Pour remettre votre travail :

* Créez votre document de remise qui servira à la correction à partir de celui-ci. Changez son nom pour

**<votre nom de famille>\_<votre DA>\_TP3.docx)**

**Exemple :**

**Tremblay\_1324567\_TP3.docx**

* Placez le document Word ET le répertoire de projet NetBeans dans un répertoire que vous nommerez **TP3** (sans votre nom ni votre DA puisqu’il il sera identifié automatiquement par Léa avec votre nom et votre DA.
* À partir de ce répertoire, créez un document archive (.zip) qui se nommera **TP3.zip**
* Envoyez le document ainsi créé à votre prof **par Léa**

# Barème de correction

Ne pas modifier cette section.

Voir les commentaires en marge du texte pour plus d’explications.

## Résultat final : /15

## Classe Equipe /2

* Encapsulation réalisée correctement
* Constructeurs réalisés tel que demandé
* Méthodes demandées bien faites
* Respect des consignes

## Classe Tournoi /3

* Encapsulation réalisée correctement
* Constructeurs réalisés tel que demandé
* Méthodes demandées bien faites
* Respect des consignes

## Programme de gestion des tournois /4

* Utilisation judicieuse des classes Tournoi, Equipe et Match
* Clarté, simplicité
* Découpage bien fait
* Respect des consignes
* Respect des normes

## Fonctionnement du programme et respect des consignes /6

* Option « Nouveau tournoi »
  + à partir du clavier
  + à partir d’un fichier texte
* Option « Lire tournoi »
* Option « Nouveau match »
* Option « Rapport matchs »
* Option « Afficher classement »
  + Algorithme de tri choisi bien appliqué
* Option « Sauvegarder »
* Option « Quitter »

## Code de la classe Equipe

/\*\*

\* Classe représentant des équipes de hockey

\*

\* @author Sharomova Olga

\*/

public class Equipe {

private String nom;

private int nbJoues;

private int nbGagnes;

private int nbPerdus;

private int nbNuls;

private int nbPoints;

/\*\*

\* Constructeur paramétrique pour créer des équipes à partir de toutes les

\* informations

\*

\* @param nom le nom d'une équipe

\* @param nbJoues le nombre de match joues

\* @param nbGagnes le nombre de match gagner

\* @param nbPerdus le nombre de match perdues

\* @param nbNuls le nombre de match nuls

\* @param nbPoints le nombre points

\*/

public Equipe(String nom, int nbJoues, int nbGagnes, int nbPerdus,

int nbNuls, int nbPoints) {

setNom(nom);

setNbJoues(nbJoues);

setNbGagnes(nbGagnes);

setNbPerdus(nbPerdus);

setNbNuls(nbNuls);

setNbPoints(nbPoints);

}

/\*\*

\*

\* @return le nom d'une équipe

\*/

public String getNom() {

return nom;

}

/\*\*

\*

\* @return le nombre de match joues

\*/

public int getNbJoues() {

return nbJoues;

}

/\*\*

\*

\* @return le nombre de match gagner

\*/

public int getNbGagnes() {

return nbGagnes;

}

/\*\*

\*

\* @return le nombre de match perdues

\*/

public int getNbPerdus() {

return nbPerdus;

}

/\*\*

\*

\* @return le nombre de match nuls

\*/

public int getNbNuls() {

return nbNuls;

}

/\*\*

\*

\* @return le nombre points

\*/

public int getNbPoints() {

return nbPoints;

}

/\*\*

\*

\* @param nom le nom d'une équipe

\*/

private void setNom(String nom) {

this.nom = nom;

}

/\*\*

\*

\* @param nbJoues le nombre de match joues

\*/

private void setNbJoues(int nbJoues) {

this.nbJoues = nbJoues;

}

/\*\*

\*

\* @param nbGagnes le nombre de match gagner

\*/

private void setNbGagnes(int nbGagnes) {

this.nbGagnes = nbGagnes;

}

/\*\*

\*

\* @param nbPerdus le nombre de match perdues

\*/

private void setNbPerdus(int nbPerdus) {

this.nbPerdus = nbPerdus;

}

/\*\*

\*

\* @param nbNuls le nombre de match nuls

\*/

private void setNbNuls(int nbNuls) {

this.nbNuls = nbNuls;

