Imprimer tout le livre

TD1 : Pas à Pas, de l'analyse au développement

Site: LMS UCA
Cours: Conception et programmation objet avancees - TDFA315
Livre: TD1 : Pas à Pas, de l'analyse au développement

Imprimé par: Steve strazzeri
Date: mardi 8 septembre 2020, 14:23

Table des matières

1. Objectifs

Par le développement d'un mini-Forum tels que Twitter, Slack ou Discorde, nous rappelons les notions de base de la COO.

Rappel : le test est obligatoire. Il doit vous permettre de réviser et de vous améliorer. Il doit être réalisé en dehors des heures de cours.

Vous pouvez choisir de visualiser l'ensemble du TD en sélectionnant ***Imprimer tout le livre.***

2. Présentation de l'application

Un **Forum** (*e.g.* Slack, Discorde) permet à ses membres de poster des “messages” (texte, contenu multimedia, ...) et de lire les messages quand ils le veulent.
Il s'agit de construire les **bases “rudimentaires” d'un “Forum”** (on fera mieux la semaine prochaine).

2.1. Enoncé

- Un administrateur peut créer un forum.
- Un internaute peut s'inscrire à un forum, il en devient membre.
- Des membres postent des messages sur un forum.
- Des membres lisent des messages sur le Forum.
- Un administrateur peut connaitre la liste des membres d'un forum.
- Le membre qui a créé un message peut effacer **ce** message.
- Les messages postés il y a plus de 10 minutes sont effacés.

2.2. Un exemple de scénario

Attention, il s'agit d'un exemple concret mais c'est bien la notion de Forum que l'on vous demande dans un premier temps de mettre en oeuvre.

1. L'agence “oogle-stade” (*Administrateur*) crée un forum “OGCN”.
2. Mario s'inscrit sur le forum “OGCN”.
3. Mario (*Membre*) poste un message *WaitAndSee: “a quoi cela sert de courir?”* sur le forum “OGCN”.
4. Walter (*Membre*) demande s'il y a de nouvelles informations sur le forum et obtient le message *WaitAndSee*.
5. Il pose la même question un peu plus tard, et le système lui répond qu'il n'y a pas de nouveaux messages.
6. Walter demande à lire tous les messages.
7. Walter poste un message *Yes : “Tout à fait d'accord!”*.
8. Alban (*Membre*) demande s'il y a de nouveaux messages et obtient les messages *WaitAndSee and Yes.*
9. Youcef s'inscrit sur le forum puis poste un message *PFFF : “Vous rigolez?”*.
10. Walter demande à lire les nouveaux messages.
11. Walter demande à effacer le message réalisé par Youcef, il n'a pas le droit, cela ne fait rien.
12. Youcef efface son message
13. Les messages créés il y a plus de (temps à adapter pour vos tests en ms) sont effacés.

Les messages postés il y a plus de 10mn (adapté la durée pour les tests) sont détruits par “oogle-stade”.

3. Comprendre le contexte

5mn maximum

Quels sont les grands cas d'utilisation?
Vous pouvez les faire simplement sur Papier, sinon utilisez l'outil de l'an dernier : https://app.genmymodel.com/

4. De l'analyse à la conception du forum (simplifié)

Analyse : **15mn**

1. Définissez un **diagramme de classes de niveau Analyse**, en vous interessant essentiellement aux associations entre les classes.
2. Définissez, **par cas d'utilisation, un diagramme de séquence** élémentaire mettant en jeu les objets de votre système, à nouveau vous pouvez le faire sur papier, si cela vous permet d'aller plus vite à l'essentiel. **Complétez** votre diagramme de classes au fur et à mesure. Pour cela utilisez, évidemment des lignes de vie qui font référence à des classes et les messages qui vous permettent d'identifier les méthodes et mettre à jour vos diagrammes de classes.

- Avez-vous vérifié que le scénario initial est bien couvert par votre modélisation?
- Un forum délivre-t-il un ou plusieurs messages?
- Gérez vous bien les messages déjà lus ou non?

