TD2 : qui est Responsable?

Site: LMS UCA
Cours: Conception et programmation objet avancees - TDFA315
Livre: TD2 : qui est Responsable?

Imprimé par: Steve strazzeri
Date: mardi 22 septembre 2020, 14:42

Description

Objectifs : Travailler, sur de petits exemples, différents points d'un développement pragmatique.

Table des matières

1. Enoncé

Vous développez à présent une application du type Slack ou Discord, dont vous avez ci-après les directives.

- [ ] Un membre demande à créer un forum en précisant le nom du forum. Il est alors administrateur du forum. Si un forum avec ce nom existe déjà, il ne se passe rien.
- [ ] Un forum est maintenant composé de *canaux*.
- [ ] Un membre peut demander s'il existe un forum de nom donné et s'inscrire au forum.
- [ ] Un membre peut lister les noms des forums existants qu'il a créé : cela veut dire obtenir la liste des noms de forum, pas les afficher!
- [ ] Un membre demande à créer un canal dans un forum en précisant le nom du canal.
  - Si le canal existe déjà, il ne se passe rien.
- [ ] Par défaut un canal ne peut pas contenir plus de "MAX_MESSAGES", mais il est possible de modifier le nombre maximum de messages autorisés. Proposez une gestion du dépassement du nombre de messages (exception, ne rien faire, ...)
- [ ] Un membre demande à créer un canal de "brèves" (Messages) associé à un forum en précisant le nom du canal et la taille maximum des brèves, par exemple 140 caractères.
  - Une brève est un message dont le nombre de caractères autorisés est limité par le canal. Pensez aux messages Twitter par exemple.
- [ ] Un membre poste un message dans un forum et éventuellement ses un canal spécifique du forum, en précisant leurs noms (forum et canal) et le contenu du message. En l'absence de canal, le message est posté dans le *canal par défaut*.
- [ ] Un membre peut demander à lister les messages qui se trouvent dans un canal sur un forum et/ou les nouveaux messages (ceux qu'il n'a pas encore lus).
- [ ] Un membre peut demander à lister les nouveaux messages ou tous les messages qui se trouvent sur le forum, dans ce cas tous les nouveaux ou les messages dans tous les canaux lui sont retournés.

Pour vous aider vous pouvez, dans un premier temps, ne pas tenir compte du canal par défaut et n'en tenir compte qu'à la fin du TD en identifiant bien les impacts sur le modèle et sur le code, en particulier avec les responsabilités qui en résultent.

2. Responsabilités : Analyse et développement de la partie "Métier"

1. Les questions ci-après sont là pour vous guider dans le développement. Il n'est pas demandé d'y répondre par écrit.

Utilisez **SonarLint** pour vous aider à améliorer vos codes
A la fin de cette page, vous avez des "tuyaux" pour manipuler des listes et des HashMap

A Faire

Modéliser, implémenter et tester cette application mais pour cela aidez vous des éléments donnés ci-dessous.

Questions pour prendre du recul avant le développement

Dans ce qui suit *responsable* est utilisé au sens de GRASP.

1. Quels modèles utilisez-vous pour analyser le problème ?
2. Dans vos diagrammes de séquences, faites bien apparaître les interfaces graphiques et les contrôleurs mais dans les diagrammes de classe concentrez vous uniquement sur la partie métier dans un premier temps. *Cela ne signifie pas que vous devez modéliser tous les diagrammes de séquences.*
3. Qui est responsable de retrouver à partir de son nom un forum? de retouver un canal de messages? (Pattern expert)
4. Qui est responsable de créer un forum? un canal de message? pourquoi? (Pattern créateur)
5. Qui est responsable de créer un message ? pourquoi? (Pattern créateur)
6. Avez-vous des relations bi-directionnelles? Si oui, avez-vous décidé de qui est le "maître" et de qui est l'"esclave"?
7. Avez-vous bien prévu les tests unitaires? Que se passe-t-il si vous demandez
   1. à lire les messages sur un canal qui n'existe pas?
   2. à créer un forum d'un nom déjà utilisé?
   3. ...

Un scenario

A partir de maintenant, on s'intéresse au scénario suivant, sans tenir compte de l'interface graphique. Est-ce que vous pouvez l'implémenter, revenez sur votre code si besoin.

1. *créer un forum* "NiceInformation" par *Jean-Pierre*
2. *créer une canal* "@soirees" sur le forum "NiceInformation"
3. *créer un canal de Breves* "@dept-info-iut-Nice" qui autorise uniquement des messages de 140 caractères.
4. *Modou poste un message* "Siesta" dans le canal "@soirees" du forum "NiceInformation"
5. *Yassine poste un message* "ShadowBar" dans le canal "@soirees" du forum "NiceInformation"

1. *poster un message* "ShadowBar" dans le canal "@soirees" du forum pareil pour le forum "AntibesInformation", ça passe?
2. *lire les messages* qui se trouvent *dans le canal* "@soirees", i.e. récupérer les messages.
3. *effacer les messages du canal* "@soirees" du forum "NiceInformation"
4. *poster un message* "SoireeIUT" sur le forum "NiceInformation" (donc sans préciser le canal)
5. *poster un message* "WEI ... apres les cours de ..." dans le canal "@dept-info-iut-Nice"
6. Yoann plus bavard *poste un message de plus de 140 caractères* " N'oubliez pas d'emmener ..... et de choisir votre filleul ....." dans le canal "@dept-info-iut-Nice"... que se passe-t-il ?
7. Attendre 2s (Thread.sleep(2000));
8. Poster un message "SoireeIntegrationIUT Choisissez vos parrain...." sur le forum "NiceInformation"
9. Attendre 2s (Thread.sleep(2000));
10. <Optionnel> Effacer tous les messages postés depuis plus de 4s sur le forum "NiceInformation"

1. Fixer MAX_MESSAGES à 1... comment se comporte votre programme?
2. Complétez les tests pour prendre en compte les "brèves".
3. Comment avez-vous géré le canal par défaut? (Avez-vous pensé à définir des constantes?)
4. Pouvez-vous décrire chacune de vos classes avec une seule phrase? Faîtes-le dans les commentaires de vos classes.

