Développement informatique avancé orienté application - TP

Premières lignes de Java

Virginie Van den Schrieck, Louis Van Dormael

La partie pratique du cours de Développement informatique avancé orienté application (que nous appelerons désormais « cours de Java »par souci de brièveté) se déclinera en deux volets :

1. Des séances de travaux pratiques consistant à implémenter de petites applications Java dans l'environnement de développement Eclipse, afin de vous habituer à travailler avec des outils professionnels
2. Un projet en binôme, en autonomie.

1 Premiers pas avec Eclipse

1.1 Qu'est ce qu'un IDE ?

Définissez le terme avec vos propres mots.

1.2 Quelle est la différence entre un IDE et un éditeur de texte ?

1.3 Démarrer Eclipse

1.3.1 Workspace

Ouvrez Eclipse. Il vous est demandé de sélectionner le workspace. Choisissez le répertoire dans lequel vous souhaitez travailler (sur votre disque G), par exemple dans un sous-dossier `code` de votre dossier consacré au cours de Java. Expliquez en quelques mots ce qu'est un *workspace* :

1.3.2 Les Views

Une fois dans Eclipse, vous pouvez explorer l'environnement de travail. Prenez quelques minutes pour observer la fenêtre de l'IDE. Notez l'existence de différentes zones dans la fenêtre principale. Certaines zones possèdent des onglets permettant de changer leur contenu, appelé « view ». Nous verrons plus tard à quoi chacune va servir. Essayez de déplacer les Views d'une zone à l'autre, de les fermer ou de les ré-ouvrir (menu Window-> Show View). Les Views *Package Explorer* et *Console* vous seront particulièrement utiles.

1.3.3 Cheatsheet

A ce stade, il vous est recommandé de créer un document *trucs et astuces Eclipse* dans lequel vous noterez tout ce qui peut vous être utile pour vous faciliter la vie lorsque vous travaillerez avec cet IDE. Notez par exemple l'emplacement des menus et des commandes que vous avez eu du mal à trouver, afin de ne pas perdre de temps la prochaine fois que vous en aurez besoin. N'hésitez pas non plus à noter les raccourcis clavier les plus intéressants. Gardez ce document ouvert à chaque fois que vous utiliserez Eclipse, pour pouvoir l'exploiter et le compléter à bon escient.

1.3.4 Perspectives

Les Views peuvent être combinées et ré-organisées, mais afin de ne pas s'emmêler les pinceaux, Eclipse propose le concept de Perspective. Une Perspective est un regroupement de Views dédiées à un même usage. Il y a la Perspective Java par défaut, une autre appelée Java Browsing, une Debug... On peut changer de perspective en passant par les onglets en haut à droite. Habituez-vous à changer de perspective, ou encore à ré-initialiser les Views d'une perspective donnée (Reset Perspective). Lorsque vous serez familiarisés avec les bases du Java, il sera utile de tester chaque perspective et de noter dans quelles circonstances elle peut vous être utile.

2 Premier projet

2.1 Créer le projet

A présent que vous vous êtes familiarisés avec l'environnement de travail, il est temps de commencer à l'exploiter. La première étape est de créer un projet. Repérez le menu et l'icône permettant de créer un nouveau projet, que nous appellerons TP1_Java. Sélectionnez « Projet java » comme type de projet. Dans la fenêtre qui s'ouvre, il y a toute une série de paramètres à compléter. Prenez le temps de les examiner une première fois. Notez l'emplacement proposé dans le système de fichier pour le stockage du projet : Vous irez l'examiner avec l'explorateur de fichiers. Vous pouvez également choisir la version de Java à utiliser, en fonction de ce qui est installé sur votre machine. Préférez la version 12. Enfin, vous pouvez choisir le mode de stockage des fichiers sources et des fichiers compilés : séparés ou regroupés. Préférez le stockage séparé.

A l'avenir, pour chaque TP, pensez à bien organiser votre code. Une manière de procéder est par exemple de créer un package par TP, comme nous le verrons ci-dessous.