}

/\*\*

\*

\* @param nbPoints le nombre des points

\*/

private void setNbPoints(int nbPoints) {

this.nbPoints = nbPoints;

}

/\*\*

\* Méthode qui ajoute des points gagnes

\*/

public void ajouterMatcheGagne() {

int nb = 2;

nbGagnes = nbGagnes + 1;

calculerPoints(nb);

}

/\*\*

\* Méthode qui ajoute des points nols

\*/

public void ajouterMatcheNul() {

int nb = 1;

nbNuls = nbNuls + 1;

calculerPoints(nb);

}

/\*\*

\* Méthode qui ajoute des points perdus

\*/

public void ajouterMatchePerdu() {

nbPerdus = nbPerdus + 1;

nbJoues = nbJoues + 1;

}

/\*\*

\* Méthode qui ajoute calcule des points

\*

\* @param nb nombre des points

\*/

private void calculerPoints(int nb) {

nbPoints = nbPoints + nb;

nbJoues = nbJoues + 1;

}

}

## Code de la classe Tournoi

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import javax.swing.JOptionPane;

/\*\*

\* Classe représentant un tournoi de hockey

\*

\* @author Sharomova Olga

\*/

public class Tournoi {

/\*\*

\* nombre minimale d'équipes

\*/

public final int MIN\_NOMBRE\_EQUIPES = 8;

/\*\*

\* nombre maximale d'équipes

\*/

public final int MAX\_NOMBRE\_EQUIPES = 50;

private Equipe[][] classement;//tableau d'équipes

private Match[][] matchs;//tableau de match

private int[] nbMatchGroupe;//tableau de nombre de match

/\*\*

\* Constructeur paramétrique pour créer un tournoi en memoir

\*

\* @param nom des équipes

\* @param nbGroupes nombre de groupes

\*/

public Tournoi(String[] nom, int nbGroupes) {

int numero = 0;//compteur

classement = new Equipe[nbGroupes][];

if (nom.length <= MAX\_NOMBRE\_EQUIPES

&& nom.length >= MIN\_NOMBRE\_EQUIPES) {

int reste = nom.length % nbGroupes;

if (reste == 0) {

for (int i = 0; i < nbGroupes; i++) {

classement[i] = new Equipe[nom.length / nbGroupes];

}

} else {

for (int i = 0; i < reste; i++) {

classement[i] = new Equipe[nom.length / nbGroupes + 1];

}

for (int i = reste; i < nbGroupes; i++) {

classement[i] = new Equipe[nom.length / nbGroupes];

}

for (int i = 0; i < classement.length; i++) {

for (int j = 0; j < classement[i].length; j++) {

classement[i][j] =

new Equipe(nom[numero], 0, 0, 0, 0, 0);

++numero;

}

}

matchs = new Match[nbGroupes][];

for (int i = 0; i < matchs.length; i++) {

matchs[i] = new Match[(classement[i].length

\* (classement[i].length - 1)) / 2];

for (int j = 0; j < matchs[i].length; j++) {

matchs[i][j] = new Match();

nbMatchGroupe = new int[j];

}

}

}

}

}

/\*\*

\* Constructeur paramétrique pour créer un tournoi à partir d'un fichier

\* data

\*

\* @param nomFichier nom du fichie data

\* @throws IOException

\*/

public Tournoi(String nomFichier) throws IOException {

int nbGroup = 0;//nombre du groupes

int nbEquip = 0;//nombre des équipes

int joues = 0;//nombre de joues

int gagnes = 0;//nombre du match gagner

int perdus = 0;//nombre du match perdues

int nuls = 0;//nombre du match nuls

int points = 0;//nombre des points

int nbMatch = 0;//nombre du match - linge

int nbMatchDate = 0;//nombre du match - colonne

int butsHote = 0;//nombre des buts de hote

int butsVisiteuse = 0;//nombre des buts de visiteuse

String nom = null;//nom d'équipe

String nomHote = null;//nom d'équipe hote

String nomVisiteuse = null;//nom d'équipe visiteuse

String dateMatch = null;//date du match

DataInputStream fichierBinaire = null;//fichier binaire

try {

fichierBinaire = new DataInputStream(new FileInputStream(nomFichier));

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Erreur d'ouverture"

+ " du fichier",

"Lectur un match ",

JOptionPane.ERROR\_MESSAGE);

