HAUTE ÉCOLE DE
NAMUR-LIÈGE-LUXEMBOURG
Catégorie technique
Année académique : 2020-2021

TI215 : programmation orientée objet 1
Exercices : série 1

Domaine d'application : **Tournoi de jeux**
Chaque rencontre du tournoi de jeux oppose deux équipes de joueurs.
Une équipe, dite équipe locale, reçoit une équipe de joueurs visiteurs.
Au fur et à mesure du déroulement de la rencontre, des points sont attribués à chacune des équipes par un arbitre.
Le fair-play étant important, l'arbitre peut également pénaliser une équipe en lui attribuant des fautes.

Objectif 1 : Ecrire des classes, créer des objets et appeler des méthodes sur des objets

N.B. *Créez la classe* ***Rencontre (cf. 1.1.)*** *et la classe* ***Principal (cf. 1.2.)*** *en parallèle : n'attendez pas d'avoir créé toutes les méthodes de la classe Rencontre pour les tester. Créez une méthode à la fois et testez-la directement.*

1.1. Créez un projet intitulé ***Série1*** puis un package ***série1***. Créez ensuite dans ce package une classe intitulée **Rencontre** qui contient les **variables d'instances** suivantes :
de type String :
- ***nomLocaux*** (nom de l'équipe qui reçoit)
- ***nomVisiteurs*** (nom de l'équipe des joueurs visiteurs)

de type int :
- ***pointsLocaux*** (nombre de points de l'équipe locale)
- ***pointsVisiteurs*** (nombre de points de l'équipe des visiteurs)
- ***nbFautesLocaux*** (nombre de fautes de l'équipe locale)
- ***nbFautesVisiteurs*** (nombre de fautes de l'équipe des visiteurs).

Prévoyez un **constructeur** comprenant *autant d'arguments qu'il y a de variables d'instance.*

Ajoutez les **méthodes** suivantes :

- une méthode appelée ***vainqueur*** qui retourne le nom de l'équipe victorieuse ou le message « Ex æquo »;

- une méthode appelée ***equipeFairPlay*** qui retourne le nom de l'équipe qui a fait le moins de fautes ou le message « Ex aequo » ;

- une méthode appelée ***exAequo*** qui retourne un booléen : vrai (true) si les équipes ont toutes deux le même nombre de points, faux (false) sinon ;

- une méthode appelée ***presenterLocaux*** qui retourne une chaîne de caractères (de type String) présentant l'équipe locale (respectez le format ci-dessous) :
  soit, par exemple, une rencontre avec ***nomLocaux*** = *la guilde de Talrant,*
  la méthode *presenterLocaux* doit retourner la chaîne de caractères :
  L'équipe locale *la guilde de Talrant*

- une méthode appelée ***presenterVisiteurs*** qui retourne une chaîne de caractères (de type String) présentant l'équipe des visiteurs (respectez le format ci-dessous) :
  soit, par exemple, une rencontre avec ***nomVisiteurs*** = *les faucons de Valmort*
  la méthode *presenterVisiteurs* doit retourner la chaîne de caractères :
  l'équipe des visiteurs *les faucons de Valmort*

- une méthode appelée ***ajouterPointsLocaux*** qui ajoute 1 point au total des points de l'équipe locale ;

- une méthode appelée ***ajouterPointsVisiteurs*** qui ajoute 1 point au total des points de l'équipe des visiteurs ;

- une méthode appelée ***ajouterFauteLocaux*** qui ajoute 1 faute à l'équipe locale ;

- une méthode appelée ***ajouterFauteVisiteurs*** qui ajoute 1 faute à l'équipe des visiteurs.

1.2. Créez dans le même package une classe **Principal** contenant la méthode **main**. Cette méthode doit :

- créer *au moins un* objet de type **Rencontre** ;

- ajouter des points et des fautes aux deux équipes de la rencontre *en faisant appel aux méthodes adéquates* ;

- afficher à l'écran *en faisant appel aux méthodes adéquates*:
  - la présentation des deux équipes de la rencontre;
  - le nom de l'équipe victorieuse ;
  - le nom de l'équipe la plus « fair-play » ;

- tester *en faisant appel à la méthode adéquate* s'il s'agit d'une rencontre où les équipes sont ex aequo.

