TD1 : Alarme connectée

Site: LMS UCA 2021/2022
Cours: TDFM315 - Conception et programmation objet avancees
Livre: TD1 : Alarme connectée

Imprimé par: Donati Leo
Date: dimanche 5 septembre 2021, 16:49

Table des matières

1. Objectifs

Pas à pas : de l'analyse au développement (premiers pas)

On revoit les notions de base de la COO sur un projet concret : **concevoir le programme de gestion d'un système d'alarme connectée pour particuliers.**

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*** dans le menu d'actions (en haut à droite).

2. Présentation du projet

Une entreprise veut mettre sur le marché un système complet d'alarme connectée avec une box connectée à internet à laquelle peuvent être associés des capteurs, des caméras, des sirènes, etc...

On vous demande de concevoir le programme d'administration qui permet au client de gérer des capteurs, vérifier l'état du système, visualiser les images des caméras de surveillance, activer la sirène, etc...

Dans ce premier TP on veut commencer à mettre en place les principaux éléments constitutifs du système.

2.1. Enoncé

Le client gestionnaire du système doit pouvoir :

- lister l'ensemble des dispositifs connectés au système
- connaître l'état de chargement de la pile de chaque dispositif
- activer/désactiver individuellement chaque dispositif
- ajouter un dispositif au système au système en spécifiant son emplacement dans la maison et/ou le jardin
- associer un ou plusieurs numéros de portable pour recevoir les notifications du système d'alarme
- connaître l'historique des déclenchements des dispositifs (logs) et les effacer

Les habitants de la maison doivent pouvoir
- activer/désactiver l'alarme
- éteindre les sirènes connectées au système
- recevoir les images associées aux caméras connectées au système

Pour des raisons de service après-vente et de hot-line des administrateurs au sein de l'entreprise doivent pouvoir vérifier à distance l'état du système et des dispositifs connectés pour détecter des pannes et éventuellement faire des mises à jour du système.

2.2. Exemple de scenario

1. Le gestionnaire Arthur se connecte au système en s'identifiant avec un mot de passe
2. Arthur demande la liste des dispositifs connectés au système
3. Dans la liste Arthur sélectionne la "caméra du salon" et vérifie la charge de la pile
4. Arthur allume la "caméra du salon" et visionne son flux vidéo"
5. Arthur éteint la "camera du salon"
6. Arthur sélectionne le détecteur de mouvement du salon et l'active
7. Arthur demande la liste de tous les logs du détecteur de mouvement
8. Arthur ajoute au système un détecteur d'ouverture relié à la porte-fenêtre du salon en saisissant son identifiant unique (présent sur la boite)
9. Arthur vérifie que ce nouveau dispositif est bien dans la liste et l'active.
10. Arthur règle l'activation du système d'alarme à 10h00, se déconnecte et sort de sa maison.

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

Analyse : 15mn

Définissez un **diagramme de classes de niveau Analyse**, en vous intéressant essentiellement aux associations entre les classes.

*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?
- Votre système est-il ouvert à tous les types de dispositifs possibles?
- Peut-on effacer le fichier de log ?

- Est-il possible de connaître les dispositifs présents dans une pièce donnée de la maison?

Conception : 15mn

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.