}

nbGroup = fichierBinaire.readInt();

classement = new Equipe[nbGroup][];

for (int i = 0; i < nbGroup; i++) {

nbEquip = fichierBinaire.readInt();

classement[i] = new Equipe[nbEquip];

for (int j = 0; j < nbEquip; j++) {

nom = fichierBinaire.readUTF();

joues = fichierBinaire.readInt();

gagnes = fichierBinaire.readInt();

perdus = fichierBinaire.readInt();

nuls = fichierBinaire.readInt();

points = fichierBinaire.readInt();

classement[i][j] = new Equipe(nom, joues, gagnes,

perdus, nuls, points);

}

}

matchs = new Match[nbMatch][];

for (int i = 0; i < nbMatch; i++) {

nbMatch = fichierBinaire.readInt();

matchs[i] = new Match[nbMatch];

nbMatchDate = fichierBinaire.readInt();

for (int j = 0; j < nbMatchDate; j++) {

nomHote = fichierBinaire.readUTF();

nomVisiteuse = fichierBinaire.readUTF();

butsHote = fichierBinaire.readInt();

butsVisiteuse = fichierBinaire.readInt();

dateMatch = fichierBinaire.readUTF();

matchs[i][j] = new Match(nomHote, nomVisiteuse,

butsHote, butsVisiteuse, dateMatch);

}

for (int j = nbMatchDate; j < matchs[i].length; j++) {

matchs[i][j] = new Match();

}

}

}

/\*\*

\* Methode publique écrit tous les données du tournoi dans le fichier data

\*

\* @param nomFichier nom du fichier

\* @throws IOException

\*/

public void sauvegarder(String nomFichier) throws IOException {

DataOutputStream fichierBinaire = null;//fichier

try {

fichierBinaire = new DataOutputStream(

new FileOutputStream(nomFichier));

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Erreur d'ouverture"

+ " du fichier",

"Lectur un match ",

JOptionPane.ERROR\_MESSAGE);

}

fichierBinaire.writeInt(classement.length);

for (int i = 0; i < classement.length; i++) {

fichierBinaire.writeInt(classement[i].length);

for (int j = 0; j < classement[i].length; j++) {

fichierBinaire.writeUTF(classement[i][j].getNom());

fichierBinaire.writeInt(classement[i][j].getNbJoues());

fichierBinaire.writeInt(classement[i][j].getNbGagnes());

fichierBinaire.writeInt(classement[i][j].getNbPerdus());

fichierBinaire.writeInt(classement[i][j].getNbNuls());

fichierBinaire.writeInt(classement[i][j].getNbPoints());

}

}

for (int i = 0; i < matchs.length; i++) {

fichierBinaire.writeInt(matchs[i].length);

fichierBinaire.writeInt(nbMatchGroupe[i]);

for (int j = 0; j < classement[i].length; j++) {

fichierBinaire.writeUTF(matchs[i][j].getEquipeHote());

fichierBinaire.writeUTF(matchs[i][j].getEquipeVisiteuse());

fichierBinaire.writeInt(matchs[i][j].getNbButsHote());

fichierBinaire.writeInt(matchs[i][j].getNbButsVisiteuse());

fichierBinaire.writeUTF(matchs[i][j].getDateMatch());

}

for (int j = nbMatchGroupe[i]; j < matchs[i].length; j++) {

matchs[i][j] = new Match();

}

}

}

/\*\*

\*

\* @param noGroupes numero d'un groupe

\* @return nombre d'éuipes dans ce groupe

\*/

public int getNbEquipes(int noGroupes) {

return classement[noGroupes].length;

}

/\*\*

\*

\* @return nombre de groupes dans un tournoi

\*/

public int getNbGroupes() {

return classement.length;

}

/\*\*

\*

\* @param noGroupes numero d'un groupe

\* @return les noms des éuipes dans ce groupe

\*/

public String[] getNomsEquipes(int noGroupes) {

String tabNom[] = new String[classement[noGroupes - 1].length];

for (int i = 0; i < tabNom.length; i++) {

tabNom[i] = classement[noGroupes - 1][i].getNom();

}

return tabNom;

