Test JAVA Avancé
Stage 17-3a DEVSI/SIDAIR/RSI
--
Le 20 avril 2017
--

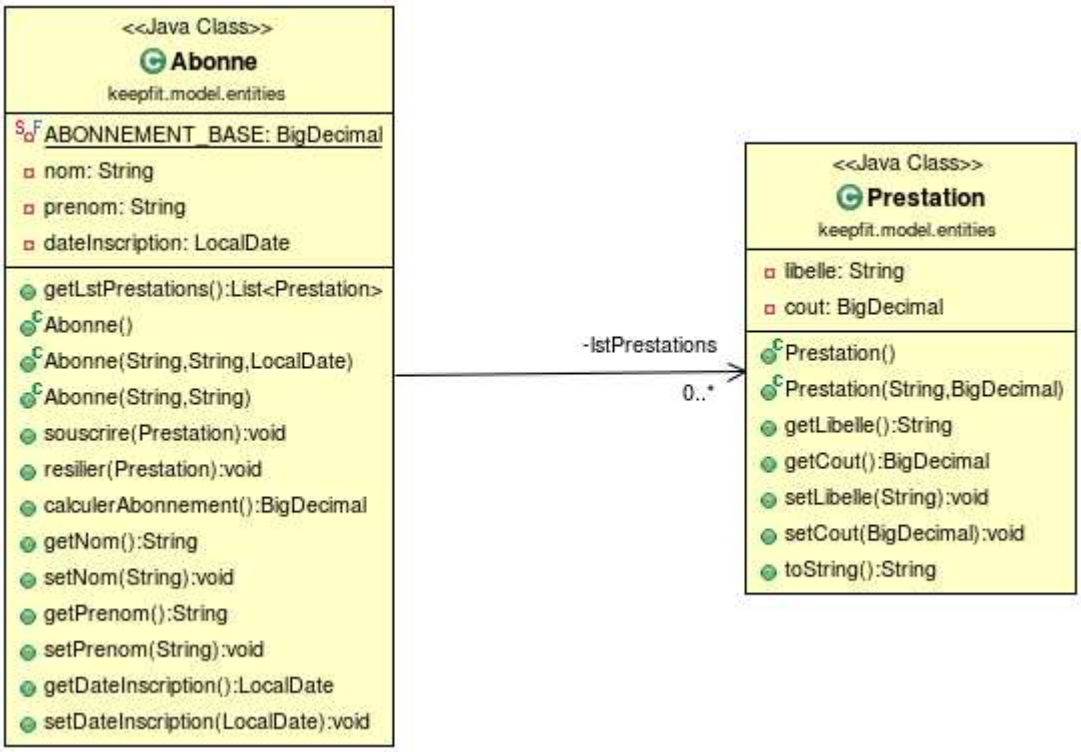
NB: La lisibilité du code (aération, indentation) et le respect des normes de développement (conventions de nommage, Javadoc…) prennent une part non négligeable dans le barème de la note

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 :



Le diagramme complet est joint à l'énoncé.

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.

- ✓ Caractérisé 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 :

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

Priorisation des fonctionnalités métiers attendues :

