Projet P21-P22
SAÉ 21

Année 2022-2023

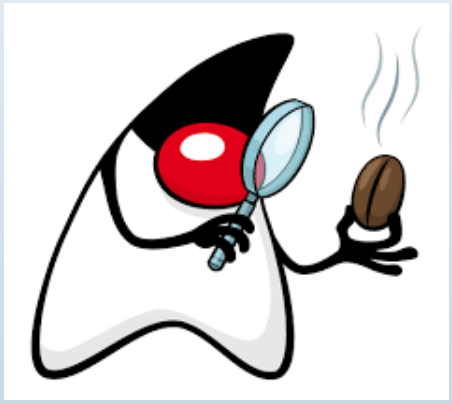
R. Schneider

IUT Robert Schuman

Organisation

- travail en binôme
  (formé dans un groupe de TP)
- durée : 6 semaines
- horaire : environ 6h /semaine (+ heures en autonomie)
  - P21-P22 :
    - **TD – 8h** (2+2+2+2+0+0) ;
    - **TP – 16h** (4+4+2+2+2+2)
  - SAE21 :
    - **Tutorat**:
      10h (0+0+2+2+4+2)
- évaluation
  - rendus hebdomadaires
  - présentation finale : 12 et 13 juin 2023

Sujet du projet



L'objectif du projet est de générer automatiquement
le source PlantUML de diagrammes de classes UML
pour des éléments (classes ou interfaces)
qui figurent dans un ou des packages
et/ou un ou plusieurs fichiers sources Java.

Les diagrammes générés sont des DCA et des DCC.

Pour produire le source PlantUml des deux types de diagrammes,
le projet nécessite la définition et la réalisation

- d'une API,
- et d'une commande qui utilise cette API,

Production de diagrammes de classes UML à partir du code source de classes Java.

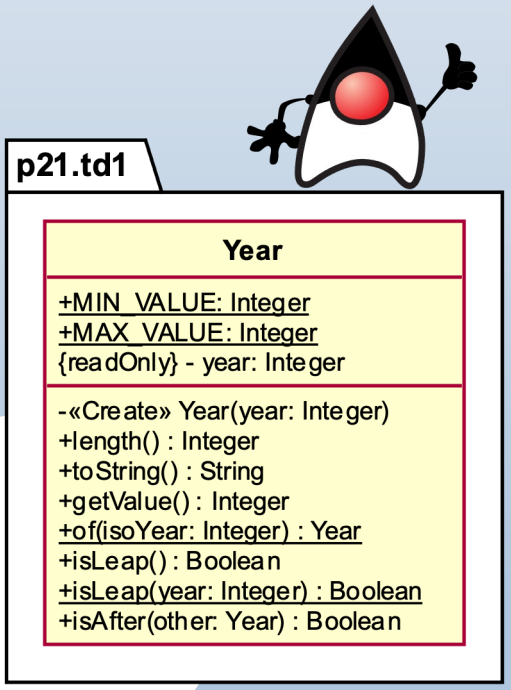
```
package p21.td1;

public class Year
{
   public static final int MIN_VALUE = -999_999_999;
   public static final int MAX_VALUE = 999_999_999;

   private final int year;

   public static boolean isLeap(long year)
   {

      ....
```



Comment ?