2.2 Structure interne du projet

Une fois le projet créé, allez explorer le dossier dans lequel il a été placé. Notez le répertoire `src`, dans lequel seront stockés les fichiers comprenant le code source java. Un répertoire `bin` sera créé après la première compilation.

3 Premiers pas en Java

3.1 Création de la classe Hello World

3.1.1 Création du fichier et notion de package

Dans votre projet Java, créez une nouvelle *classe*. Nous verrons plus loin ce qu'est une classe. En l'occurrence, cette classe correspondra à notre premier fichier source Java, dans lequel nous allons écrire nos premières lignes de code. En pratique, Eclipse va faire le plus gros du travail pour nous !

Remarquez le dossier proposé par Eclipse : Il s'agit bien du répertoire `src` dans notre dossier de projet.

Eclipse vous propose de choisir un *package*. Un package est un regroupement de classes (et d'autres fichiers que nous verrons plus tard). Nous pouvons laisser ce champ vide pour sélectionner le package par défaut, mais l'IDE génère un *warning* : il est fortement conseillé de suivre ces recommandations ! Dans ce cas-ci, choisissez `tp1` comme nom de package. Remarquez que si vous écrivez `TP1` en majuscules, un nouvel avertissement est donné. Les

noms de packages doivent, par convention, commencer par une minuscule. Il vous est demandé de toujours suivre ces conventions.

3.1.2 Hello World

Nous allons écrire un programme qui imprime `Hello World` (voir `http://fr.wikipedia.org/wiki/Hello_world`). Nous appellerons donc cette classe `HelloWorld`. Notez les majuscules en début de chaque mot (convention Upper CamelCase pour les noms de classe, voir `http://fr.wikipedia.org/wiki/CamelCase`).

Demandez la création automatique de la *méthode main*. C'est le point de départ de l'exécution du programme.

Demandez également la génération automatique de commentaires. Dans le cadre de ce cours, il vous sera demandé d'être systématiques dans vos commentaires. Nous verrons plus loin qu'un outil permet la génération automatique de documentation sur base de ces commentaires.

Laissez les autres paramètres à leur valeur par défaut. Vous prendrez le temps plus tard de comprendre à quoi chacun correspond.

3.2 Examiner la classe HelloWorld

A présent que le `template` de votre première classe Java est créé, prenez le temps d'examiner l'interface que vous offre Eclipse, similaire à celle montrée à la figure 1 : La View Package Explorer à gauche, qui reprend la hiérarchie de tous les éléments du projet (packages, classes, etc.), la View d'édition de fichier au centre, la View Outline à droite qui reprend la structure du fichier en cours d'édition, et enfin, en bas, la View Problems. Nous n'utiliserons pas la View Task List, vous pouvez la fermer (et y revenir plus tard si elle vous intéresse).

Notez la structure du code généré :

- Un commentaire en début de fichier. Vous pouvez y indiquer ce que contient le fichier (la classe et son but), et tout information relative au fichier (par exemple, la date de création, ou de dernière modification (pas utile si utilisation d'un gestionnaire de version, voir plus tard). Vous pouvez également y ajouter le nom du cours dans le cadre duquel le fichier a été produit.
- Le nom du package auquel appartient le fichier
- Un commentaire de début de classe, qui indiquera le but de la classe.
- Le début de la classe (`public class HelloWorld`)
- La méthode `main`, précédée de son commentaire. Il s'agit du point de départ de l'exécution.
- Un commentaire `TODO` qui indique les emplacements où il faut compléter le code généré.