}

/\*\*

\* Methode qui ajoute un matche

\*

\* @param numeroGr numero du groupe

\* @param nomsHote nom d'éuipe hote

\* @param nomsVisiteuses nom d'éuipe visiteuse

\* @param nbButHote nombre buts de hote

\* @param nbButVis nombre buts de visiteuse

\* @param dateMatche date du match

\*/

public void ajouterMatche(int numeroGr, String nomsHote,

String nomsVisiteuses, int nbButHote, int nbButVis,

String dateMatche) {

int numeroHote = 0;//numero équipe de hote

int numeroVis = 0;//numero équipe de visiteuse

boolean stop = false;//pour ajouter un match

for (int j = 0; j < matchs[numeroGr - 1].length; j++) {

if (stop == false) {

if (matchs[numeroGr - 1][j].getDateMatch().equals("")) {

matchs[numeroGr - 1][j].setMatch(nomsHote, nomsVisiteuses,

nbButHote, nbButVis, dateMatche);

stop = true;

}

}

}

for (int j = 0; j < classement[numeroGr - 1].length; j++) {

if (nomsHote.equals(classement[numeroGr - 1][j].getNom())) {

numeroHote = j;

}

if (nomsVisiteuses.equals(classement[numeroGr - 1][j].getNom())) {

numeroVis = j;

}

}

if (nbButHote > nbButVis) {

classement[numeroGr - 1][numeroHote].ajouterMatcheGagne();

classement[numeroGr - 1][numeroVis].ajouterMatchePerdu();

}

if (nbButHote < nbButVis) {

classement[numeroGr - 1][numeroHote].ajouterMatchePerdu();

classement[numeroGr - 1][numeroVis].ajouterMatcheGagne();

}

if (nbButHote == 0) {

classement[numeroGr - 1][numeroHote].ajouterMatcheNul();

}

if (nbButVis == 0) {

classement[numeroGr - 1][numeroVis].ajouterMatcheNul();

}

}

/\*\*

\* Methode qui trie à bulles le tablaeu classement

\*

\* @param classement tableau d'équipes

\*/

private void trierBullesDecroissant(Equipe[] classement) {

Equipe tampon;

boolean permut;

int longueur = classement.length;

do {

// hypothèse : le tableau est trié

permut = false;

for (int i = 0; i < longueur - 1; i++) {

// Teste si 2 éléments successifs sont dans le bon ordre ou non

if (classement[i].getNbPoints() < classement[i + 1].getNbPoints()) {

// s'ils ne le sont pas, on échange leurs positions

tampon = classement[i];

classement[i] = classement[i + 1];

classement[i + 1] = tampon;

permut = true;

}

}

} while (permut);

}

/\*\*

\*

\* @param noGroupe numero du groupe

\* @return tableau d'équipe correspondant au groupe

\*/

public Equipe[] getClassement(int noGroupe) {

trierBullesDecroissant(classement[noGroupe]);

return classement[noGroupe];

}

/\*\*

\*

\* @param noGroupe numero du groupe

\* @return tableau des matchs correspondant au groupe

\*/

public Match[] getMatchs(int noGroupe) {

return matchs[noGroupe];

}

}

## Code du programme de gestion des tournois

import java.awt.Color;

import java.awt.Font;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

import javax.swing.JOptionPane;

import javax.swing.JTextArea;

import javax.swing.border.Border;

/\*\*

\* TP3 Gestion de tournois de hockey

\*

\* Programme créera le ou les groupes d'équipes inscrites aux différents

\* tournois à partir de données de base fournies dans un fichier texte contenant

\* les données pour chaque tournoi.

\*

\* Date: 07 mai 2014

\*

\* @author Olga Sharomova

\*/

public class TP3 {

/\*\*

\* numero du bouton 1 por créer un tournoi

\*/

private static final int NOUV\_TOUR = 0;

/\*\*

\* numero du bouton 1 pour lire les données dans une fichier data

\*/

private static final int LIRE\_TOUR = 1;

/\*\*

\* numero du bouton 2 pour créer nouveau match

\*/

private static final int NOUV\_MATCH = 2;

/\*\*

\* numero du bouton 3 pour écrir dans une fichier txt

\*/

private static final int RAPPORT = 3;

/\*\*

\* numero du bouton 4 pour afficher les group

\*/

private static final int AFFICHER = 4;

/\*\*

\* numero du bouton 5 pour fermer la fichier

\*/

private static final int FERM\_FICHIER = 5;

/\*\*

\* numero du bouton 6 pour quitter le programme

\*/

private static final int SORTIR\_PROG = 6;

/\*\*

\* nom du fichier txt à écrir

\*/

private static final String NOM\_FICHIER = "ListeMatchs.txt";