- Gérez-vous la destruction d'un message par son auteur?

-Avez vous prévu la destruction des messages postés il y a “longtemps”?

Conception : **15mn**

1. **Compléter votre modèle à classes** pour préparer l'implémentation :
   - Réfléchir au sens des relations
   - Vérifier les “couplages”
   - Cet exemple est très petit, en conséquence, il est possible qu'il n'y ait quasi rien à faire à cette étape.

Attention, ce tout petit exemple sera très largement étendu par la suite. Prenez-donc un soin particulier pour bien construire un modèle qui respecte les propriétés de faibles couplages… Les diagrammes de séquence peuvent vous aider.

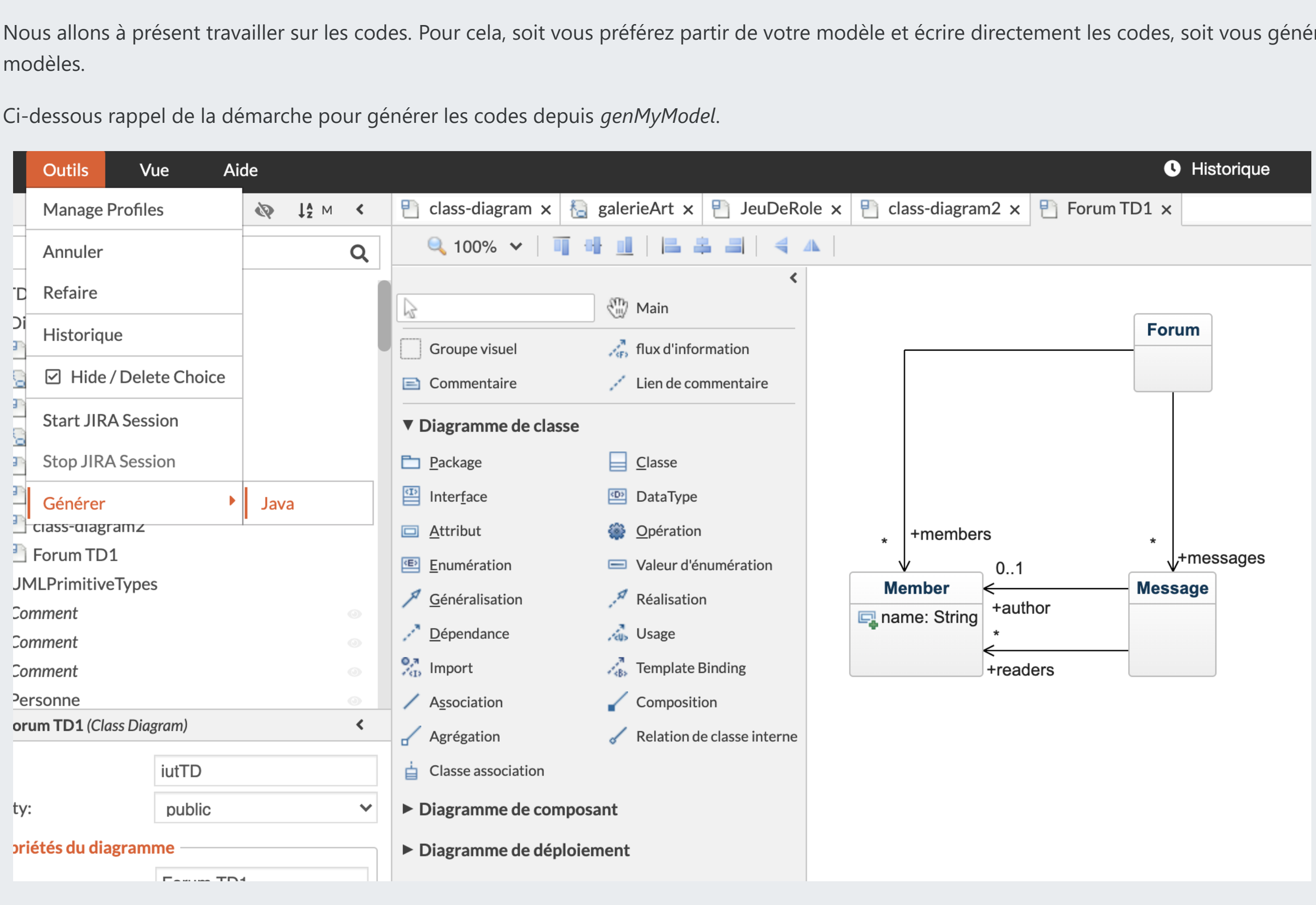
5. Mise en oeuvre

Dans cette partie nous travaillons les codes.

5.1. Génération des codes (10 mn)

Nous allons à présent travailler sur les codes. Pour cela, soit vous préférez partir de votre modèle et écrire directement les codes, soit vous générez la base du code à partir des modèles.

Ci-dessous rappel de la démarche pour générer les codes depuis *genMyModel*.



1. Générer les codes

5.2. Un projet java (2mn)

Créez un projet java sous Eclipse, en mettant bien en place un "folder source" **tests**.

Si vous le souhaitez, vous pouvez aussi le faire avec maven comme nous l'avons vu rapidement en M331.

5.3. Un projet sous tests

- Si vous avez généré les codes, déposer ces codes dans votre éditeur.
- Notre objectif est à présent de préparer les tests qui accompagneront notre développement.
- Pour cela, nous utiliserons l'environnement JUNIT.
- La structuration du projet en une partie principale et une partie test est exigée pour toute la suite de ce module.

Sous ECLIPSE, en projet Java (si Maven c'est à peine différent).

1. Dans le menu contextuel de, par exemple la classe *Message*, cliquez sur *New – Others - Java - JUnit Test Case*[2].Dans le panneau qui s'affiche
