Test JAVA Avancé

Stage 17-2 SIDAIR/RSI

--

Le 04 octobre 2017

--

NB: La lisibilité du code (aération, indentation) et le respect des normes de développement (conventions de nommage, Javadoc…) prennent une part non négligeable dans le barème de la note

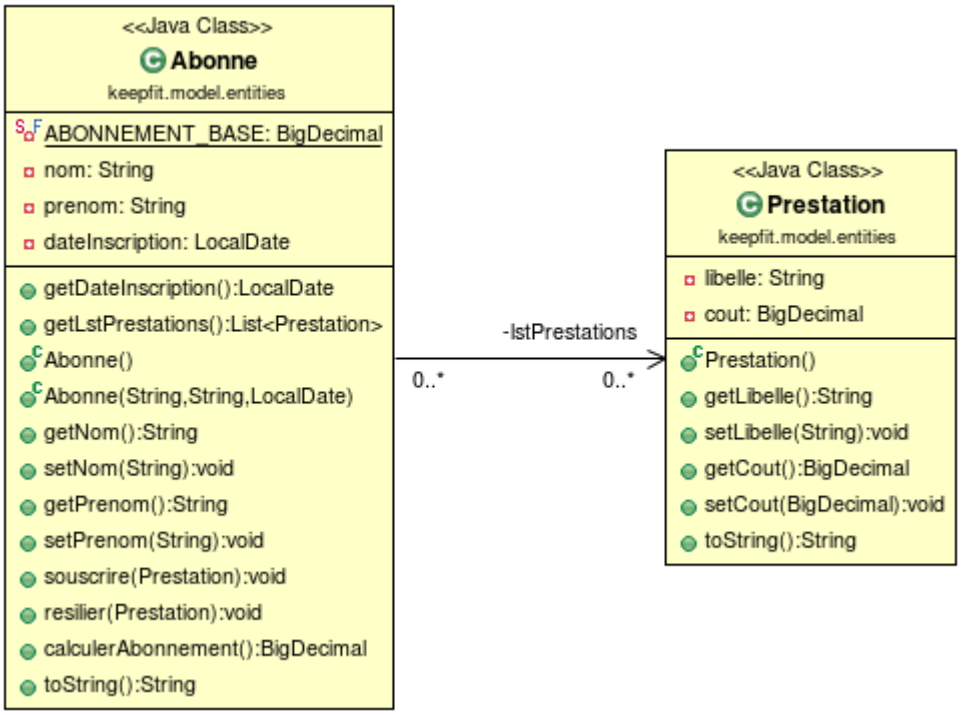
Projet « keepfit »

Contexte

Un ancien collègue formateur à l'ETRS, qui a pris sa retraite il y a quelques années, s'est reconverti dans les clubs de remise en forme. Sa chaîne de clubs : « keep fit » connaît un grand succès dans l'ouest de la France. Et bien que « keep fit » soit devenu un acteur majeur dans son secteur, la gestion du fichier client se fait toujours « à l'ancienne » avec des fiches papier.

Notre ancien collègue a les dents longues et souhaite optimiser la gestion de ses clients en s'appuyant sur un système d'information. Il nous sollicite afin de lui développer un prototype d'application type « client lourd ».

Voici le diagramme de classes métier résultant de l'analyse menée :



Les diagrammes sont joints à l'énoncé (cf. dclam_Keepfit.png et metier_Keepfit.png)

L'entité Abonne :

- ✓ Possède une constante (commune à toutes les instances d'Abonne) ABONNEMENT_BASE de type BigDecimal : c'est le montant de l'abonnement annuel « de base » qui comprend uniquement l'accès à la salle de cardio-training (ergomètre, vélo d'appartement, tapis roulant…) et au bar (ouf), ce montant est de 240 euros annuel.

- ✓ Caractérisée par :
  - o Un nom et un prénom : ils seront composés uniquement de lettres minuscules/majuscules (cf. Bean Validation)
  - o Une date d'inscription : si elle n'est pas spécifiée, la date du jour sera la date d'inscription par défaut.