Notez également les différents types de délimiteurs de commentaires :

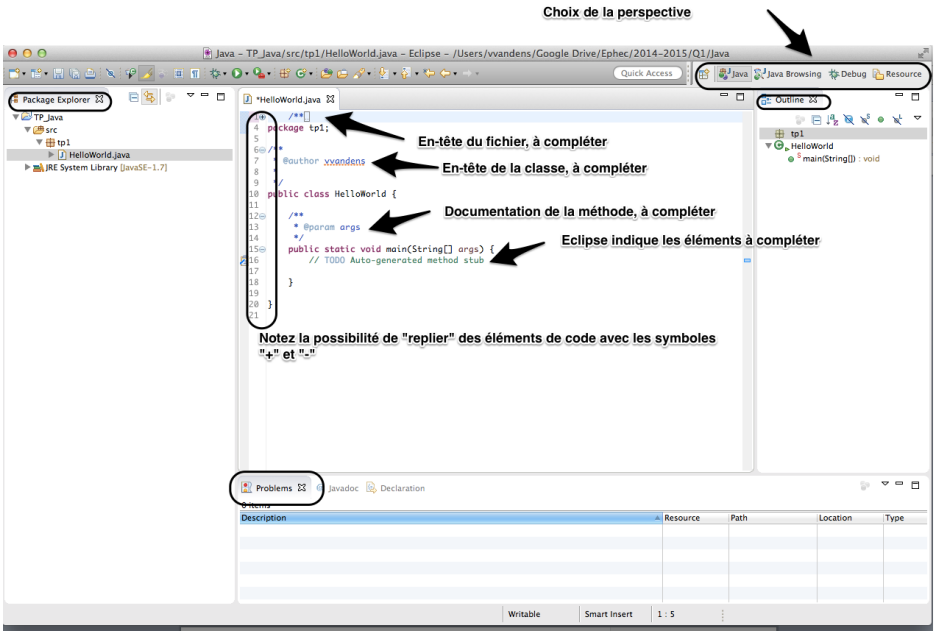

FIGURE 1 – Screenshot de la classe HelloWorld

— `//` : Commentaire « inline » , affiché en vert par Eclipse
— `/* */` : Commentaire « multi-lignes », également affiché en vert (créez-en un pour le vérifier)
— `/** */` : Commentaire « Javadoc », affiché en bleu par Eclipse. L'outil de documentation Javadoc récupère tous ces commentaires pour générer la documentation du fichier. Il est possible d'utiliser des tags HTML pour soigner la mise en page de ces commentaires.

Vous pouvez à présent exploiter la View Javadoc, située dans la zone du bas de la fenêtre Eclipse. Cliquez sur l'onglet Javadoc pour la faire venir au premier plan. A présent, quand vous sélectionnez un élément Java, la documentation que vous avez rédigée apparaît dans cette fenêtre, comme montré sur la figure 2. Vérifiez-le en sélectionnant le nom de la classe HelloWorld, puis le nom de la méthode `main`.

Les curieux pourront en apprendre plus sur la Javadoc ici :

— `http://fr.openclassrooms.com/informatique/cours/presentation-de-la-javadoc` (en français)
— `http://www.edparrish.net/common/javadoc.html` (en anglais)
— `http://www.javapractices.com/topic/TopicAction.do?Id=60` (en anglais)

