printWarning(d.getPosition(),
           "Class names should start with a capital letter");
  }
}
```

Pour générer du code

```
class MyAnnotationProcessor implements AnnotationProcessor {
  // constructeur, champs, …
  @Override
  public void process() {
    AnnotationTypeDeclaration annotDecl =
    (AnnotationTypeDeclaration)_env.getTypeDeclaration(
                                          "miage.m1.MyAnnotation");
    if (annotDecl==null) return;

    Collection<Declaration> decls =
                    _env.getDeclarationsAnnotatedWith(annotDecl);
    if (decls == null) return;

    for(Declaration declaration : decls) {
      generate(_env.getFiler()));
    }
}
```

Pour générer du code

```
class MyAnnotationProcessor implements AnnotationProcessor {
  // …
  private void generate(Filer filer) {
    PrintWriter pw = filer.createSourceFile("Essai.java");
    pw.println("class Essai {");
    pw.println("  static public void main(String[] args) {");
    pw.println("    System.out.println(\"Fichier généré.\");");
    pw.println("  }");
    pw.println("}");
    pw.close();
  }
}
```

- Le code généré
  - Est généré dans le bon dossier (package)
  - Est compilé et peut contenir des annotations

Utilisation des annotations

- Outil apt (disponible avec JDK)
  - `apt -factorypath { où se trouve MyFactory.class }`
  - `-factory miage.m1.MyFactory`
  - `-s {répertoire où générer les .java}`
  - `-d {répertoire où générer les .class}`
  - `-cp {chemin pour la compilation}`
  - `{fichiers Java à compiler avec processeur}`

Sources

- Tutoriels de Sun:
  - http://download.oracle.com/docs/books/tutorial/java/javaOO/annotations.html
  - http://download.oracle.com/javase/1.5.0/docs/guide/language/annotations.html
  - http://download.oracle.com/javase/1.5.0/docs/guide/apt/