SAE 1.01 - Groupe 3B3

uwu

Guychel BABELA, Baptiste
MARTIN, Vincent BARETTE

Description du rôle de l'application **2**
- Rappel du contexte 2
- Principe du projet 2
- Saisie 2
- Traitement 2
- Affichage 2

Répartition des tâches **3**
- Vincent 3
  - Traitement des données 3
  - Finitions 3
- Guychel 3
  - Saisie 3
- Baptiste 3
  - Affichage 3
- Tiers 3

Jeux d'essais **3**
- additionTableaux 3
- saisirJoueurs 4
- trierJoueurs 4
- echangeNbre 4
- indMin 4
- afficherClassementFinal 4

Tchat privé (à supprimer impérativement) **5**

Description du rôle de l'application

Rappel du contexte

Le client pour cette SAE nous est imposé, et il s'agit d'une organisation proposant des courses de canoë. Des arbitres sont présents lors des courses, mais ne sont pas forcément d'accord sur les erreurs que les joueurs pourraient avoir commises. Ils font donc face à une problématique, et nous devons implémenter une application qui y répond.

Principe du projet

L'application a pour objectif d'aider les arbitres à faire se dérouler la course sans contestation. Le programme permettra aux arbitres de saisir les données de la course, et les temps des joueurs sur les deux manches. Il traitera ensuite toutes les données en arrière-plan, effectuera un classement des joueurs. Il affiche ensuite le podium, en sélectionnant les 3 meilleurs joueurs.

Saisie

L'application demande à l'arbitre de saisir les éléments suivants.
Saisie des données de la course

1. le nombre de portes
2. le nombre de joueurs
3. les données de la manche 1, pour chaque joueur
    a. le booléen de qualification du joueur
    b. le temps du joueur
    c. le nombre de portes touchées
    d. le nombre de portes ratées
4. les données de la manche 2, pour chaque joueur qui n'a pas été disqualifié à la manche 1
    a. le booléen de qualification du joueur
    b. le temps du joueur
    c. le nombre de portes touchées
    d. le nombre de portes ratées

Traitement

L'application traite les informations précédentes, et calcule notamment le temps composé, additionne le temps des deux courses, et en déduit le classement final.

Affichage

L'application affiche un podium représentant le top 3 des résultats, avec le temps qu'ils ont mis.

Mec, à quand le top 10 besoins mdrr
jsp mec vien on se fait ca pendant les vacances avec l'album ligne A XD
XDDDD génie, go faire ça (mais pour l'instant, sae 🤡😹👎😣

Répartition des tâches

Vincent

Traitement des données

- traitementDonnees()
- tempsFinal()

Finitions

- Travail divers
- Amélioration des fonctions
- Synchronisation de toutes les parties du code

Guychel

Saisie

- saisieIntC()
- saisirJoueurs()

Baptiste

Affichage

- trierJoueurs()
- echangeNbre()
- indMin()
- afficherClassementFinal()

Tiers

Aucune tierce source.

Jeux d'essais

additionTableaux

manche1	manche2	nbJoueurs	elimination	OUT tabFinal
{{10,0,0},{20,0,0}}	{{20,0,0},{40,0,0}}	2	{0, 0}	{30, 60}
{{0,0,0}}	{{0,0,0}}	1	{1}	{999999999}
{{8521,0,0}}	{{0,0,0}}	1	{1}	{999999999}

saisirJoueurs

nbPortes	nbJoueurs	OUT pfTab	OUT elimination
18	2	{{10,2,1}, {0,0,0}}	{0,1}
18	1	{{19,18,0}}	{0}

trierJoueurs

listeJ	nomJoueurs	**output**
{12,19,48}	{5,9,2}	joueurs = {48,19,12} list = {2,9,5}
{2,31,17,9,11}	{2,3,7,4,6}	joueurs = {2,31,9,17,11} list = {2,3,4,7,6}