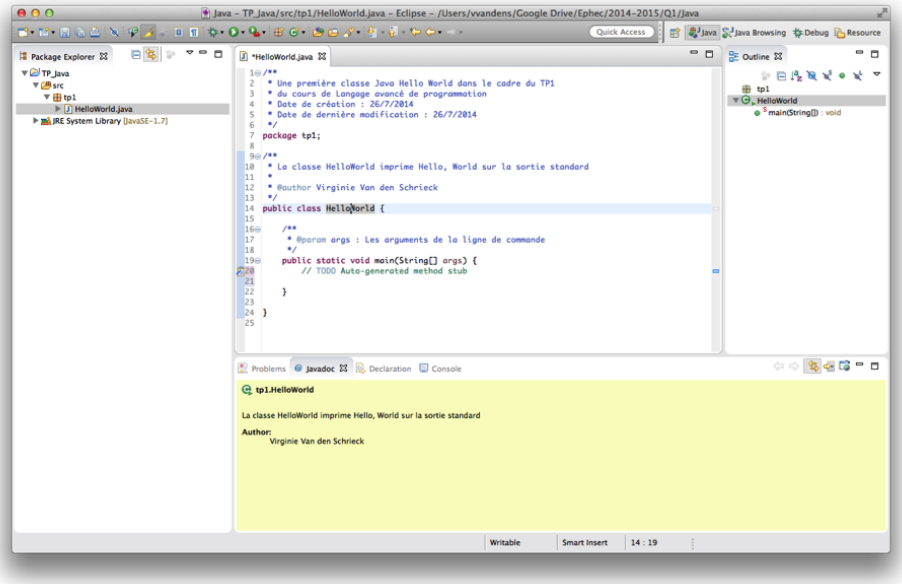

FIGURE 2 – Affichage de la Javadoc dans Eclipse

3.3 Compléter et exécuter la classe HelloWorld

3.3.1 Imprimer à la console

La classe HelloWorld ne contient pas encore de code exécutable. Nous souhaitons imprimer les mots `Hello World`, dans la « console java ». Pour cela, il faut rajouter le code suivant à l'intérieur de la méthode `main` de la classe HelloWorld :

```
System.out.println("Hello World");
```

Nous allons pour le moment interpréter cette ligne de la manière suivante : On demande à l'élément `System` d'imprimer sur une ligne (`println`) la chaîne de caractères `Hello World` , sur son flux de sortie. Nous y reviendrons plus tard pour une explication plus précise. Notez également que les instructions Java se terminent par un `;`, comme en Javascript. Profitez-en pour examiner la javadoc des éléments extérieurs utilisés : System, out, println, ...

La figure 3 montre à quoi ressemble la classe HelloWorld complète, avec code et documentation.

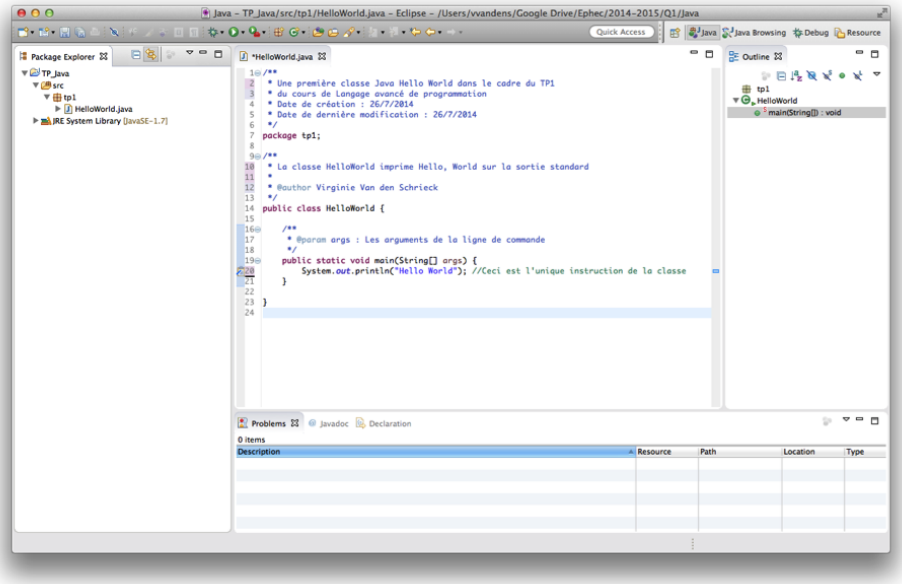

FIGURE 3 – Classe HelloWorld complète

3.3.2 Compilation et exécution

Nous allons à présent compiler et exécuter la classe HelloWorld. Pour cela, utilisez le menu Run => Run, ou bien cliquez sur l'icône ronde verte avec une flèche blanche. Observez la View Console qui passe au premier plan dans la zone du bas. La chaîne de caractères demandée s'y affiche bien. Notez qu'Eclipse vous facilite la tâche, puisqu'avec la seule commande Run, elle compile la classe ET l'exécute, en une seule étape. Retenez bien qu'il s'agit néanmoins de deux opérations séparées, pouvant chacune générer leurs propres erreurs.