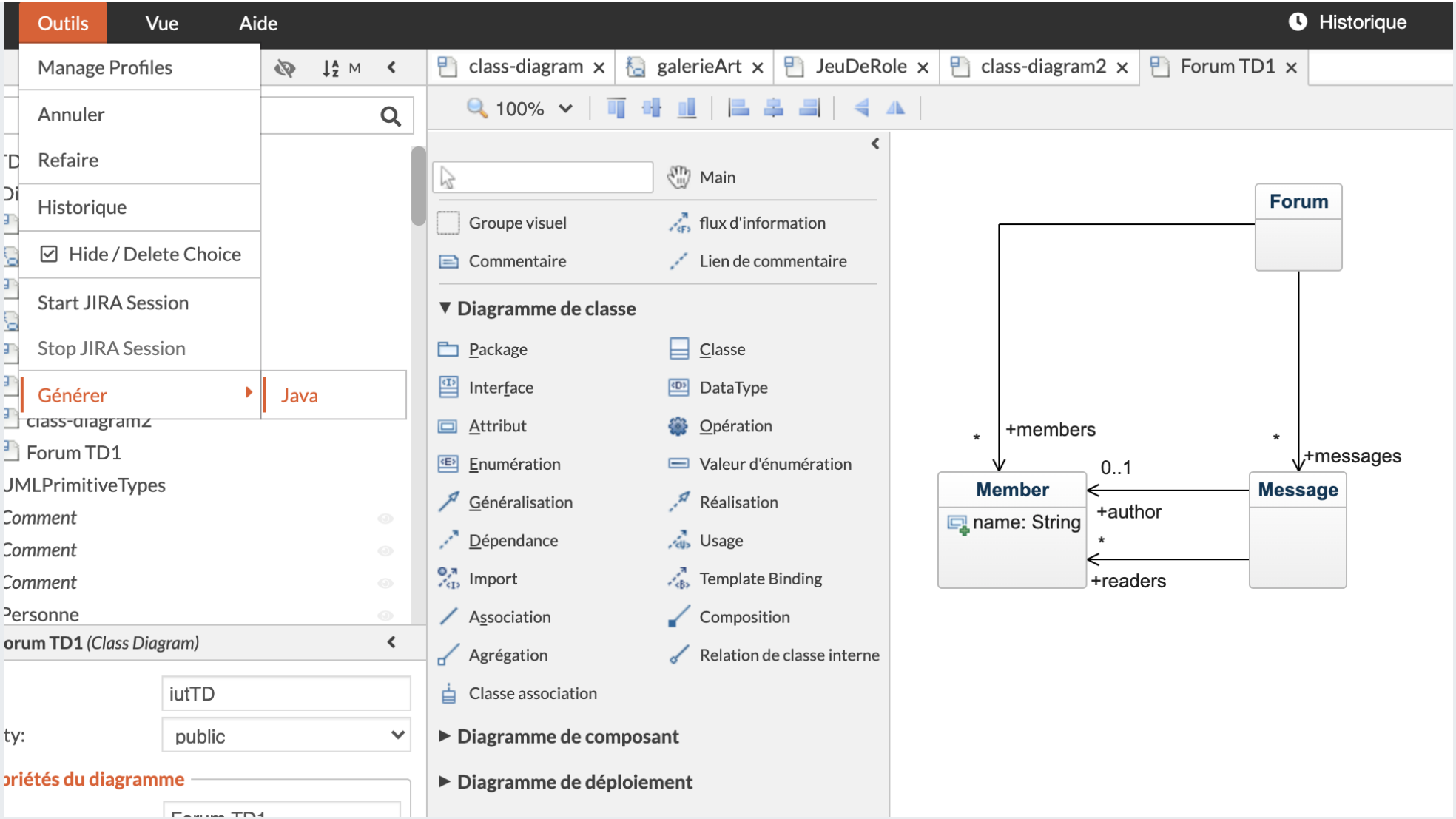
5. Mise en oeuvre

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****.*

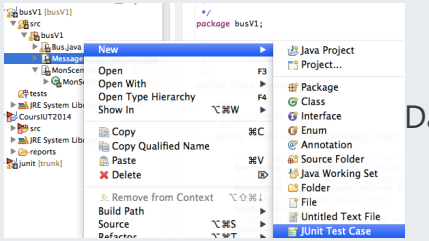
5.3. Un projet sous test

- 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.

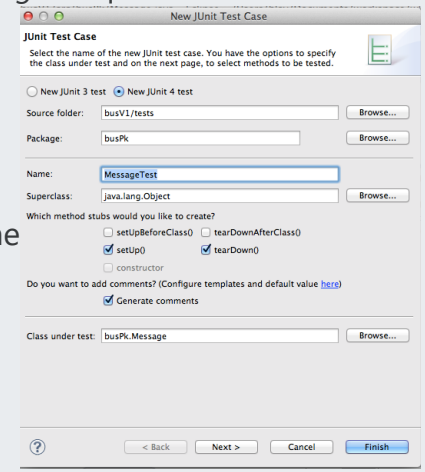
Si vous êtes à l'aise avec la gestion d'un projet sous Eclipse et JUnit, vous n'avez pas besoin de cette partie.

Sous ECLIPSE, en projet Java

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

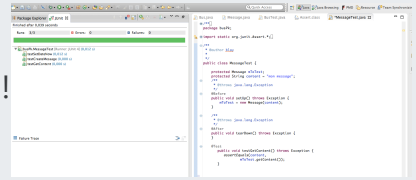


   - Sélectionnez le bouton radio *New JUnit 5 test*.
   - Changez le dossier Source folder pour celui de tests
   - Nommez la classe `MessageTest`.

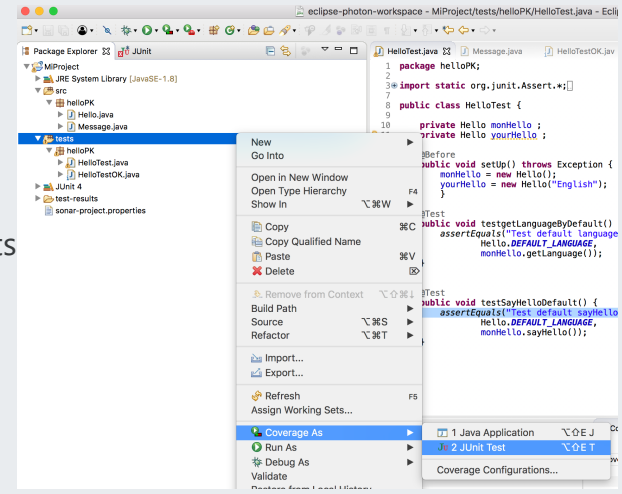
- Cochez les cases `setUp()` et `tearDown()`.
- Enfin cliquez sur *Finish*.
- Eclipse va remarquer que la bibliothèque de *JUnit* est absente du projet et vous propose d'ajouter automatiquement cette dernière au projet.
- Dans le panneau qui apparaît, cliquez sur OK.

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



Vous trouverez des codes pour vous aider dans le dépôt github : https://github.com/IUT-DEPT-INFO-UCA/M315-TD2

5.4. Développement (20mn)

Terminez la mise en œuvre du projet en faisant en sorte que chaque classe ait ses tests unitaires.

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

Mettre en place un test de validation en reprenant l'exemple de scenario d'utilisation donné plus haut, avec des modifications éventuellement pour l'adapter à votre conception.

Le principe étant que chaque étape du scenario puisse être validée par l'appel des tests unitaires développés précédemment.

Afin de vous aider, nous vous proposons ci-dessous un exemple de test de validation correspondant au sujet étudié l'année dernière (forum de chat) :

```
@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("a quoi cela sert de courir?", mario);
		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( "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( "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”.
        }
```

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 votre développement en utilisant les outils d'aide au développement.