/\*\*

\* nom du fichier txt à lire

\*/

private static final String NOM\_FICHIER\_LIRE = "clubs.txt";

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) throws IOException {

Tournoi unTournoi = null;//inicialiser le classe tournoi

String nomFichier = "";//nom fichier data

int numero = 0;//numero du group choisi

boolean exist = false;//pour verifier le fichier

int choixMenu;//choix menu des boutons

do {

// les textes figurant sur les boutons

String textesDesBoutons[] = {

"Nouveau tournoi", // indice 0

"Lire tournoi", // indice 1

"Nouveau match", // indice 2

"Rapport matchs", // indice 3

"Afficher classement", // indice 4

"Fermer fichier tournoi", // indice 5

"Quitter" // indice 6

};

//image pour l'icon

ImageIcon img = new ImageIcon("hockeyplayer.png");

choixMenu = // indice du bouton qui a été cliqué ou CLOSED\_OPTION

JOptionPane.showOptionDialog(null,

"Choisissez l'option voulue",

"Tournois de hockey",

0,

JOptionPane.PLAIN\_MESSAGE,

img, // l’image voulue

textesDesBoutons, // les textes des boutons

textesDesBoutons[1]); // le bouton par défaut

switch (choixMenu) {

case NOUV\_TOUR:

if (unTournoi == null) {

unTournoi = nouveauTournoi(unTournoi);

} else {

JOptionPane.showMessageDialog(null, "Un tournoi est"

+ " charge, veuiller le fermer avant de faire "

+ "cette action",

"Lecture d'un tournoi",

JOptionPane.PLAIN\_MESSAGE);

}

break;

case LIRE\_TOUR:

if (unTournoi == null) {

unTournoi = lireTournoi(unTournoi, nomFichier);

} else {

JOptionPane.showMessageDialog(null, "Un tournoi est"

+ " charge, veuiller le fermer avant de faire"

+ " cette action",

"Lecture d'un tournoi",

JOptionPane.PLAIN\_MESSAGE);

}

break;

case NOUV\_MATCH:

if (unTournoi != null) {

nouveauMatch(unTournoi, numero);

} else {

JOptionPane.showMessageDialog(null,

"Aucun tournoi charge",

"Entrer un match", JOptionPane.PLAIN\_MESSAGE);

}

break;

case RAPPORT:

if (unTournoi != null) {

rapportMatchs(unTournoi);

} else {

JOptionPane.showMessageDialog(null,

"Aucun tournoi charge",

"Entrer un match", JOptionPane.PLAIN\_MESSAGE);

}

break;

case AFFICHER:

if (unTournoi != null) {

afficherClassement(unTournoi);

} else {

JOptionPane.showMessageDialog(null,

"Aucun tournoi charge",

"Entrer un match", JOptionPane.PLAIN\_MESSAGE);

}

break;

case FERM\_FICHIER:

if (unTournoi != null) {

nomFichier = fermerFichier(nomFichier,

unTournoi, exist);

exist = true;

} else {

JOptionPane.showMessageDialog(null,

"Aucun tournoi charge",

"Entrer un match", JOptionPane.PLAIN\_MESSAGE);

}

break;

case SORTIR\_PROG:

quitter(nomFichier, unTournoi, exist);

}

} while (choixMenu != SORTIR\_PROG);

}

/\*\*

\* Méthode qui créer un nouveau à partir un fichier txt

\*

\* @param unTournoi information sur les données de la classe tournoi

\* @return l'information de unTournoi

\* @throws IOException

\*/

private static Tournoi nouveauTournoi(Tournoi unTournoi)

throws IOException {

BufferedReader lecteurFichier = null;//pour lire un fichier txt

String[] nom = null;//un tableau des noms

int nbEquipes = 0;//nombre équipes

int nbGroupes = 0;//nombre groupes

String txt = null;//pour entrer nom du fichier

String ligne = null;//pour lire dans un fichier

int nb = 0;//compteur

txt = JOptionPane.showInputDialog(null,

"Entrez le nom du fichier contenant les noms des équipes ",

"Lectur des noms des équipes",

JOptionPane.PLAIN\_MESSAGE);

if (txt != null) {

if (!txt.equals(NOM\_FICHIER\_LIRE)) {

JOptionPane.showMessageDialog(null, "Fichier \"" + txt

+ "\" inexistant",

"Message d'erreur", JOptionPane.ERROR\_MESSAGE);