echangeNbre

temp	ind	joueurs	list	i	**output**
12	2	{12,19,48}	{5,9,2}	0	joueurs = {48,19,12} list = {2,9,5}
7	3	{2,31,17,9,11}	{2,3,7,4,6}	2	joueurs = {2,31,9,17,11} list = {2,3,4,7,6}

indMin

listeJ	point	min	ind	**output**
{5,9,2}	0	2	2	return (2)
{2,3,7,4,6}	2	4	3	return (3)

afficherClassementFinal

Liste	**output**

{3,439,2,349,349,349}	Premier : 3 Deuxième : 1 Troisième : 4 5 6
{3,3,2}	Podium : Premier : 3 Deuxièmes : 2 , 1

Code

```
/**
 * SAE 1.1
 *
 * @3B3
 */

import java.util.Scanner;
public class CourseCanoë
{
    /** Guychel
     * Demande à l'utilisateur de saisir un entier, jusqu'à ce que
     * l'entier saisi soit entre les deux bornes en paramètres.
     *
     * @param pfBorneMin IN : borne minimale
     * @param pfBorneMax IN : borne maximale
     *
     * @return un entier entre pfBorneMin et pfBorneMax, compris
     */
    public static int saisieIntC (int pfBorneMin, int pfBorneMax){
        int valeur;
        Scanner clavier = new Scanner(System.in) ;
             System.out.println("Donnez une valeur comprise entre
"+pfBorneMin+" et "+pfBorneMax+ "?");
        valeur = clavier.nextInt();
        while (valeur<pfBorneMin || valeur>pfBorneMax) {
                  System.out.println("Erreur ! Donnez une valeur
comprise entre "+pfBorneMin+" et "+pfBorneMax+ "?");
            valeur = clavier.nextInt();
        }
        return valeur;
    }

    /**
     * Programme principal
     *
     * @param pfTab OUT : tableau à remplir
     *
     * @return le nombre de cases remplies dans le tableau
```

```
     */
    public static void main() {
        /** Vincent
         * Programme principal.
         *
         * Pas de paramètre ni de sortie.
         *
         */

        /*
         * INITIALISATION
         */
        Scanner clavier = new Scanner(System.in);
        int nbPortes;
        System.out.println("Nombre de portes du parcours : ");
        nbPortes = saisieIntC(18,22);

            int[][] pfTab1 = new int [49][3]; // Tableau de la
première manche
            int[][] pfTab2 = new int [49][3]; // Tableau de la
deuxième manche
          int[] disqualification = new int [49]; // Même tableau
utilisé pour les 2 manches.
        int[] classementFinal = new int [49];

        int nbJoueurs = 0 ;

        /*
         * SAISIE
         */

        System.out.println("Nombre de joueurs dans la partie : ");

        // Par défaut, aucun joueur n'est éliminé.
        for (int i = 0; i < nbJoueurs; i++) {
            disqualification[i] = 0;
        }

        System.out.println("Chronométrage de la Manche 1");
        // Manche 1
                saisirJoueurs(pfTab1, nbPortes, disqualification,
nbJoueurs);

        System.out.println("Chronométrage de la Manche 2");
        // Manche 2
                saisirJoueurs(pfTab2, nbPortes, disqualification,
nbJoueurs);
```

```
        /*
         * TRAITEMENT DES DONNEES
         */

        // Manche 1
        traitementDonnees(pfTab1, nbJoueurs);

        // Manche 2
        traitementDonnees(pfTab2, nbJoueurs);

        // Addition des résultats

        /*
         * AFFICHAGE
         */

                additionTableaux(classementFinal, pfTab1, pfTab2,
nbJoueurs, disqualification);

        afficherClassementFinal(classementFinal);

        return;
    }

    /** Vincent
     * Demande à l'utilisateur :
       *    1. de saisir le nombre de joueurs qui sont dans la
partie, et
     *      répète l'opération, jusqu'à ce que ce nombre soit
     *      acceptable
     *   2. de remplir les cases une à une.
     *
     * @param pfTab OUT : tableau à remplir
     *
     * @return le nombre de cases remplies dans le tableau
     */
      public static void additionTableaux (int[] classementFinal,