1-Créer des abonnés
2-Gérer les souscriptions de prestations de chacun des abonnés (ajout de prestations)
3-Lister des abonnés
4-Initialiser des prestations en base de données depuis le fichier CSV fourni, au lancement de l'applicatif, s'il n'y en a pas déjà en base
5-Visualiser par prestation le total des recettes qu'elles génèrent

```java
public interface IFacadeMetier {

    /**
     * Initialisation de prestations en base de données depuis un fichier csv fourni
     */
    void init() throws InitialisationImpossibleException;

    /**
     * Fonctionnalité "Ajouter un abonné"
     * @param a : nouvel abonné
     * @throws CreationAbonneImpossible
     */
    void creerUnAbonne(Abonne a) throws CreationAbonneImpossible;

    /**
     * Fonctionnalité "Lister tous les abonnés"
     * @return List<Abonne> : les abonnés de la bdd
     * @throws ListingAbonnesImpossible
     */
    List<Abonne> listerLesAbonnes() throws ListingAbonnesImpossible;

    /**
     * sous-fonctionnalité "Lister les prestations"
     * @return List<Prestation> : les prestations de la bdd
     * @throws ListingMarquesImpossible
     */
    List<Prestation> listerLesPrestations() throws ListingMarquesImpossible;

    /**
     * Fonctionnalité "Ajouter une prestation a un abonné"
     *
     * @param p : la prestation choisie
     * @param a : l'abonné choisi
     * @throws MajAbonneImpossible
     * @throws DejaAbonneException
     */
    void ajouterPrestationAbonne(Prestation p, Abonne a) throws MajAbonneImpossible, DejaAbonneException;

    /**
     * Fonctionnalité qui donne par Prestation le montant total des recettes sous la forme d'un dictionnaire:
     * Clé      : Valeur
     * FITNESS : 4500€
     * AGUAGYM : 6200€
     * UV       : 3560€
     * etc...
     *
     * @return Map<Prestation, BigDecimal>
     */
    Map<Prestation, BigDecimal> detaillerRecettePrestation();

}
```

Contraintes techniques :

- ✓ Votre application doit respecter le paradigme MVP afin d'offrir de la souplesse dans la maintenabilité du code et de faciliter les évolutions futures (de l'interface homme/machine et de la persistance notamment)

- ✓ Votre projet sera nommé Keepfit_VOTRE_NOM

- ✓ La persistance des entités métier sera réalisée au moyen de l'implémentation JPA "EclipseLink" : un identifiant « technique » sera obligatoirement utilisé comme clé primaire (cf. tables.png fourni pour la structure des tables)

- ✓ L'utilisation de lombok est obligatoire

- ✓ Le package « view » est joint à cet énoncé : une vue graphique « Swing » y est fournie, à vous de coder la vue « Console »

- ✓ Les packages « dao » et « facade », la classe Messages contenant les messages d'exceptions ainsi qu'une vidéo exemple vous sont également fournies

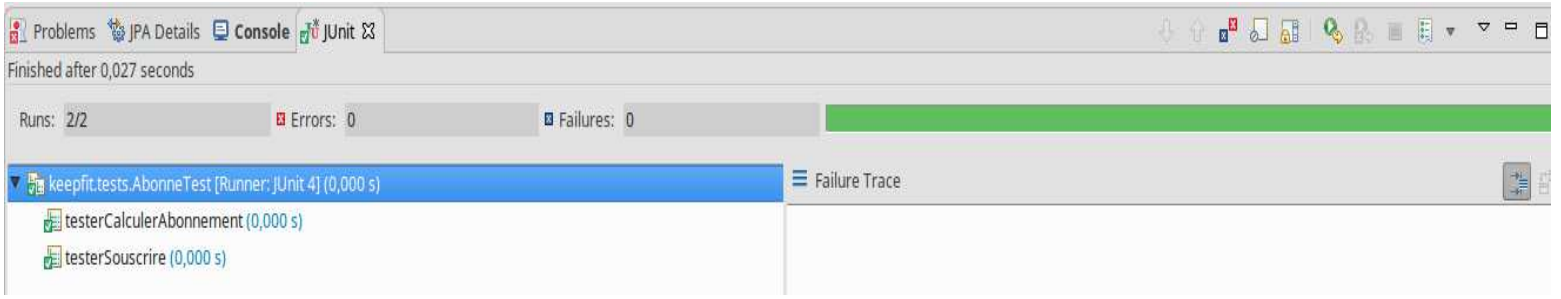
- ✓ Un système de journalisation (log4j+ commons.loging) est attendu : toutes les fonctionnalités métier seront loggées niveau INFO dans un fichier HTML et les exceptions métier de niveau WARN dans cette même page HTML (à la racine de votre projet) :

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

- ✓ Des tests unitaires sont indispensables : créer une classe JUnit 4 « AbonneTest » qui :
  - o teste le bon fonctionnement de la méthode calculerAbonnement de la classe Abonne
  - o 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.