Institut Paul Lambin Session de janvier 2020

Examen d’Algorithmique

G. Seront – A. Dupont – J. Vander Meulen – A. Legrand – L. Lecharlier – I.Cambron

Informatique de Gestion - Bloc 1

Durée de l’examen : 2 h30 ; pas de sortie durant les 60 premières minutes

L’examen se déroule en 2 phases.

Le premier questionnaire se remplit sans PC.

Vous recevrez le deuxième questionnaire et vous aurez accès au PC lorsque vous aurez remis le premier questionnaire.

PHASE 2 : AVEC PC :

* **Notez sur cette page de garde vos nom et prénom et votre login Windows, ainsi que le numéro du PC sur lequel vous travaillez (il se trouve à l’arrière du PC ; par exemple LEN1301)**
* **A la racine de votre Z:/ vous trouverez deux fichiers .java**
* **Complétez le fichier ClassementDesFinalistes.java avec l’éditeur de votre choix.**
* **Une fois votre code terminé :**
  + **Créez à la racine du drive Z : un répertoire portant vos noms en suivant la nomenclature suivante : NOM\_PRENOM. Par exemple SERONT\_GREGORY.**
  + **copiez vos solutions dans le répertoire portant votre nom.**
* **A titre de vérification, ouvrez ces fichiers avec Notepad.**
* **L’examen se termine quand vous aurez montré au surveillant votre code et remis le questionnaire. Pensez à fermer votre session.**

Un client aimerait avoir une application pour gérer le classement des finalistes d’un concours. Dans celui-ci, les participants doivent gagner des points lors d’épreuves. Dès qu’un participant a des points, il sera considéré comme un finaliste à condition que le nombre maximum de finalises acceptés ne soit pas atteint.

On vous demande d’implémenter la classe ClassementDesFinalistes va gérer les résultats et le classement.

Cette classe utilise

* l’entier nombreDeFinalistes qui représente le nombre de participants ayant déjà des points et se trouvant donc en finale ;
* une table de String finalistes contenant les noms des finalistes ;
* une table d’entiers points pour leurs points respectifs **classés par ordre décroissant**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| loic | luc | jean |  |  |

Exemple :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 7 | 3 |  |  |

finalistes =

points =

nombreDeFinalistes = 3

Cela signifie que pour l’instant loic est premier avec 10 points, luc deuxième avec 7 points et jean troisième avec 3 points et qu’il reste encore deux places de finaliste disponibles.

Vous devrez implémenter 4 méthodes de cette classe.

# Méthode ajouter

La méthode ajouter() va essayer soit d’ajouter des points à un finaliste soir d’ajouter un nouveau finaliste.

Donc la méthode ajouter(**int** pts, String participant) va ajouter pts points à participant si celui-ci est finaliste ou l’ajouter aux finalistes s’il reste une place de finaliste. En cas d’ajout, la méthode fait en sorte que le table points reste triée par ordre décroissant tout en maintenant le nom du finaliste au même indice que ses points. De plus la méthode va renvoyer true si l’ajout a pu se faire et false sinon.

Exemples :

Si

|  |  |  |
| --- | --- | --- |
| loic | Luc | jean |

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 8 | 7 |

points =

nombreDeFinalistes = 3

alors

ajouter(9, « pierre ») va renvoyer false car les tables sont pleines et « pierre » n’est pas dans la table finalistes. Malgré qu’il soit meilleur, il ne sera pas repris.

|  |  |  |
| --- | --- | --- |
| loic | Luc | jean |

Si

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 8 | 7 |

points =

nombreDeFinalistes = 3

alors

|  |  |  |
| --- | --- | --- |
| loic | jean | Luc |

ajouter(2, « jean ») renvoie true, et la situation après ajout est

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 9 | 8 |

points =

nombreDeFinalistes = 3

Si

|  |  |  |
| --- | --- | --- |
| loic | Luc |  |

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 8 |  |

points =

nombreDeFinalistes = 2

alors

ajouter(11, « toto ») renvoie true, et la situation après ajout est

|  |  |  |
| --- | --- | --- |
| toto | loic | luc |

finalistes =

|  |  |  |
| --- | --- | --- |
| 11 | 10 | 8 |

points =

nombreDeFinalistes = 3

# Méthode premier

La méthode premier() renvoie une table contenant le(s) nom(s) du ou des premiers (en cas d’exæquo) dans l’ordre où il(s) se trouve(nt) dans la table finalistes.

Par exemple

Si

|  |  |  |
| --- | --- | --- |
| loic | Luc |  |

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 8 |  |

points =

nombreDeFinalistes = 2

|  |
| --- |
| loic |

alors

premier() renvoie

Si

|  |  |  |
| --- | --- | --- |
| loic | Luc | jean |

finalistes =

|  |  |  |
| --- | --- | --- |
| 10 | 10 | 5 |

points =

nombreDeFinalistes = 3

alors

|  |  |
| --- | --- |
| loic | Luc |

premier() renvoie

S’il n’y a pas encore de finaliste alors premier()renvoie une table vide.

# Méthode ecartMaximum

La méthode ecartMaximum() calcule le plus grand écart de points entre 2 finalistes qui se suivent dans le classement.

Par exemple :

Si

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| loic | luc | jean | toto | isa |  |

finalistes =

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 10 | 6 | 3 | 2 |  |

points =

nombreDeFinalistes = 5

alors

ecarMaximum() renvoie 4 car le plus grand écart se trouve entre luc et jean et est de 4 points.

S’il n’y a pas encore de finaliste alors ecartMaximum()renvoie 0.

# Méthode disqualifier

La méthode disqualifier() va essayer de retirer le participant participant des finalistes.

Donc la méthode disqualifier(String participant) va supprimer participant dans la table finalistes, supprimer ses points de la table points et renvoyer true si participant est présent dans la table finalistes. Sinon la méthode ne fait rien et renvoie false. En cas de suppression, la méthode fait en sorte que le table points reste triée par ordre décroissant tout en maintenant le nom du finaliste au même indice que ses points.

Par exemple :

Si

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| loic | luc | jean | toto | isa | elsa | marc |  |

finalistes =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 9 | 8 | 6 | 6 | 2 |  |

points =

nombreDeFinalistes = 7

alors

disqualifier(«phil»)renvoie false

Si

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| loic | luc | jean | toto | isa | elsa | marc |  |

finalistes =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 9 | 8 | 6 | 6 | 2 |  |

points =

nombreDeFinalistes = 7

alors

disqualifier(«jean»)renvoie true et

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| loic | luc | toto | isa | elsa | marc |  |  |

finalistes =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 10 | 8 | 6 | 6 | 2 |  |  |

points =

nombreDeFinalistes = 6

# Jeux de tests

Nous vous fournissons la classe de test TestClassementDesFinalistes contenant quelques cas intéressants. Celle-ci contient un menu vous permettant de tester les méthodes séparément sans obligation que les autres méthodes fonctionnent.

# Consigne obligatoire

N’utilisez pas les méthodes déjà existantes des classes String, System,Arrays, ArrayList, …

# Derniers Conseils

* Lisez bien entièrement l’énoncé avant de commencer à coder.

**Brouillon :**

**Brouillon :**