int[][] manche1, int[][] manche2, int nbJoueurs, int[]
elimination){
        /**
          * Demande à l'utilisateur de saisir un entier, jusqu'à ce
que
               * l'entier saisi soit entre les deux bornes en
paramètres.
         *
         * @param classementFinal IN : tableau 1D vide
          * @param manche1 IN : tableau d'entiers des joueurs de la
manche 1
```

```
          * @param manche2 IN : tableau d'entiers des joueurs de la
manche 2
          * @param nbJoueurs IN : entier, nombre de joueurs
            * @param elimination IN : tableau d'entiers élimination
des joueurs
          *
            * @return Un tableau de temps finaux, avec comme indice
le dossard du joueur.
          */
         for (int i = 0; i < nbJoueurs; i++) {
             if (elimination[i] == 0) {
                             classementFinal[i]  =  manche1[i][0]  +
manche2[i][0];
             } else {
                 classementFinal[i] = 999999999;
             }
         }
     }

     /** Guychel
      * Demande à l'utilisateur :
        *     1. de saisir le nombre de joueurs qui sont dans la
partie, et
      *       répète l'opération, jusqu'à ce que ce nombre soit
      *       acceptable
      *    2. de remplir les cases une à une.
      *
      * @param pfTab OUT : tableau à remplir
      *
      * @return le nombre de cases remplies dans le tableau
      */
     public static void saisirJoueurs (int[][] pfTab, int nbPortes,
int[] elimination, int nbJoueurs){
         Scanner clavier = new Scanner(System.in) ;
         for(int i = 0 ; i < nbJoueurs ; i++){
             int res = 0;
             while (elimination[i] == 0 && (res < 1 || res > 2)) {
                       System.out.println("Le joueur {"+i+"} est-il
disqualifié ? 1 pour OUI, 2 pour NON");
                 res = clavier.nextInt();
             }
             if (elimination[i] == 0 && (res == 2)) {
                  System.out.println("Saisir le temps du joueurs : "
+ i);
                 pfTab[i][0] = clavier.nextInt();
                 pfTab[i][1] = 23;
                 while(pfTab[i][1] > nbPortes){
```

```
                    System.out.println("Saisir le nombre de portes
touchées : " + i);
                    pfTab[i][1] = clavier.nextInt();
                }
                pfTab[i][2] = 23;
                while(pfTab[i][1] + pfTab[i][2] > nbPortes){
                     System.out.println("Saisir le temps de portes
ratées : " + i);
                    pfTab[i][2] = clavier.nextInt();
                }
            } else {
                elimination[i] = 1;
            }
            res = 0;
        }
        System.out.println("J'avais " + pfTab.length
            + " cases disponibles, et j'en ai rempli "
            + nbJoueurs);
        return;
    }

    /** Vincent
       * Calcule le temps final de chacun des joueurs d'une même
manche
     *
     * @param tab IN OUT : tableau d'entiers d'une manche
     * @param nbJoueurs IN : entier, nombre de joueurs
     *
     * @return Rien
     */
        public static void traitementDonnees(int[][] tab, int
nbJoueurs) {
        for (int i = 0; i < nbJoueurs; i++) {
                    tab[0][i] = tempsFinal(tab[0][i], tab[1][i],
tab[2][i]);
        }
        return;
    }

    /** Vincent
     * Calcule le temps final pour un cas
     *
     * @param tab IN : tableau d'entiers d'une manche
     * @param nbJoueurs IN : entier, nombre de joueurs
     *
     * @return Temps final
     */
    public static int tempsFinal(int ms, int touch, int miss) {
```

```
        return (ms + 2000 * touch + 50000 * miss);
    }

    /** Baptiste
     * Fonction créant une liste triée par une boucle for
     *
     * @param listeJ IN : liste d'entiers, temps des joueurs