Rendez-vous à présent dans le répertoire du projet avec l'explorateur de fichiers. Où se trouve la classe HelloWorld ? Quel est son nom de fichier, et plus précisément, son extension (obligatoire pour toute classe Java) ?

En plus du fichier source, un second fichier contenant la classe HelloWorld compilée a été créé par Eclipse. Dans quel répertoire se trouve-t-il ? Quel est son nom, et son extension ?

3.3.3 Pour les curieux : Compiler et exécuter à la main

Pour compiler et exécuter une classe Java, Eclipse utilise deux exécutables, javac (java compiler) et java (machine virtuelle). Où se trouvent ces deux fichiers ? Indiquez leur chemin d'accès complet :

— javac :

— java :

Utilisez la console Windows pour vous rendre dans votre répertoire de projet Java_TP. Essayez de compiler la classe HelloWorld.java avec l'exécutable `javac`, puis essayez de l'exécuter avec l'exécutable `java`.

Qu'est ce que le PATH ? Comment pouvez-vous le modifier pour utiliser plus facilement les exécutables javac et java ?

Sans utiliser Eclipse, créez une deuxième classe HelloLLN, qui affiche « Hello LLN ». Vous pouvez utiliser n'importe quel éditeur de texte. Compilez le fichier et exécutez la classe depuis la console Windows.

3.3.4 Génération de la Javadoc

Nous allons à présent générer la documentation de notre projet, qui ne contient pour le moment que la classe HelloWorld dans le package TP1. Allez dans « Project => Generate Javadoc » . Dans la fenêtre qui s'ouvre, notez que Eclipse vous indique le chemin d'accès de l'exécutable Javadoc. Laissez toutes les options par défaut. Remarquez également que la documentation sera générée dans le dossier doc de votre projet. Pour les curieux : En cliquant sur Next, vous avez accès à d'autres options. Appuyez sur Finish, et, si la génération s'est déroulé sans problème, allez observer ce qui a été créé dans le répertoire doc.

Dans le répertoire doc, nous trouvons des fichiers html. En ouvrant index.html dans un navigateur, nous pouvons découvrir la documentation générée. Cette documentation concerne la classe HelloWorld. Il n'y a cependant pas de documentation pour le package tp1 qui englobe cette classe. Pour générer la documentation du package, nous devons créer un fichier spécial dans le répertoire du package : `package-info.java`. La figure 4 montre un exemple de fichier package-info.java pour notre package tp1. Créez un tel fichier, puis regénérez la javadoc et observez ce qui a changé.

Notez bien qu'à partir de maintenant, vous DEVEZ générer la javadoc de tout ce que vous produisez (classes et package), avec les commentaires pertinents pour tous les éléments requis (commentaires de classe, de méthode, de champ, . . .)

3.3.5 Utilisation des arguments de la ligne de commande

En Java, il est possible de transmettre des arguments au programme qu'on exécute, en les spécifiant à la suite du nom du programme à exécuter. On les récupère via le tableau de chaînes de caractères args.