} else {

try {

lecteurFichier = new BufferedReader(new FileReader(txt));

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Erreur d'ouverture"

+ " du fichier",

"Lectur un match ",

JOptionPane.ERROR\_MESSAGE);

}

nbEquipes = Integer.parseInt(lecteurFichier.readLine());

nbGroupes = Integer.parseInt(lecteurFichier.readLine());

nom = new String[nbEquipes];

while ((ligne = lecteurFichier.readLine()) != null) {

nom[nb] = ligne;

++nb;

}

unTournoi = new Tournoi(nom, nbGroupes);

}

}

return unTournoi;

}

/\*\*

\* Méthode qui permet de lire un tournoi- dans un fichier data

\*

\* @param unTournoi information sur les données de la classe tournoi

\* @return l'information de unTournoi

\* @throws IOException

\*/

private static Tournoi lireTournoi(Tournoi unTournoi, String nomFichier)

throws IOException {

String nomFichierData = null;//nom du fichier data

nomFichierData = JOptionPane.showInputDialog(null,

"Entrez le nom du fichier tournoi ",

"Lectur d'un tournoi",

JOptionPane.PLAIN\_MESSAGE);

if (nomFichierData == null) {

JOptionPane.showMessageDialog(null, "Fichier \""

+ nomFichierData

+ "\" inexistant",

"Message d'erreur", JOptionPane.ERROR\_MESSAGE);

} else {

unTournoi = new Tournoi(nomFichier);

}

return unTournoi;

}

/\*\*

\* Méthode qui permet d'enter les données d'un nouveau match pour un groupe

\*

\* @param unTournoi information sur les données de la classe tournoi

\* @param numero numero du group choisi

\* @return numero du group choisi

\*/

private static int nouveauMatch(Tournoi unTournoi, int numero) {

String date = "";//pour entrer une date

String equipeHot = "";// nom équipe hote

String equipeVis = "";//nom équipe visiteuse

String butsHote = "";//nombre buts de hote

String butsVis = "";//nombre buts de visiteuse

String nomHot = "hote";//titre hote

String nomVis = "visiteuse";//titre visiteus

String txt = JOptionPane.showInputDialog(null,

"Entrez le numero de groupe 1 a " + unTournoi.getNbGroupes(),

"Nouveau match",

JOptionPane.PLAIN\_MESSAGE);

if (txt == null) {

JOptionPane.showMessageDialog(null, "Le match n'a pas été ajouté",

"Entrer un match ", JOptionPane.PLAIN\_MESSAGE);

} else {

numero = Integer.parseInt(txt);

if (numero > unTournoi.getNbGroupes() || numero < 0) {

JOptionPane.showMessageDialog(null, "Le numero de groupe"

+ " inexistant",

"Message d'erreur", JOptionPane.ERROR\_MESSAGE);

} else {

equipeHot = demenderEquipe(unTournoi, numero, nomHot);

if (equipeHot.equals("")) {

JOptionPane.showMessageDialog(null, "Le match n'a pas "

+ "été ajouté",

"Entrer un match ", JOptionPane.PLAIN\_MESSAGE);

} else {

do {

equipeVis = demenderEquipe(unTournoi, numero, nomVis);

} while (equipeHot.equals(equipeVis)

&& equipeVis.equals(""));

if (equipeVis.equals("")) {

JOptionPane.showMessageDialog(null, "Le match n'a pas"

+ " été ajouté",

"Entrer un match ", JOptionPane.PLAIN\_MESSAGE);

} else {

butsHote = demenderButs(nomHot);

if (butsHote == null) {

JOptionPane.showMessageDialog(null, "Le match n'a "

+ "pas été ajouté",

"Entrer un match ",

JOptionPane.PLAIN\_MESSAGE);

} else {

butsVis = demenderButs(nomVis);

if (butsVis == null) {

JOptionPane.showMessageDialog(null, "Le match"

+ " n'a pas été ajouté",

"Entrer un match ",

JOptionPane.PLAIN\_MESSAGE);

} else {

date = JOptionPane.showInputDialog(null,

"Entrez la date en format jj/mm/aaaa ",

"Nouveau match",

JOptionPane.PLAIN\_MESSAGE);

if (date == null) {

JOptionPane.showMessageDialog(null, "Le "