Objectif 2 : Appeler au sein d'une méthode une autre méthode de la même classe

1.3. Ajoutez une méthode appelée ***presenterAdversaires*** qui retourne une chaîne de caractères présentant les deux équipes (respectez le format ci-dessous) :

Soit, par exemple, une rencontre avec

nomLocaux = *la guilde de Talrant*

nomVisiteurs = *les faucons de Valmort*

la méthode *presenterAdversaires* doit retourner la chaîne de caractères :

L'équipe locale *la guilde de Talrant* ***reçoit***

l'équipe des visiteurs *les faucons de Valmort****.***

Attention : vous devez impérativement **faire appel aux méthodes existantes** *presenterLocaux* et *presenterVisiteurs*. Testez ensuite cette méthode sur l'objet de type **Rencontre** créé dans la méthode **main** de la classe **Principal**.

1.4. Ajoutez une méthode appelée ***ajouterPoints***

- qui reçoit un argument: un caractère désignant l'équipe ('L' pour l'équipe locale, 'V' pour l'équipe des visiteurs) et qui ajoute 1 point au total des points de l'équipe correspondante.

Attention :

Vous devez impérativement **faire appel aux méthodes existantes** *ajouterPointsLocaux* et *ajouterPointsVisiteurs*.

Testez ensuite cette méthode sur l'objet de type **Rencontre** créé dans la méthode **main** de la classe **Principal**.

Objectif 3 : Surcharger les constructeurs

1.5.1. Ajoutez un *second constructeur* ne comprenant que deux arguments (les noms des deux équipes) qui initialise à 0 les points et le nombre de fautes des deux équipes.

1.5.2. Ajoutez un *troisième constructeur* qui concerne toutes les rencontres dont l'équipe locale est « Les seigneurs du combat » et qui initialise à 0 les points et le nombre de fautes des deux équipes.

Testez ensuite ces deux constructeurs dans la méthode main de la classe **Principal** en créant de nouvelles rencontres et en les exploitant ensuite (cf. 1.2).

Objectif 4 : Appeler implicitement la méthode toString( )

1.6. Prévoyez dans la classe **Rencontre** la méthode ***toString( )*** qui décrit une rencontre suivant le format ci-dessous. Vous devez impérativement **faire appel aux méthodes** *presenterAdversaires* et *vainqueur.*

Ex : Si la méthode *presenterAdversaires* retourne la chaîne de caractères suivante:

L'équipe locale *la guilde de Talrant* ***reçoit***
l'équipe des visiteurs *les faucons de Valmort.*

et si la méthode *vainqueur* déclare victorieuse l'équipe locale,
la méthode toString( ) doit retourner la chaîne de caractères suivante :

L'équipe locale *la guilde de Talrant* ***reçoit***
l'équipe des visiteurs *les faucons de Valmort.*
Equipe victorieuse *: la guilde de Talrant*

Testez dans la méthode main de la classe **Principal**, l'appel implicite à cette méthode toString( ) en demandant d'afficher une rencontre (via l'instruction ***System.out.println(r)***, r étant un objet de la classe **Rencontre**).

Objectif 5 : Surcharger les méthodes

1.7. Surchargez les méthodes ***ajouterPointsLocaux*** et ***ajouterPointsVisiteurs*** en créant :

- une méthode appelée *ajouterPointsLocaux* **qui reçoit en argument un nombre de points** et qui ajoute ce nombre au total des points de l'équipe locale ;

- une méthode appelée *ajouterPointsVisiteurs* **qui reçoit en argument un nombre de points** et qui ajoute ce nombre au total des points de l'équipe des visiteurs ;

Testez ensuite le mécanisme de surcharge de méthodes en appelant sur le même objet (rencontre) la même méthode d'abord sans argument puis avec argument. Pour vous convaincre de l'effet de ces appels de méthode, affichez après chaque appel le nombre de points de l'équipe correspondante.