   - Sélectionnez le bouton radio *New JUnit 5 test*.
   - Changez le dossier Source folder pour celui de tests
   - Nommez la classe `MessageTest`.
   - Cochez les cases `setUp()` et `tearDown()`.
   - Enfin cliquez sur *Finish*.
   - Eclipse va remarquer que la bibliothèque de *JUnit* est absente du projet et vous propose d’ajouter automatiquement cette dernière au projet.
   - Dans le panneau qui apparaît, cliquez sur OK.
2. Eclipse a maintenant créé automatiquement le squelette de la classe de test. Il ne reste plus alors qu’à remplir cette dernière. **Voir un exemple de code ci-dessous, mais vous en avez aussi dans le TD réalisé en M331.**
3. Dans le menu contextuel, cliquez sur *Run As – JUnit test*.

Enfin, le premier rapport de tests s’affiche !

1. Visualiser la couverture de tests

Voici un exemple de code de test pour Junit 5

```
protected Message mToTest;
protected String contents;

@BeforeEach
public void setUp() throws Exception {
        contents = "mon message";
        mToTest = new Message(contents);
}

@Test
public void setContents() throws Exception {
        String nv = "newValue";
        mToTest.setContents("newValue");
        assertEquals(" message contents is as expected",nv,mToTest.getContents());
        assertFalse(" message contents has been modified", mToTest.getContents().equals(contents));
}
```

5.4. Mise en place d'un programme de tests de validation (10mn)

Voici un test global **à adapter à votre code.**

Vous pouvez compléter cet exemple par l'exemple donné plus haut.

Ce test ne peut pas encore tourner puisque quasi rien n'est implémenté, par contre vous pouvez vous aider de votre IDE pour qu'au moins il soit “syntaxiquement” juste et créer les bonnes méthodes si elles n'existent pas déjà, ou les appeler! Vous êtes libres de modifier les appels aux méthodes pour les adapter à votre propre modélisation.

