Evaluation 1 - partie pratique

Le boardercross

Travail à accomplir

Le boardercross est une course de snowboard où le principe est simple : quatre riders s'élancent en même temps sur un parcours créé artificiellement. Avant de franchir la ligne d'arrivée, ils doivent effectuer divers sauts, enchaîner les whoops (série de petites vagues) et les virages relevés. Le classement est basé sur le système d'élimination directe, les deux coureurs les plus rapides se qualifiant pour l'étape suivante.

On vous demande de réaliser un programme permettant de simuler une course de boardercross.

Attention : Pour les besoins de l'exercice, on va changer légèrement les règles puisqu'on qualifiera les riders uniquement sur la base de leur temps et non pas de leur rang final dans la série.

Deux tableaux nous permettrons de gérer cela. Dans le premier, on retrouvera le nom des riders engagés. Dans le deuxième, les temps réalisés (en secondes) à chaque tour de la compétition.

Exemple :

Après le premier tour (quart de final) :

Tableaux des riders :

Kilian	Max	Noé	Beat	André	Alex	John	Fred

Tableaux des temps :

55	40	38	42	47	51	43	60

Kilian a donc réalisé un temps de 55s, Max un temps de 40s, etc…

Après le deuxième tour (demi-finale) :

Tableaux des riders :

Max	Noé	Beat	John

Tableaux des temps :

38	45	39	50

Après le troisième tour (finale) :

Tableaux des riders :

Max	Beat

Tableaux des temps :

48	43

Le gagnant de la compétition est donc Beat !

Etapes à réaliser

1. Créez un nouveau projet Java dans NetBeans que vous nommerez **403_E1_NOM** (exemple 403_E1_ROUILLER).
2. Créez les constantes `TEMPS_MIN` et `TEMPS_MAX` qui contiendront des entiers avec, respectivement, les valeurs suivantes : 35 et 75.
3. Créez une méthode nommée `genererTemps`. Cette méthode sera responsable de créer et retourner un tableau d'entiers. La taille du tableau sera passée en paramètre et les cellules contiendront des valeurs aléatoires comprises entre `TEMPS_MIN` et `TEMPS_MAX`.
4. Créez une méthode nommée `positionMeilleurTemps`. Cette méthode prendra en paramètre un tableau d'entiers nommé `tabTemps` et elle retournera l'index de la cellule contenant la plus petite valeur. En cas d'égalité, ce sera l'index de la première valeur trouvée qui sera retourné.
5. Créez la méthode `calculQualification`. Cette méthode prendra en paramètre un tableau de String nommé `riders` qui représentera les riders encore engagés dans la compétition. Cette méthode prendra également en paramètre un tableau d'entiers nommé `tabTemps` qui représentera les temps réalisés par les riders encore engagés dans la compétition. Le but de cette méthode est de retourner un tableau de String avec les riders qualifiés pour le tour suivant. Pour cela, on retiendra la moitié des riders engagés sur la base de leur temps (comme montré dans l'exemple de la première page). Voici les étapes afin de réaliser cela :
   5.1. Déclarez et créez le tableau de String `ridersRestants` dont la taille sera de la moitié du tableau `riders` passé en paramètre.
   5.2. Pour chaque cellule du tableau `ridersRestants` :
      5.2.1. Utilisez la méthode `positionMeilleurTemps` qui désignera le rider à retenir dans le tableau `riders`. Affectez ensuite ce rider dans la cellule courante du tableau `ridersRestants`.
      5.2.2. Affectez `Integer.MAX_VALUE` comme valeur à la cellule du tableau `tabTemps` désignée par le résultat de l'appel à la méthode `positionMeilleurTemps` exécutée au point précédent. Cette opération permet d'exclure le meilleur temps actuel et de déterminer ainsi le prochain meilleur temps lors de l'itération suivante.
6. Dans la méthode `main` :
   6.1. Déclarez la variable `riders` qui sera un tableau de String initialisé directement avec les valeurs suivantes : "Kilian", "Max", "Noé", "Beat", "André", "Alex", "John", "Fred".
   6.2. Déclarer la variable `tour` qui sera en entier représentant le numéro du tour. Vous pouvez directement affecter la valeur 1 à cette variable.
   6.3. Tant que le tableau `riders` a une taille d'au moins 2 :
      6.3.1. Déclarez la variable `tabTemps` qui sera un tableau d'entiers initialisé directement avec le retour de l'appel à la méthode `genererTemps`. Il s'agit de générer autant de temps qu'il y a de rider (indication utile pour savoir quel paramètre donner à l'appel de la méthode).
      6.3.2. A l'aide d'un switch déterminez le traitement à réaliser en fonction de la variable `tour`.
         6.3.2.1. Si la variable est égale à 1, affichez : « `Résultat du quart de final :` »
         6.3.2.2. Si la variable est égale à 2, affichez : « `Résultat de la demi-finale :` »
         6.3.2.3. Si la variable est égale à 3, affichez : « `Résultat de la finale :` »

6.3.3. Affichez ensuite les noms et les temps de tous les riders encore en lice. Attention aux retours à la ligne qui doivent être conformes à ce qui est présenté plus bas dans l'exemple de résultat.

6.3.4. Après avoir afficher tous les riders, affichez une séparation avec le texte : « ----------». Là aussi, vous êtes invités à voir l'exemple de résultat pour avoir le même affichage dans votre programme.

6.3.5. Incrémentez la variable `tour`.

6.3.6. Appelez la méthode `calculQualification`, afin d'obtenir le nouveau tableau de `riders`.

6.4. Affichez le nom du gagnant de la compétition.

Exemple de résultat à la console :

```
Résultat du quart de final :
Kilian [48s], Max [66s], Noé [63s], Beat [38s], André [56s], Alex [70s], John [57s], Fred [59s]
----------
Résultat de la demi-finale :
Beat [73s], Kilian [53s], André [57s], John [49s]
----------
Résultat de la finale :
John [46s], Kilian [54s]
----------
Le gagnant est : John
```

Restitution

Lorsque vous avez terminé, faites-signe au professeur pour lui remettre votre travail.

N'oubliez pas de formater votre code !