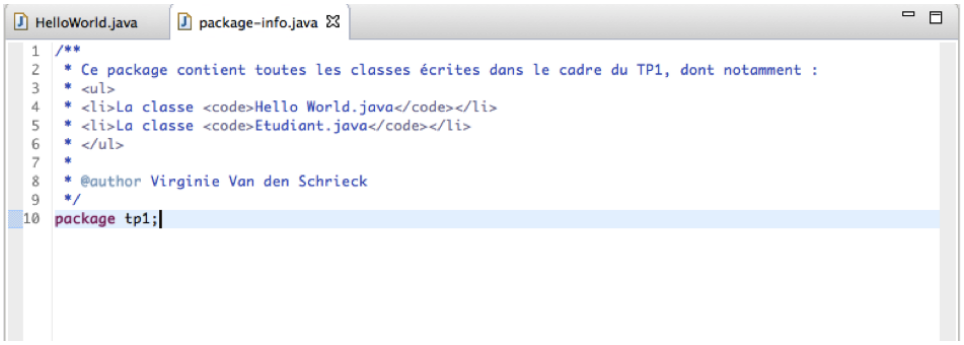

FIGURE 4 – Fichier package-info.java pour documenter le package

Nous allons à présent créer, toujours dans le package tp1, une nouvelle classe appelée PrintArg qui va imprimer sur la sortie le mot transmis en argument. Cette fois, à l'intérieur de la méthode main, on va écrire l'instruction suivante :

```
System.out.println(args[0]);
```

Astuce : L'instruction System.out.println est utilisée très fréquemment. Pour aller plus vite, on peut se contenter d'écrire sysout, puis d'appuyer sur ctrl+espace.

Documentez correctement la classe, puis exécutez-là en passant par le menu Run => Run Configuration. Sélectionnez la configuration pour la classe PrintArg, puis choisissez l'onglet Arguments. Indiquez l'argument dans la zone «Program arguments ». Une fois l'argument indiqué, vous pouvez utiliser le bouton Run. Observez ce qu'il se passe lorsqu'on met un mot en argument, puis deux, puis aucun. Comment l'expliquez-vous ? .

Les curieux qui ont réussi à exécuter la classe HelloWorld depuis la ligne de commande peuvent faire de même avec la classe PrintArg.

4 UML et premières classes

Vous avez découvert au cours théorique les notions de classe, d'objets, d'attribut et de méthode.

Un objet est une entité en mémoire composée d'un ensemble d'attributs auxquels sont associés des valeurs, et auquel on peut appliquer des méthodes qui permettent de manipuler ces attributs. Un objet est décrit par la classe à laquelle il appartient. On peut donc voir une classe Java comme la description des objets qui l'instancieront. Une classe contient donc, d'une part, la définition des attributs des objets qui l'instancient, et d'autre part, l'implémentation des méthodes qu'on peut appliquer à ces objets.

Lors de la phase d'analyse du projet, et dans le cadre de la programmation orientée-objet, il est de bonne pratique d'utiliser des diagrammes UML pour représenter les classes des applications que l'on conçoit. Prenons l'exemple d'une entreprise qui souhaite développer un outil de gestion de son personnel. Le cahier des charges mentionne les éléments figurant dans le dossier des membres du personnel, et il est décidé de créer une classe Personne permettant de représenter cela. La figure 5 vous montre le diagramme UML de cette classe Personne, qui possède deux attributs : Le nom et le prénom. Ce diagramme est actuellement simplifié, nous l'enrichirons au fur et à mesure que des nouvelles notions seront vues.

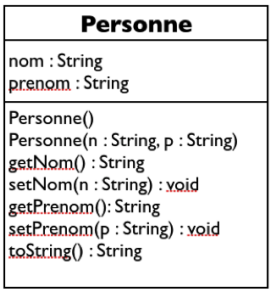

FIGURE 5 – Diagramme UML de la classe Personne

Pour chacun des exercices suivants, prenez bien soin d'ajouter toute la javadoc nécessaire, et dessinez le diagramme UML correspondant.

4.1 Classe Calculatrice

Soit un enseignant qui souhaite développer une calculatrice simplifiée adaptée au niveau de maitrise de ses élèves de niveau primaire. Il souhaite que les élèves puissent se concentrer sur les notions travaillées en classe sans être distraits par des fonctions superflues. De plus, les élèves sont invités à effectuer des calculs à la chaîne, en ajoutant, soustrayant ou élevant au carré depuis le résultat de l'opération précédente.

Sur base de ce Cahier des Charges, nous allons créer une classe Calculatrice, qui simule une telle calculatrice. Cette classe contient une seule variable d'instance : La valeur actuellement affichée sur l'écran de la calculatrice. Toutes les opérations de cette calculatrice seront appliquées à cette valeur.