A la fin de ce TD, évidemment, un test avec les mêmes objectifs devra fonctionner.

https://github.com/IUT-DEPT-INFO-UCA/M315-TD2

```
@Test
      void testMainScenario() throws InterruptedException {
            //L'agence “oogle-stade” (Administrateur) crée un forum “OGCN”.
            Forum ogcn = new Forum("OGCN");

            //Initialisation du Forum avec les membres
            Member mario = new Member("Mario");
            Member walter = new Member("Walter");
            Member alban = new Member("Alban");
            ogcn.addMember(mario);
            ogcn.addMember(walter);
            ogcn.addMember(alban);
            List<member> members = ogcn.getMembers();
            assertEquals(3, members.size());

            //Mario (Membre) poste un message WaitAndSee: “a quoi cela sert de courir?” sur le forum “OGCN”.
            Message wait = new Message("WaitAndSee", "a quoi cela sert de courir?", mario);
            Thread.sleep(6000);
            assertTrue(wait.getCreationDate().before(new Date()));
            ogcn.addMessage(wait);
            assertEquals(1, ogcn.getAllMessages().size());

            //Walter (Membre) demande s'il y a de nouvelles informations sur le forum et obtient le message WaitAndSee.
            //Il pose la même question un peu plus tard, et le système lui répond qu'il n'y a pas de nouveaux messages.
            //Walter demande à lire tous les messages.
            //Walter poste un message Yes : “Tout à fait d'accord!”.
            List<message> messages = ogcn.getNewMessages(walter);
            assertTrue(messages.contains(wait));
            messages = ogcn.getNewMessages(walter);
            assertEquals(0,messages.size());
            assertEquals(1, ogcn.getAllMessages().size());
            Message yes = new Message("Yes", "Tout à fait d'accord!", walter);
            ogcn.addMessage(yes);


            //Alban (Membre) demande s'il y a de nouveaux messages et obtient les messages WaitAndSee and Yes.
            messages = ogcn.getNewMessages(alban);
            assertTrue(messages.contains(wait));
            assertTrue(messages.contains(yes));

            //Youcef s'inscrit sur le forum puis poste un message PFFF : “Vous rigolez?”.
            Member youcef = new Member("Youcef");
            ogcn.addMember(youcef);
            Message pfff = new Message("PFFF", "Vous rigolez?", youcef);
            ogcn.addMessage(pfff);
            messages = ogcn.getNewMessages(youcef);
            assertEquals(3, messages.size());
            messages = ogcn.getNewMessages(youcef);
            assertEquals(0, messages.size());

            //Walter demande à lire les nouveaux messages.
            messages = ogcn.getNewMessages(walter);
            assertEquals(2, messages.size());

            //Walter demande à effacer le message réalisé par Youcef, il n'a pas le droit, cela ne fait rien.
            messages = ogcn.getAllMessages();
            int numberOfMessages = messages.size();

            boolean removed = ogcn.remove(pfff, walter);
            assertFalse(removed);
            messages = ogcn.getAllMessages();
            assertEquals(numberOfMessages,messages.size());

            //Youcef efface son message
            removed = ogcn.remove(pfff, youcef);
            assertTrue(removed);
            messages = ogcn.getAllMessages();
            assertEquals(numberOfMessages-1,messages.size());

            assertFalse(pfff.isOutOfDate(2));
            Thread.sleep(2001);
            assertTrue(pfff.isOutOfDate(2));
            //Les messages postés il y a plus de 10mn (adapté la durée pour les tests) sont détruits par “oogle-stade”.
      }
```

5.5. Développement (20mn)

Terminez la mise en oeuvre du forum en faisant en sorte que les tests passent au fur et à mesure.

Pour gérer le temps

```
Date d = new Date();
long ms = d.getTime();
ms = ms - secondsElapsed*1000;
d.setTime(ms);
return creationDate.before(d);
```

https://github.com/IUT-DEPT-INFO-UCA/M315-TD2

6. Vous savez ...

- Créer un projet sous un IDE avancé, et le structurer correctement.
- (Rappel) Faire correspondre le code java et un modèle de classes en UML.
- Mettre en place des tests unitaires.
- Utiliser un IDE pour améliorer mon développement en utilisant les outils d'aide au développement.

Pour les avancés

Des exemples de code Producteur/Consommateur