Problème 1 : connaître les informations sur les packages et les classes en Java

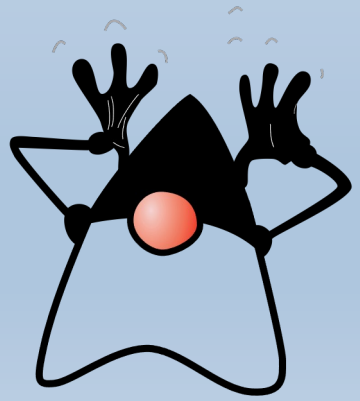
une solution : Language Model API

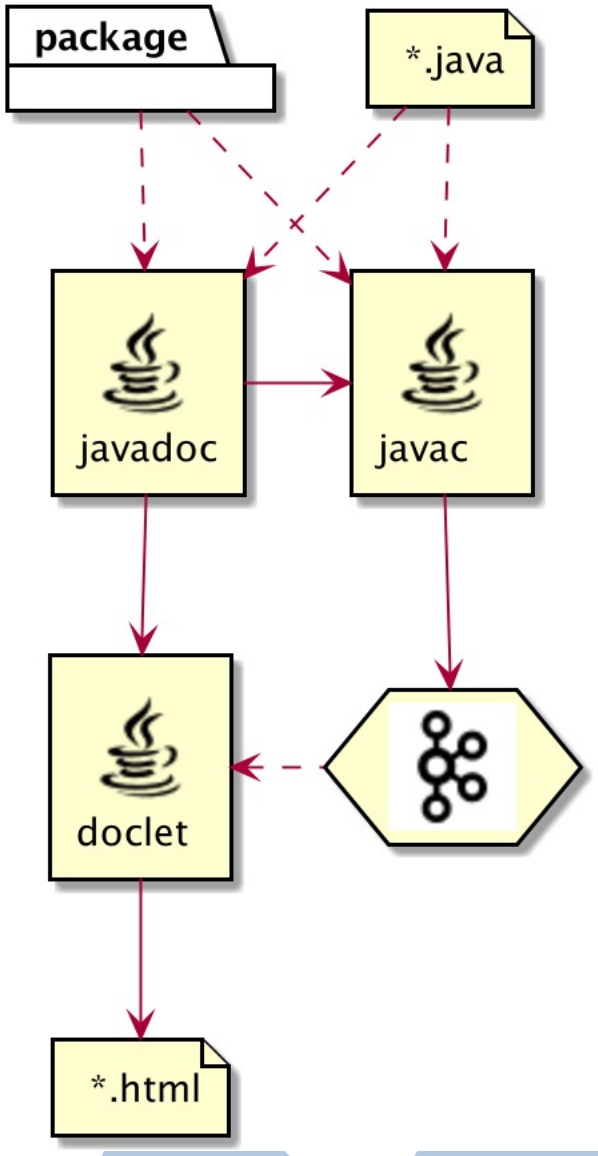
- permet d'analyser les éléments d'un programme Java.

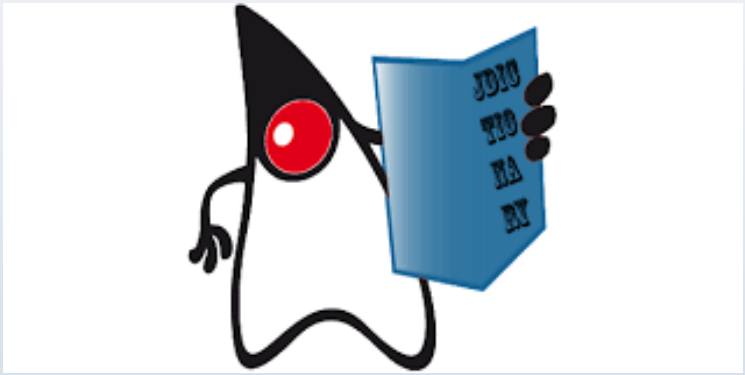
Problème 2 : accèder à l'API Java Language Model

une solution : javadoc

- outil de production de la documentation à partir des commentaires de documentation et d'annotations ;







javadoc

Fonctionnement : *The main javadoc tool.*

It uses the javac front end to parse and analyze source and class files to be documented,

and then invokes a nominated doclet to process the elements and their documentation comments.

- donne accès à l'API Language Model par l'intermédiaire des **doclets**.

doclet

permet de fournir du code externe exécuté par javadoc (plug-in).

Doclets are invoked by javadoc.
For example, the standard doclet is invoked by default, to generate HTML documentation. The standard doclet generates HTML documentation based on the doc comments.

The invocation is defined by the interface Doclet
-- the run interface method, defines the entry point.

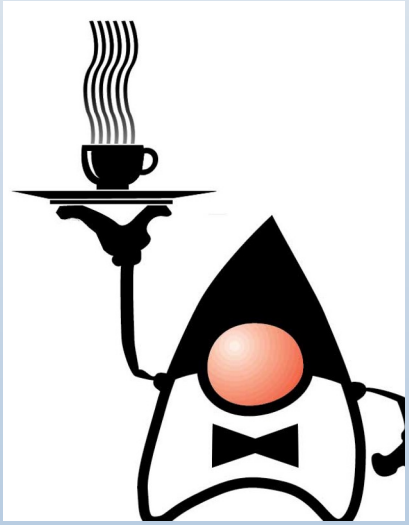
```
public boolean run(DocletEnvironment environment)
```

The DocletEnvironment instance holds the environment that the doclet will be initialized with. From this environment all other information can be extracted, in the form of elements.