La figure 6 montre le diagramme UML de la classe Calculatrice.

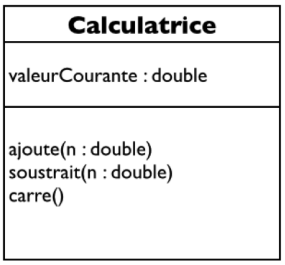

FIGURE 6 – Diagramme UML de la classe Calculatrice

Créez une nouvelle classe Calculatrice dans le package tp1. Déclarez une variable d'instance valeurCourante, initialisée à la valeur 0. Implémentez ensuite les trois méthodes indiquées dans le diagramme. Notez bien que ces méthodes ne renvoient rien ! Elles se contentent de modifier l'état de l'objet.

Pour tester votre classe, ajouter une méthode main qui :

1. crée un objet de type Calculatrice (la valeur courante sera donc 0),
2. ajoute 5 à la valeur courante,
3. soustrait 2 à la valeur courante,

tout en imprimant après chaque étape sur la console la valeur courante.

4.2 Classe Etudiant

Une Haute Ecole vous contacte pour informatiser la gestion de ses dossiers étudiants. Ces dossiers contiennent le nom, le prénom, et un numéro de matricule (entier positif) propre à chaque étudiant.

1. Dessinez le diagramme UML de cette classe sur base de ce cahier des charges

2. Implémentez la classe Etudiant
3. Ajoutez une méthode main à ce programme, qui instanciera la classe, donnera des valeurs aux différents attributs et les affichera dans la fenêtre console.
4. Modifiez la méthode main pour qu'elle instancie un objet Etudiant sur base des arguments de la ligne de commande.
5. Ecrivez et générez la javadoc de la classe Etudiant

Astuce : Pour convertir une chaîne de caractère en entier, vous pouvez utiliser la méthode Integer.parseInt(String s). Pensez à vérifier que les valeurs entrées sont valides !

4.3 Classe Date

La Haute Ecole souhaiterait enrichir le dossier de ces étudiants en y ajoutant sa date de naissance, nécessaire pour calculer son âge.

1. Dessinez le diagramme UML d'une classe Date. Chaque date sera caractérisée par 3 entiers représentant le jour, le mois et l'année.
2. Implémtez la classe Date
3. Ajoutez une méthode `main` qui instanciera la classe, donnera des valeurs aux différents attributs et qui les affichera dans la console. Pensez à vérifier que les valeurs des arguments sont valides.
4. Ecrivez et générez la javadoc de la classe Date.
5. Reprenez la classe Etudiant. Ajoutez à la classe une date de naissance de type Date. Modifiez la méthode main afin de donner une valeur à la date de naissance.

4.4 Classe Livre

Votre grand-père souhaite faire l'inventaire de sa bibliothèque, et consulter facilement cet inventaire de manière informatisée. Il n'a besoin que du titre, du nom de l'auteur et de son numéro ISBN afin de pouvoir y accéder rapidement via Amazon.

1. Dessinez le diagramme UML. Chaque livre est caractérisé par un titre, un auteur et un numéro ISBN (encodé sous forme de chaîne de caractères).
2. Réalisez la classe Livre sur base du diagramme UML.
3. Ajoutez une méthode main qui crée une instance de la classe Livre, donne des valeurs à chacun des attributs qui les affiche dans la fenêtre console.

4.5 Classe IP

Votre chef vous demande de réfléchir à la conception d'un outil qui permettra la gestion de l'adressage IP du réseau de l'entreprise dans laquelle vous travaillez.

1. Dessinez le diagramme d'une classe IPv4 qui contient quatre attributs, chacun correspondant à un octet de l'adresse IPv4.
2. Créez la classe correspondante.
3. Ajoutez une méthode main qui crée un objet IPv4 sur base de quatre entiers passés à la ligne de commande.