Feuille de TP N°04

Exercice 1

On se propose de gérer un agenda de rendez-vous. L'objectif est donc de construire une classe `Agenda` comme agrégation d'un ensemble de rendez-vous (objets instances de la classe `RendezVous`).

Dans le répertoire TP4, vous y trouverez les fichiers ci-dessous :
- `Agenda.java` (à compléter)
- `RendezVous.java` (à compléter)
- `Horaire.java` (à compléter)
- `Test1.java` (pour tester vos implémentations)

1 La classe Horaire

La classe `Horaire` va nous être utile pour représenter les débuts des rendez-vous. Une instance de cette classe correspond comme son nom l'indique à un horaire, indiquant à la fois la date et l'heure.

1.1 Variables d'instance

La classe `Horaire` est caractérisée par :
- `jour` : Un entier compris entre 1 et le nombre de jour du mois
- `mois` : Un entier de [1, 12]
- `annee` : Un entier quelconque
- `heures` : Un entier de [0, 23]
- `minutes` : Un entier de [0, 59]

1.2 Constructeurs

La classe `Horaire` définie deux constructeurs :
- Un constructeur qui initialise les attributs d'instances avec des valeurs initiales explicites
  `public Horaire(int jour, int mois, int annee, int heures, int minutes)`
- Un constructeur par recopie
  `public Horaire( Horaire h)`
  l'objet receveur sera construit avec les mêmes caractéristiques que `h`.

1.3 Méthodes d'instance

La classe `Horaire` est munie des méthodes d'instance suivantes :
- les accesseurs et modifieurs (getters & setters)

– `public Horaire plus( int duree )` : Retourne l'horaire situé `duree` minutes après l'horaire receveur
– la méthode `equals` pour tester si deux horaires sont identiques (mêmes dates et mêmes heures).
– `public boolean apres( Horaire h)` qui retourne `true` si et seulement si le receveur correspond à un horaire strictement posterieur (supérieur) à `h`.
– public boolean apresOuSimultane(Horaire h) retourne `true` si et seulement si le receveur correspond à un horaire postérieur ou égal a `h`.
– `public boolean estValide()` return `true` si et seulement si l'horaire est valide (heure comprise entre `0:0` et `23:59` et `jour/mois/année` correspondant à un jour du calendrier)
– `String toString()` qui retourne une chaine de caractères représentant textuellement l'objet receveur sous la forme : le `jour/mois/annee` à `heures:minutes`

1.4 Méthodes de classe (statiques)

La classe `Horaire` contient les deux méthodes de classes suivantes :
– `private static boolean bissextile(int a)` : retourne `true` si et seulement si l'année passée en paramètre est bissextile
– `private static int nombreJours(int m, int a)` : retourne le nombre de jours que comporte le mois `m` (dans [1, 12]) de l'année `a`

1.5 Programme principal

Exécutez la méthode `main` de `Test1.java` afin de vérifier le bon fonctionnement de votre implémentation (ne consultez que les premiers affichages, car la méthode main teste également d'autres méthodes que vous n'avez pas encore implémentées).

1.6 Une documentation facile à réaliser

Vous connaissez deux moyens de disposer des commentaires dans un code source. // pour un commentaire tenant sur une seule ligne, et /* ... */ pour des commentaires s'étendant sur plusieurs lignes.

Dans `Horaire.java` figurent un nouveau type de commentaires : /** ... */. Ils sont utilisés par l'outils `javadoc` pour délimiter des commentaires devant apparaître dans la documentation de la classe.

Nous allons utiliser cet outil pour générer une documentation décrivant les interfaces des classes de notre projet.
– Faire un clic-droit sur votre projet et sélectionnez `[export]`.
– Cliquez sur `[javadoc]` (si vous ne le voyez pas, il figure dans le dossier `java` de la fenêtre) puis sur `[Next>]`.
– Vous indiquez dans le champs `[javadoc command]` le fichier `javadoc.exe` dans le sous-répertoire `bin` du répertoire où a été installé le `JDK` sur votre machine (si vous ne savez pas où trouver le fichier `javadoc.exe` sur votre machine, lancez une recherche sur votre disque dur), puis cliquez sur `[Finish]`.
– vous aurez éventuellement à confirmer le répertoire d'exportation en cliquant sur `[yes to all]`.