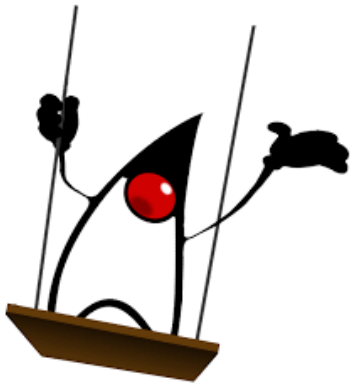
One can further use the APIs and utilities described by Language Model API to query Elements and Types.

Quels livrables ?

1. une `API (pumlFromJava)` à définir et écrire

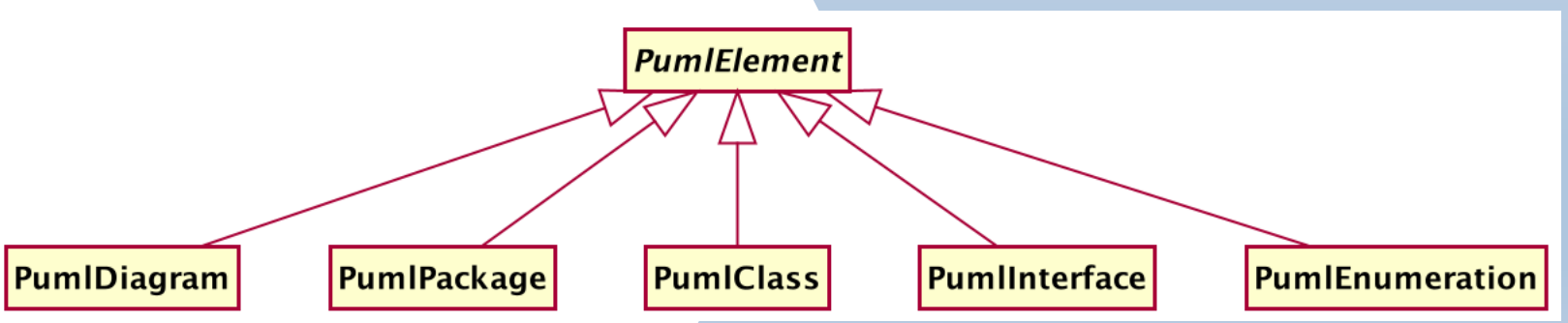


- L'API est un package Java.
- L'API définit une classe pour chaque élément du langage UML
- Chaque classe de l'API fournit le code PlantUML qui représente l 'élément dans le cadre d'un DCA et d'un DCC.
- Les éléments UML sont construits sur la base des éléments du langage Java



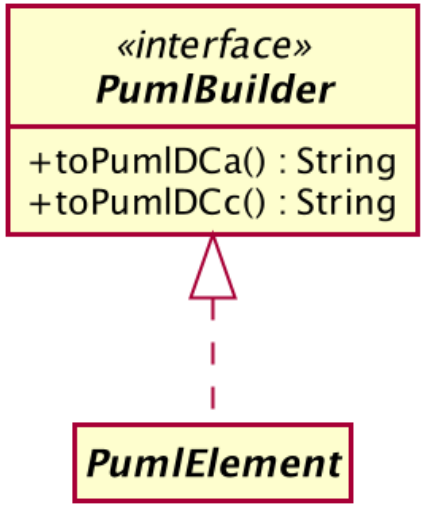
API pumlFromJava

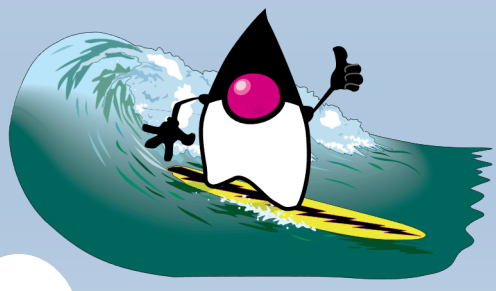
- chaque classe de l'API correspond à un élément du langage UML.



API pumlFromJava

- chaque élément de l'API doit être capable de donner son code PlantUML dans le cadre d'un DCA ou d'un DCC

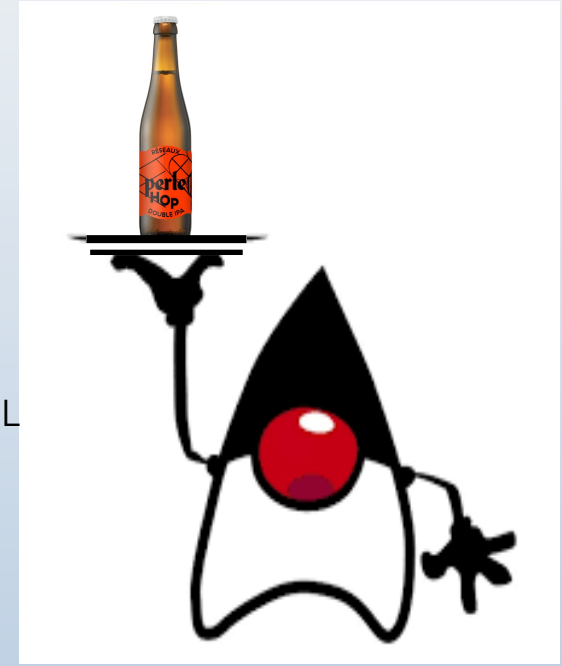




Quels livrables ?