Aie aie votre "Product Owner" vous annonce que sur des forums de type "Community" il veut interdire d'avoir plus de 2 canaux, sur un forum "Premium" on ne peut pas en avoir plus de 5 et sur les forum "PRO" le nombre de canaux n'est pas limité. En plus, **il veut savoir quel membre a créé un canal**.Comment réagissez vous à ce changement? Modifiez vos codes et modèles pour tenir compte de ces nouvelles fonctionnalités.

3. Architecture : Interface graphique et contrôleur

Exigences

L'objectif est ici de décomposer l'application pour que :

1. le **programme principal** consiste à créer un contrôleur et à le démarrer.
2. **l'interface** n'interagisse jamais directement avec la logique applicative (vos classes qui correspondent au Forum etc ici).
   - Elle ne connait pas non plus le contrôleur dans cet exercice
   - Voici un exemple de code pour l'interface. Pour se concentrer sur l'architecture, l'interface est une classe Java qui utilise Scanner pour lire au clavier et System.out.print... pour l'affichage.
3. Le **contrôleur** joue le rôle de chef d'orchestre entre le métier et l'interface. Il ne peut pas utiliser System.out...; tout affichage passe par l'interface. Il n'a pas le droit de modifier la partie métier sans passer par des objets métiers. Il n'est pas autorisé à stocker la liste des bus etc. Pensez à utiliser le principe de délégation. Voici des extraits de code pour vous aider ci-après.

A Faire

1. Développez le code correspondant et vérifier qu'il fonctionne

Questions de recul

1. Quel type de contrôleur avez-vous défini en respectant les directives données?

1. - Vous devez à présent pouvoir jouer le scénario précédent en utilisant votre interface graphique. - Visualisez votre code correspondant à la création d'un forum (et uniquement cela) sous la forme d'un diagramme de séquences.\\ - Pouvez-vous décrire chacune de vos classes avec une seule phrase? Faîtes-le dans les commentaires de vos classes.

4. Architecture : Persistance simplifiée

On désire sauvegarder l'ensemble des forums à chaque fin d'exécution du programme principal et recharger l'état des forums à chaque lancement du programme principal. Vous pouvez utiliser (L2018_2019:s3:concprogobjet:memoire.java[le code suivant]). Pour l'utiliser il suffit que vos classes (gestionnaire, Message, Forum, ...) **implements Serializable**. **Exemple de sauvegarde :**

```
Memoire.save(registre, NomFichier);
```

Exemple de lecture :

```
      Object o   = Memoire.read(NomFichier);
      if (o instanceof Gestionnaire){
            registre = (Gestionnaire) o;}
}
```

5. Evolutions

1. Dans certains canaux dès qu'un message est lu, il est effacé. Modifier votre code pour permettre la création de tels canaux. Attention, les autres canaux sont toujours utilisables. Utilisez bien les tests unitaires, pour vérifier que vous ne "cassez" pas vos codes. Comment votre modèle est-il modifié?
2. Pour certains forums, lorsque l'on poste un message en donnant comme nom de canal "*", le message est posté dans tous les canaux. Modifier votre code pour tenir compte de cette nouvelle fonctionnalité
3. Analyser les dépendances entre vos classes.

6. Codes et tuyaux pour vous aider

Vous avez besoin dans ce TD de manipuler des collections et des dates, voici quelques tuyaux que vous pourriez retrouver sur le Web et qui sont extraits des codes que nous avons mis en oeuvre pour ce TD. Vous pouvez évidemment avoir d'autres solutions :

Pour manipuler des HashMap

```
    HashMap canaux = new HashMap<>();
    //Ajout d'un object dans la hashmap
    canaux.put(c.getNom(), c);

    ArrayList messageList = new ArrayList<>();
    // obtenir les objets qui se trouvent dans la hashmap (.values)  et les parcourir (for)
   for (Canal c : canaux.values())
        messageList.addAll(c.getMessages());

   // retrouver un objet de nom donné
   Canal c = canaux.get(nom);

   //Obtenir toutes les clefs (ici le nom des canaux)
   Set  noms = canaux.keySet();
```

Pour manipuler des collections

Pour effacer un élement dans une liste si une condition est vérifiée :

```
      public void detruireMessagePerime(int dureeEnSecondes) {
            messageList.removeIf(message -> message.estPerime(dureeEnSecondes));
      }
```

Pour gérer les dates

Une manière "facile" pour savoir si un message est périmé de plus de x secondes (utilisation de "java.util.Date") (Nous l'avons déjà vu au TD précédent).

```
public boolean estPerime(int second){
      Date d = new Date();
      long ms = d.getTime();
      ms = ms - second*1000;
      d.setTime(ms);
      return this.dateEmission.before(d);
   }
```