- ✓ méthode souscrire (Prestation p) : méthode qui ajoute une prestation à la liste des prestations de l'abonné : elle lève une DejaAbonneException si jamais l'abonne souscrit déjà à la prestation passée en paramètre Exemple de prestations : musculation, aquabiking, sauna, fitness…

- ✓ méthode resilier (Prestation p) : méthode qui enlève une prestation à la liste des prestations de l'abonné

- ✓ méthode calculerAbonnement( ) : méthode qui calcule le montant total de l'abonnement : abonnement de base + montant de chacune des prestations souscrites par l'abonné

L'entité Prestation :

- ✓ Caractérisée par :
  - o Un nom
  - o Un cout de type BigDecimal

Priorisation des fonctionnalités métiers attendues :

1-Lister des abonnés / Détailler les abonnés
2-Créer des abonnés
3-Gérer les souscriptions/résiliations de prestations de chacun des abonnés (ajout/suppression de prestations)
4-Initialiser des prestations en base de données depuis le fichier fourni, au lancement de l'applicatif, seulement s'il n'y en a pas déjà en base
5-Supprimer un abonné

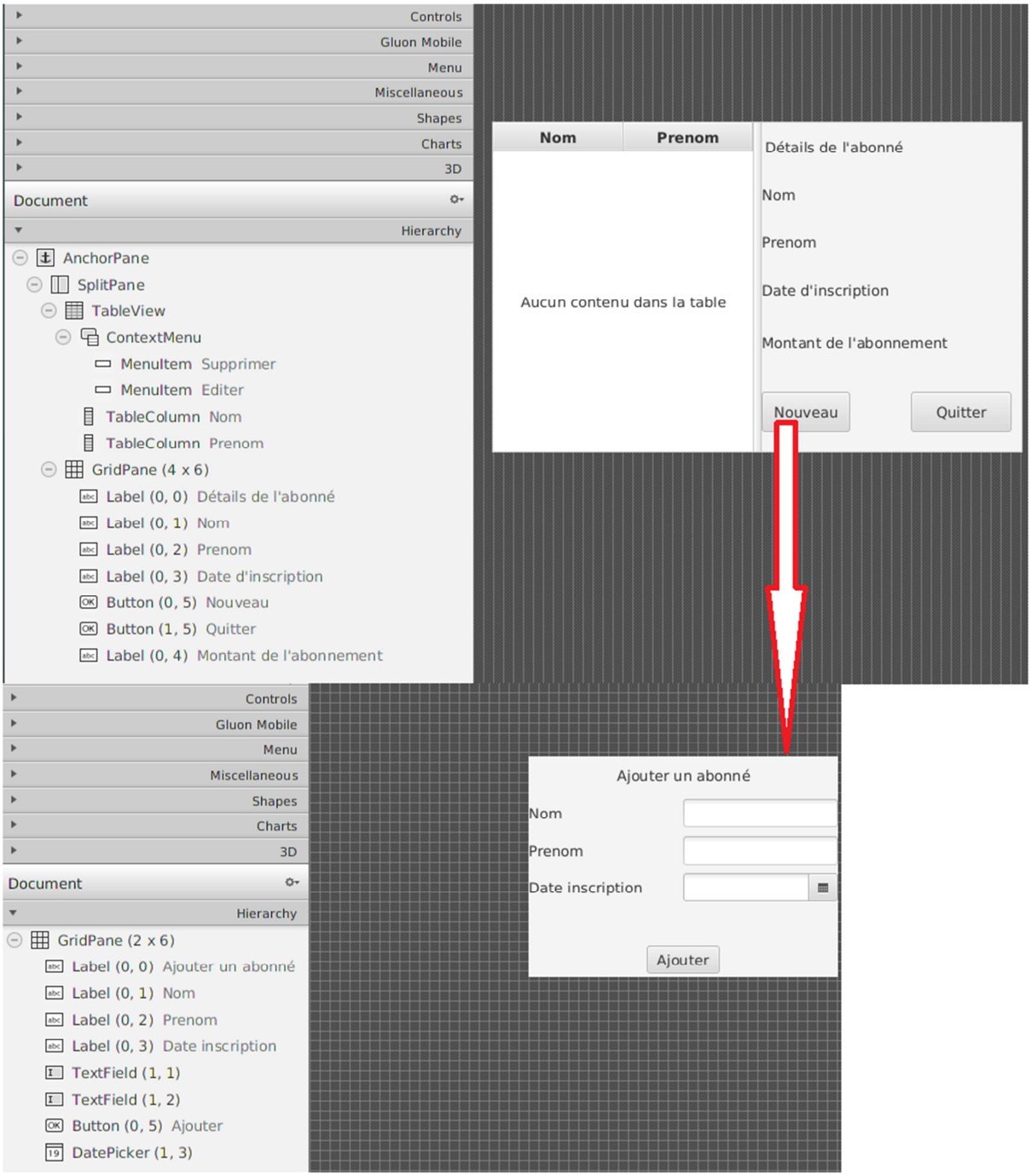
Contraintes techniques :

- ✓ Le projet devra impérativement être nommé « Keepfit_VOTRE_NOM ».