2. commande (Java2Uml)

La commande Java2uml exécute javadoc et produit le code PlantUML à travers un doclet : PumlDoclet



javadoc

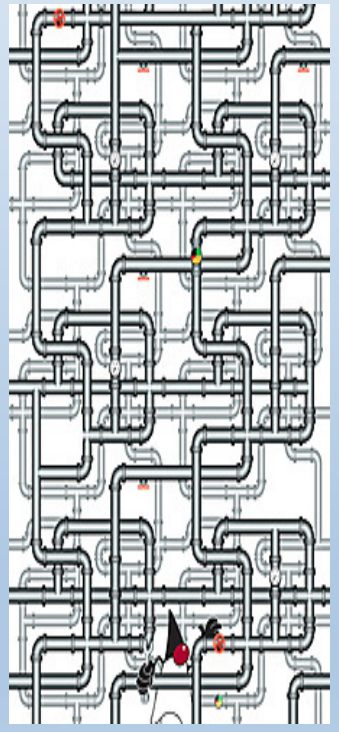
- sélection des packages et fichiers qui contiennent les éléments à intégrer dans le diagramme

javadoc exécute le doclet PumlDoclet

PumlDoclet

- analyse les éléments Java sélectionnés avec l'API Java Language Model
- crée les éléments UML de l'API pumlFromJava pour chaque élément Java
- génère le code PlantUml du diagramme de classes pour les éléments UML.
- écrit le code du diagramme de classes dans un fichier .puml

Développement itératif



Problème 3 :

comment développer ?

Réponse : 6 phases (1 phase par semaine) – git.unistra.fr/P21_Projet

Semaine 1 :

étude de javadoc et des doclets ; mise en place de l 'environnement

Semaine 2 :

étude de l'API *Language Model* ; 1ère maquette : production de DCA sans associations ;

Semaine 3 :

un meilleur DCA : ajout des associations (généralisation, réalisation, association)

Semaine 4 :

un DCC sans associations : opérations, paramètres, types, visibilités, multiplicités, ...

Semaine 5 :

un DCC avec associations : rôle, multiplicités, dépendances, ...

Semaine 6 :

préparation du rendu final et de la présentation

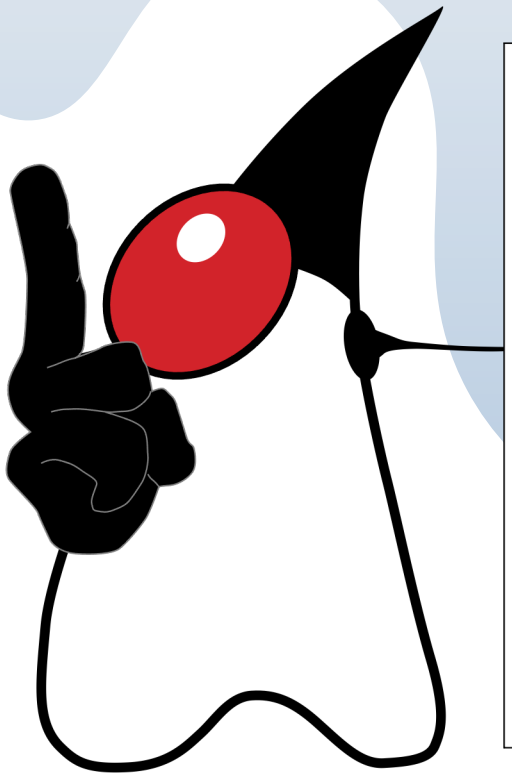
Qualité du développement

- l'accent doit être mis sur le développement orienté objet et la qualité du développement

❖ DOO : Conception et programmation orientées objet

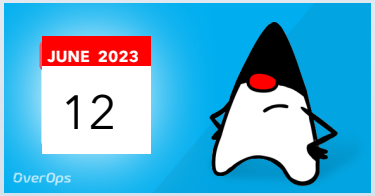
- API : création d'une hiérarchie de classes représentant les éléments UML
- Une partie de l'architecture de l'API est imposée