+ "match n'a pas été ajouté",

"Entrer un match ",

JOptionPane.PLAIN\_MESSAGE);

} else {

try {

unTournoi.ajouterMatche(numero,

equipeHot, equipeVis,

Integer.parseInt(butsHote),

Integer.parseInt(butsVis),

date);

JOptionPane.showMessageDialog(null,

"Le match a été ajouté",

"Entrer un match ",

JOptionPane.PLAIN\_MESSAGE);

} catch (IllegalArgumentException e) {

JOptionPane.showMessageDialog(null,

e.getMessage(),

"Message d'erreur",

JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

}

}

}

return numero;

}

/\*\*

\* Méthode qui demende noms équipe hote et nom équipe visiteus

\*

\* @param unTournoi information sur les données de la classe tournoi

\* @param numero numero du group choisi

\* @param nom équipe

\* @return nom équipe choisi

\*/

private static String demenderEquipe(Tournoi unTournoi, int numero,

String nom) {

String[] tabNom = null;//noms équipes

String txt = null;//pour demender numero d'équipe

String affiche = "";//chaine pour afficher des équipe

String equipe = "";//nom équipe trouver

tabNom = unTournoi.getNomsEquipes(numero);

for (int i = 0; i < tabNom.length; i++) {

affiche += i + 1 + ": " + tabNom[i] + "\n ";

}

txt = JOptionPane.showInputDialog(null,

affiche + "Entrez le numero correspondant a l'equipe " + nom

+ "entre 1 et " + tabNom.length,

"Nouveau match",

JOptionPane.PLAIN\_MESSAGE);

if (txt != null) {

if (Integer.parseInt(txt) > tabNom.length

|| Integer.parseInt(txt) == 0) {

JOptionPane.showMessageDialog(null, "Le numero d'equipe"

+ " inexistant",

"Message d'erreur", JOptionPane.ERROR\_MESSAGE);

txt = null;

} else {

for (int i = 0; i < tabNom.length; i++) {

if (Integer.parseInt(txt) == i) {

equipe = tabNom[i - 1];

}

}

}

}

return equipe;

}

/\*\*

\* Méthode qui demende buts équipe hote et buts équipe visiteus

\*

\* @param nom équipe

\* @return numero de but

\*/

private static String demenderButs(String nom) {

String buts = null;//pour demender buts

buts = JOptionPane.showInputDialog(null,

"Entrez le numero de buts de l'equipe " + nom,

"Nouveau match",

JOptionPane.PLAIN\_MESSAGE);

return buts;

}

/\*\*

\* Méthode qui produit une liste des matchs sur un fichier txt

\*

\* @param unTournoi information sur les données de la classe tournoi

\*/

private static void rapportMatchs(Tournoi unTournoi) {

final String[][] entetes = {

{

"Hote", "Visiteur", "Buts", "Buts", "Date du"

},

{

"", "", "hote", "visiteur", "match"

}

};//pou afficher entetes

PrintWriter createurFichier = null;//pour écrir dans un fichier

try {

createurFichier = new PrintWriter(

new BufferedOutputStream(new FileOutputStream(NOM\_FICHIER)));

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "Erreur d'ouverture du fichier",

"Ecrir un match ",

JOptionPane.ERROR\_MESSAGE);

}

createurFichier.println("LISTE DES MATCHS JOUER\n");

createurFichier.println("=================================\n");

for (int i = 0; i < entetes.length; i++) {

for (int j = 0; j < entetes[i].length; j++) {

createurFichier.printf("%25s", entetes[i][j]);

}

createurFichier.println("");

}

for (int j = 0; j < unTournoi.getNbGroupes(); j++) {

Match[] matchs = unTournoi.getMatchs(j);

for (int i = 0; i < matchs.length; i++) {

if (!matchs[i].getDateMatch().equals("")) {

createurFichier.printf("%25s%25s%25d%25d%25s\n",

matchs[i].getEquipeHote(),

matchs[i].getEquipeVisiteuse(),

matchs[i].getNbButsHote(),

matchs[i].getNbButsVisiteuse(),

matchs[i].getDateMatch());

}

}

}

createurFichier.close();