– Un fichier `Horaire.html` vient d'être créé dans le sous-répertoire `NomProjet/doc` de votre `workspace`. Vous pouvez y accéder depuis `eclipse` (il se trouve dans le dossier `doc` de votre projet dans le volet gauche). Ouvrez ce fichier.

En parcourant cette documentation, vous pourrez constater que les commentaires précédés du mot `@param` ont permis de préciser dans la documentation des indications sur les différents paramètres et que ceux précédés du mot `@return` ont permis d'indiquer ce que retournent les fonctions. Vous pourrez découvrir les autres balises en tapant `@` dans un commentaire `javadoc` sous `eclipse`.

Remarque 1 *Notez également que nous avons généré une documentation ne précisant que les méthodes* `public`*, celles utilisables en dehors de la classe. La classe* `Horaire` *contient également des méthodes privées (*`private`*). Il s'agit ici de deux méthodes générales déclarées* `static` *(*`bissextile` *et* `nombreJours`*). Ces méthodes étant à usage interne, il est normal de ne pas les faire apparaître dans la documentation de la classe.*

2 La classe RendezVous

La classe `Horaire` étant implémentée, nous allons pouvoir nous en servir pour créer des rendez-vous en agrégeant une instance de `Horaire` à un intitulé (de type `String`) et une durée (entière) en minutes.

La classe `RendezVous` est caractérisée par :
– `libelle` : Une chaine de caractères précisant le motif du rendez-vous
– `debut` : l'horaire de début du rendez vous, une référence (non nulle) indiquant l'horaire auquel doit commencer le rendez-vous, qui est une instance de la classe `Horaire`
– `duree` : durée en minute du rendez-vous (un entier positif)

Elle sera munie des méthodes suivantes :
– un constructeur :
`public RendezVous(String libelle, Horaire debut, int dureeEnMinutes)` avec valeurs initiales explicites pour les trois attributs d'instances. Implémentez ce constructeur **sans partage**[1] de l'horaire de début entre plusieurs rendez-vous.
– Des accesseurs et modifieurs (getters & setters) pour les attributs d'instance
– `public boolean estValide()` : test la validité de l'horaire (début de rendez-vous valide et durée positive ou nulle). Cette méthode sera implémentée par délégation (cf. la méthode `estValide` de la classe `Horaire`)
– `public boolean chevauche( RendezVous r )` : test de chevauchement entre rendez-vous
– `public String toString()` : Retourne une chaine décrivant textuellement l'objet receveur. À titre d'exemple, votre méthode devra retourner "[Examen : le 26/5/2014 a 8 :0 Duree : 90]" si elle est appelée sur un rendez-vous ayant lieu le 25 mai 2014 à 8h pour une durée de 90 minutes avec "Examen" pour intitulé.

1. Partager LA même instance de Horaire pour horaire de début entre plusieurs rendez-vous est utile pour des événements qui DOIVENT commencer au même instant. La modification de l'horaire partagé modifierait alors l'ensemble des rendez-vous l'ayant pour heure de début.

3 La classe Agenda

Nous souhaitons à présent implémenter la classe `Agenda` pour représenter un ensemble de rendez-vous. Pour cela, nous allons utiliser la classe prédéfinie `Vector`.

Remarque 2 *javadoc a été utilisé pour réaliser la documentation de toutes les classes fournies avec le jdk. Il suffit d'allez à l'adresse*
http ://docs.oracle.com/javase/7/docs/api/java/util/Vector.html (vous la trouverez en tapant "java vector" dans un moteur de recherche) pour consulter la documentation de la classe `Vector`.

- Implémentez les méthodes `getTaille`, `ajouter`, `supprimer` et `estVide` en faisant appel aux méthodes de `Vector` qui vous semblent les plus adéquates.
- Implémentez la méthode `toString`.

En plus des fonctionnalités de bases déjà présentes, nous souhaitons ajouter des méthodes nous permettant de vérifier les informations de l'agenda.
- Implémentez la méthode `estValide`.
- Implémentez la méthode `nombreChevauchements`.
- Consultez la méthode main de `Test1.java`.
- Pourquoi selon vous le déplacement de la réunion avec les membres de `IBM` supprime les chevauchements dans `aProf1` et en crée un dans `aProf2` ?

Bon travail !