Développement OO



- Java :
  encapsulation forte (visibilité privée pour les variables d'instance)
- UML:
  un attribut dont le type n'est pas primitif est représenté par un rôle dans une association

Rendus



rendus chaque semaine

- DCA et DCC de l'API pumlFromJava
- autres diagrammes demandés (semaine 1 et semaine 2)
- fichiers sources commentés
- exemples de diagrammes produits (sorties de la commande pour un package d'entrée, par exemple western4) ; fichiers puml et svg.
- rapport hebdomadaire succinct (bref - en quelques lignes) : ce qui est fait - pas fait -, modifications de la semaine, les difficultés éventuelles, index des fichiers, mode d'emploi.

dépôt git

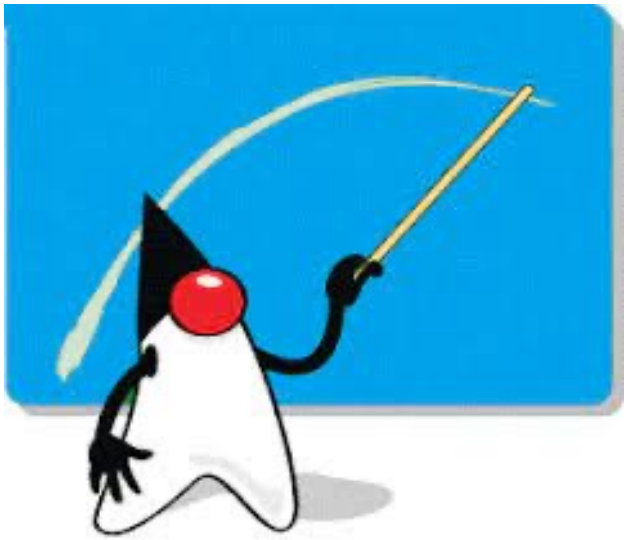
- groupe GIT pour le binôme
- fork du dépôt du projet
- une branche par semaine (nom imposé semaine1, semaine2, ...)
- répertoires: src, uml, exemples, rapports

Evaluation

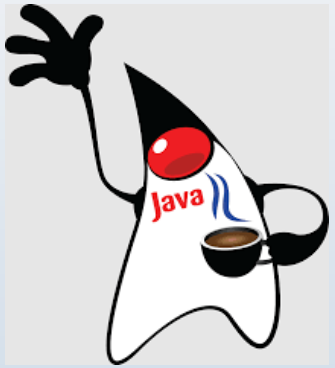
- rendu hebdomadaire
  - feedback rapide sur le rendu précédent
  - évaluation globale des rendus à la fin du projet
- présentation finale : 12 et 13 juin 2023
  - 15 minutes

    (10 minutes de présentation

    + 5 minutes de questions)



Extensions (facultatif)

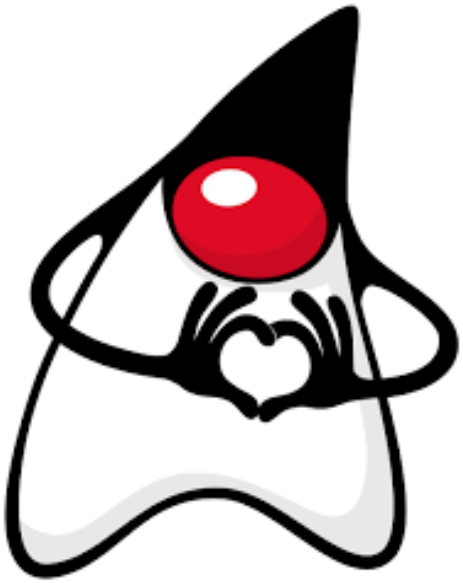


1. Indiquer les redéfinitions dans les opérations
   pour les méthodes marquées @Override

2. Utilisation des tags de documentation pour compléter manuellement
   certains renseignements à porter sur le diagramme de classe
   quand ceux-ci ne peuvent pas se trouver automatiquement dans le code.
   - @pumlType pour les multiplicités
   - @pumlAssociation pour nommage
   - @pumlAggregation et @pumlComposition
   - ...

   Cette extension implique la lecture et l'inspection des commentaires de documentation.

Questions ?



Au revoir Duke

1992 Première apparition de Duke (Star 7 Demo)

1996 Java Release 1.0

1998 Mon premier cours de Java à l'IUT

2023 Mon dernier cours de Java à l'IUT