       * @param nomJoueurs IN : nom des joueurs (par défaut, leur
numéro de dossard)
     *
     * @return Rien
     */
        public static void trierJoueurs(int[] listeJ, String[]
nomJoueurs){
        for (int i = 0 ; i < listeJ.length ; i ++){
            echangeNbre(listeJ, nomJoueurs, i);
        }
    }

    /** Baptiste
     * Fonction permettant d'échanger la valeur max d'une liste
     * avec son itération d'indice i
     *
     * @param list IN : liste d'entiers, subit les modifications
       * @param joueurs IN : nom des joueurs (par défaut, leur
numéro de dossard)
     * @param i IN : indice à échanger
     *
     * @return Rien
     */
     public static void echangeNbre(int[] list, String[] joueurs,
int i){
        int temp = list[i];
        String temp2 = joueurs[i];
        int ind = indMin(list,i);
        list[i] = list[ind];
        joueurs[i] = joueurs[ind];
        list[ind] = temp;
        joueurs[ind] = temp2;
    }

    /** Baptiste
     * Fonction renvoyant l'indice du minimum d'une liste
     *
     * @param listeJ IN : liste d'entiers
     * @param point IN : taille de la liste
     *
     * @return L'indice du maximum de listeJ
```

```
     */
    public static int indMin(int[] listeJ, int point){
        int min = 10000000;
        int ind = 0;
        for (int i = point ; i < listeJ.length ; i ++){
            if (listeJ[i] < min){
                min = listeJ[i];
                ind = i;
            }
        }
        return ind;
    }

    /** Baptiste 85% (Vincent (finitions) 15%)
     * Fonction triant la liste des joueurs, et affiche le podium.
     *
     * @param listeJ IN : liste d'entiers
     * @param point IN : taille de la liste
     *
     * @return L'indice du maximum de listeJ
     */

    public static void afficherClassementFinal(int[] liste){
        String[] joueurs = new String [49];
        for (int i = 0 ; i < liste.length ; i ++){
            joueurs[i] = Integer.toString(i+1);
        }
        if(liste.length != joueurs.length){
              System.out.println("erreur, le nombre de joueurs ne
correspond pas au nombre de scores enregistrés");
        }
        if(liste.length < 3){
               System.out.println("Erreur, il n'y a pas assez de
joueurs pour effectuer ce calcul.");
        }
        trierJoueurs(liste,joueurs);
        System.out.println("Classement :");
        for (int i = 0 ; i < liste.length ; i ++){
            System.out.println((i+1) + " : " + joueurs[i] + " ");
        }
        System.out.println("Podium :");
        String premier = "";
        String deuxieme = "";
        String troisieme = "";
        if (liste[0] == liste[1] && liste[1] == liste[2]){
            premier += "Premiers : " + joueurs[0];
            premier += " , ";
            premier += joueurs[1];
```

```
            premier += " , ";
            premier += joueurs[2];
        } else if (liste[0] == liste[1]){
            premier += "Premiers : " + joueurs[0];
            premier += " , ";
            premier += joueurs[1];
            troisieme = joueurs[2];
        } else if (liste[1] == liste[2]){
            premier += "Premier : " + joueurs[0];
            deuxieme += "Deuxièmes : " + joueurs[1];
            deuxieme += " , ";
            deuxieme += joueurs[2];
        } else {
            premier = "Premier : " + joueurs[0];
            deuxieme = "Deuxième : " + joueurs[1];
            troisieme = "Troisième(s) : ";
            for (int i = 0 ; i < liste.length ; i ++){
                if (liste[i] == liste[2]) {
                    troisieme += joueurs[i] + " ";
                }
            }
        }
           System.out.println(premier + "\n" + deuxieme + "\n" +
troisieme);
    }
}
```