- ✓ La persistance imposée est une persistance « base de données » via l'utilisation d'un outil de mapping « Objet-Relationnel » respectant la spécification JPA (JAVA Persistence API). Le département technique a choisi :
  - MySql comme SGBD
  - « EclispseLink » comme implémentation de JPA.
  - L'administrateur base de données vous impose de nommer « KEEPFIT » la base de donnée et le nom de votre connexion « dataSource Eclipse » doit être : keepfitCnx.
  - Le chef de projet vous impose de nommer votre unité de persistance «keepfit »
  - Toutes les requêtes JPQL qui vous semblent utiles seront exploitées sous forme de NamedQueries et leur écriture centralisée dans une classe dédiée.
  - Un identifiant « technique » sera obligatoirement utilisé comme clé primaire (utiliser l'AbstractEntity et l'UUIDConverter vus en cours)

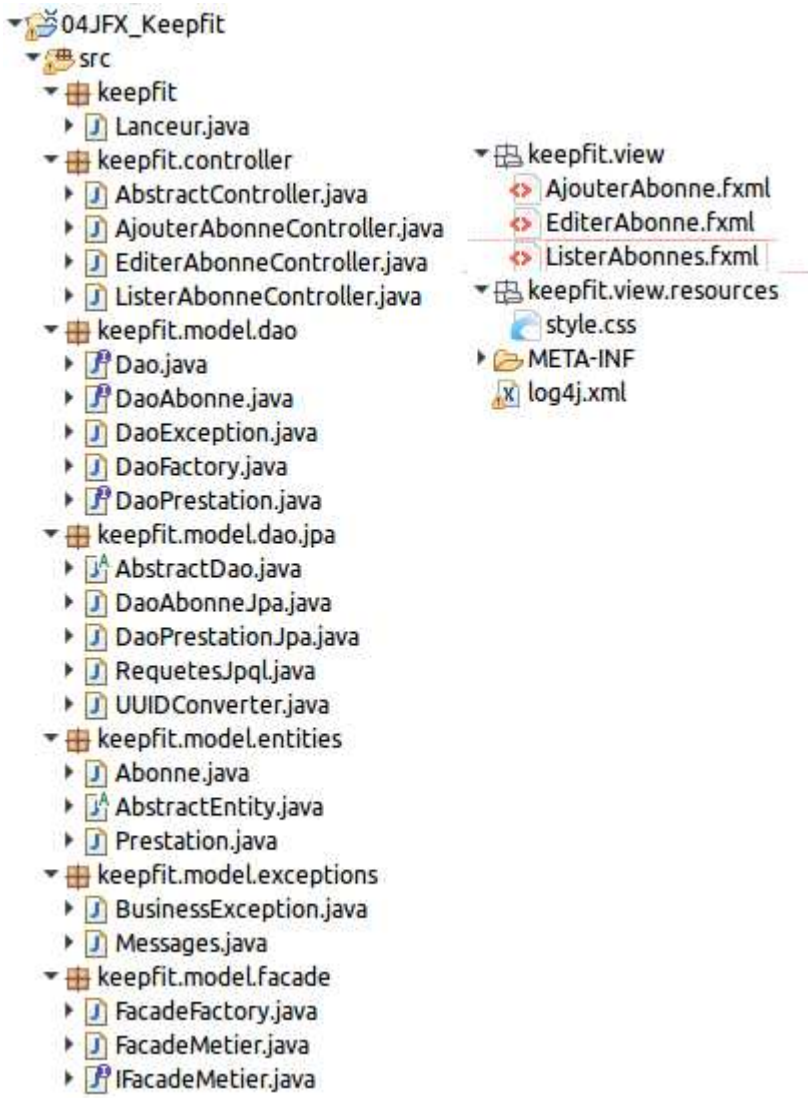
- La structure de la base de données est jointe à cet énoncé (cf. base de donees.png)

✓ L'application sera développée en JAVA FX par approche déclarative afin de garantir une séparation claire entre la vue et le reste de l'application, voici la scène attendue pour l'écran « principal » :



Le déroulé complet des fonctionnalités et de l'enchainement des écrans est disponible sous forme de vidéo jointe à cet énoncé (cf. video_keepfit.mp4).