}

/\*\*

\* Méthode qui affiche tous les groupes du tournoi à l'écran

\*

\* @param unTournoi information sur les données de la classe tournoi

\*/

private static void afficherClassement(Tournoi unTournoi) {

String affichage = "";//afficher la chaine avec les groupes

String nom = "Nom";//afficher la chaine avec les groupes

String joue = "Jouees";//afficher la chaine avec les groupes

String perdue = "Perdues";//afficher la chaine avec les groupes

String gagne = "Ganees";//afficher la chaine avec les groupes

String point = "Points";//afficher la chaine avec les groupes

String nulle = "Nulles";//afficher la chaine avec les groupes

JTextArea zoneTexte = new JTextArea(5, 30);

zoneTexte.setFont((new Font("Courier new", Font.BOLD, 12)));

zoneTexte.setBorder(BorderFactory.createLineBorder(Color.BLUE, 3, true));

for (int j = 0; j < unTournoi.getNbGroupes(); j++) {

Equipe[] classement = unTournoi.getClassement(j);

affichage += String.format("Groupe %d\n ===========\n"

+ "%-25s%10s%10s%10s%10s%10s\n",

j + 1, nom, joue, gagne, perdue, nulle, point);

for (int i = 0; i < unTournoi.getNbEquipes(j); i++) {

affichage += String.format("%-25s%10d%10d%10d%10d%10d\n",

classement[i].getNom(),

classement[i].getNbJoues(),

classement[i].getNbGagnes(),

classement[i].getNbPerdus(),

classement[i].getNbNuls(),

classement[i].getNbPoints());

}

}

zoneTexte.setText(affichage);

JOptionPane.showMessageDialog(null, zoneTexte,

"Classement", JOptionPane.PLAIN\_MESSAGE);

}

/\*\*

\* Méthode qui permet de fermer le fichier tournoi ouvert

\*

\* @param nomFichier nom du fichier binaire

\* @param unTournoi information sur les données de la classe tournoi

\* @param exist puor verifier si le fichier exist

\* @return nom du fichier binaire

\* @throws IOException

\*/

private static String fermerFichier(String nomFichier, Tournoi unTournoi,

boolean exist) throws IOException {

String nom = "";//pour entrer nom du fichier

String menu[] = {

"Oui", // indice 0

"Non" // indice 1

};

//afficher un image

ImageIcon img2 = new ImageIcon("hokeyes.png");

int choix = // indice du bouton qui a été cliqué ou CLOSED\_OPTION

JOptionPane.showOptionDialog(null,

"Le tournoi a été modifié, voulez-vous le sauvegarder?",

"Quitter",

0,

JOptionPane.PLAIN\_MESSAGE,

img2, // l’image voulue

menu, // les textes des boutons

menu[0]); // le bouton par défaut

switch (choix) {

case 0:

if (exist == false) {

nomFichier = JOptionPane.showInputDialog(null,

"Entrez le nom du fichier tournoi ",

"Lectur nom du fichier",

JOptionPane.PLAIN\_MESSAGE);

if (nomFichier == null) {

JOptionPane.showMessageDialog(null, "Fichier \""

+ nomFichier + "\" inexistant",

"Message d'erreur", JOptionPane.ERROR\_MESSAGE);

}

} else {

int fichier = // indice du bouton qui a été cliqué ou CLOSED

JOptionPane.showOptionDialog(null,

"Voulez-vous conserver le fichier ("

+ nomFichier + " )?",

"Sauvegarder",

0,

JOptionPane.PLAIN\_MESSAGE,

null,

menu, // les textes des boutons

menu[0]); // le bouton par défaut

switch (fichier) {

case 0:

unTournoi.sauvegarder(nomFichier);

unTournoi = null;

break;

case 1:

nom = JOptionPane.showInputDialog(null,

"Entrez le nom du fichier tournoi ",

"Lectur nom du fichier",

JOptionPane.PLAIN\_MESSAGE);

if (nom != null) {

unTournoi.sauvegarder(nom);

unTournoi = null;

}

}

}

case 1:

}

return nomFichier;

}

/\*\*

\* Méthode qui permet quitter le programme et sauvegarder les données

\*

\* @param nomFichier nom du fichier binaire

\* @param unTournoi information sur les données de la classe tournoi

\* @param exist puor verifier si le fichier exist

\* @throws IOException

\*/

private static void quitter(String nomFichier, Tournoi unTournoi,

boolean exist) throws IOException {

nomFichier = fermerFichier(nomFichier, unTournoi, exist);

}

}