TP Java : Un agenda

M.C. Lagasquie

27 juin 2019

On va définir en Java la classe AGENDA. Pour cela, il faut définir au préalable les classes suivantes :

- la classe HORAIRE qui permet de gérer l'objet composé heure–minutes,
- la classe DATE qui permet de gérer l'objet composé jour–mois–année,
- la classe RDV qui permet de gérer un rendez-vous composé d'un horaire, d'une date et d'un nom.

Attention :

- Pour les méthodes `equals` et `toString`, on veillera à respecter les prototypes hérités de la classe OBJECT.
- On utilisera le mécanisme des exceptions afin de traiter les cas d'erreur. La classe EXCEPTIONTP2 vous est fournie à cet effet (accessible sous Moodle).

Pour la classe HORAIRE, on définira :

- deux constructeurs, un avec deux paramètres entiers donnant l'heure et les minutes et l'autre avec pour seul paramètre un horaire à copier ;
- puis les méthodes suivantes :
  - `apres` : qui précise si l'horaire courant est *après* l'horaire passé en paramètre,
  - `equals` : qui compare l'horaire courant avec l'horaire passé en paramètre,
  - `toString` : qui retourne la chaîne de caractères représentant l'horaire courant.

Pour la classe DATE, on définira :

- deux constructeurs, un avec trois paramètres entiers donnant le jour, le mois et l'année et l'autre avec pour seul paramètre une date à copier ;
- puis les méthodes suivantes :
  - `apres` : qui précise si la date courante est *après* la date passée en paramètre,
  - `equals` : qui compare la date courante avec la date passée en paramètre,
  - `jourSuivant` : qui renvoie une date correspondant au jour suivant la date courante (attention aux années bissextiles),
  - `moisSuivant` : qui renvoie une date correspondant au mois suivant la date courante (attention aux années bissextiles),
  - `anneeSuivante` : qui renvoie une date correspondant à l'année suivant la date courante (attention aux années bissextiles),
  - `toString` : qui retourne la chaîne de caractères représentant la date courante.

Pour la classe RDV**, on définira :**

— deux constructeurs, un avec trois paramètres donnant la date, l'horaire et le nom et l'autre avec pour seul paramètre un rendez-vous à copier ;
— puis les méthodes suivantes :
  — `apres` : qui précise si le rendez-vous courant est *après* le rendez-vous passé en paramètre,
  — `equals` : qui compare le rendez-vous courant avec le rendez-vous passé en paramètre,
  — `getDate` : qui renvoie la date du rendez-vous courant,
  — `getHoraire` : qui renvoie l'horaire du rendez-vous courant,
  — `getNom` : qui renvoie le nom du rendez-vous courant,
  — `setDate` : qui modifie la date du rendez-vous courant en la remplaçant par la date donnée en paramètre,
  — `setHoraire` : qui modifie l'horaire du rendez-vous courant en le remplaçant par l'horaire donné en paramètre,
  — `setNom` : qui modifie le nom du rendez-vous courant en le remplaçant par le nom donné en paramètre,
  — `toString` : qui retourne la chaîne de caractères représentant le rendez-vous courant.

Pour la classe AGENDA**, on définira :**

— un constructeur qui fournit un agenda vide ;
— puis les méthodes suivantes :
  — `ajoutRDV` : qui ajoute un rendez-vous à l'agenda courant,
  — `getRDVParDateEtHeure` : qui renvoie le rendez-vous correspondant à la date et à l'horaire passés en paramètre de l'agenda courant,
  — `getRDVParNom` : qui renvoie le rendez-vous correspondant au nom passé en paramètre de l'agenda courant,
  — `supprimeRDVParDateEtHeure` : qui supprime de l'agenda courant le rendez-vous correspondant à la date et à l'horaire passés en paramètre,
  — `supprimeRDVParNom` : qui supprime de l'agenda courant le rendez-vous correspondant au nom passé en paramètre,
  — `toString` : qui retourne la chaîne de caractères représentant l'agenda courant.

Une attention particulière devra être portée aux tests. Pour cela, une classe TESTAGENDA vous est fournie (accessible sous Moodle) que vous installerez dans votre projet Java. Cette classe contient une méthode `main` permettant d'exécuter *automatiquement* un jeu de tests le plus complet et le plus lisible possible. Vous ne devrez pas modifier cette classe ! Et pour qu'elle fonctionne correctement, vous devrez respecter *scrupuleusement* les prototypes des constructeur et des méthodes demandés.

Cela ne vous dispensera pas d'écrire votre propre méthode `main` dans chacune de vos classes afin de tester *au fur et à mesure* votre travail.

Nous vous rappelons que, dans le cadre des évaluations, vos classes seront aussi exécutées sur d'autres jeux de tests non fournis.