- ✓ Les classes « contrôleurs » seront nommées exactement comme leur vue correspondante mais suffixée par « Controller » : cf. exemple de structure de projet :



  Le package *keepfit.model.dao*, la classe *Lanceur* et l'interface *IFacadeMetier* sont fournis et sont à copier dans votre projet.

- ✓ L'application de style CSS est indispensable : l'ergonomie et le rendu graphique sont en effet très importants aux yeux du potentiel acquéreur du logiciel qui impose sa charte graphique : la feuille de style à appliquer (cf. style.css) est jointe à cet énoncé.

- ✓ L'utilisation de lombok est obligatoire.

- ✓ Un système de journalisation est attendu: toutes les fonctionnalités métier seront loggées niveau INFO dans un fichier HTML et les exceptions métier de niveau WARN dans cette même page HTML (à la racine de votre projet) :

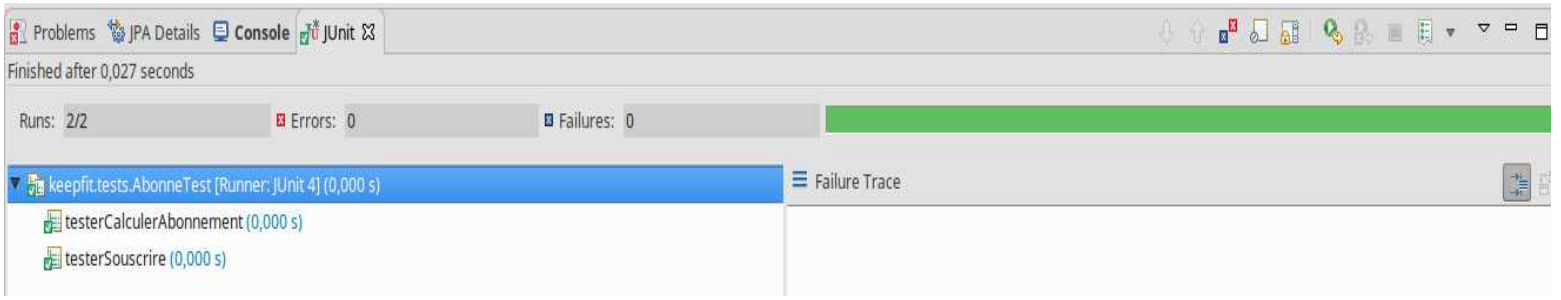
Log session start time Wed Apr 12 15:07:17 CEST 2017

Time	Thread	Level	Category	File:Line	Message
0	main	INFO	org.hibernate.validator.internal.util.Version	Version.java:27	HV000001: Hibernate Validator 5.1.1.Final
939	main	INFO	keepfit.model.facade.FacadeMetier	FacadeMetier.java:37	Initialisation de prestations en base...
74295	main	WARN	keepfit.model.facade.FacadeMetier	FacadeMetier.java:122	Impossible d'ajouter la prestation BODYPUMP pour Magniez Nicolas

Time	Thread	Level	Category	File:Line	Message
11947	main	INFO	keepfit.model.facade.FacadeMetier	FacadeMetier.java:118	Ajout de la prestation AQUAGYM pour Boucher JJ

Le fichier de configuration log4j.xml est joint à l'énoncé.

- ✓ Des tests sont indispensables : créer une classe JUnit 4 « AbonneTest » qui :
  - teste le bon fonctionnement de la méthode calculerAbonnement de la classe Abonne
  - teste la bonne levée de l'exception métier DejaAbonneException de la méthode souscrire de la classe Abonne



Vous rendrez votre projet pour 11h50 au plus tard dans « EcritPour nicolas.magniez » sous la forme d'un zip nommé VOTRE_NOM.zip.