Licence 1 Informatique-Electronique

Année universitaire 2024-2025

Présenté par : **DIALLO Aissatou Bobo**

Chargée du cours : **Mme CULLOT Nadine**

Rapport de projet

P4P – Boîte de dialogue Scores

Table des matières

I. Introduction

Cette boîte de dialogue a pour but de présenter les joueurs de façon visuelle et intuitive. Elle permet d'afficher des informations essentielles comme le pseudo, la photo, et un résumé de leurs résultats sous forme de graphique. L'utilisateur peut facilement naviguer dans la liste et voir les performances de chaque joueur en un clic. L'affichage automatique du score rend l'interface vivante et simple à comprendre. L'ensemble a été pensé pour rendre la consultation des données à la fois rapide et agréable.

II. Cahier des charges

La boîte de dialogue permet de :

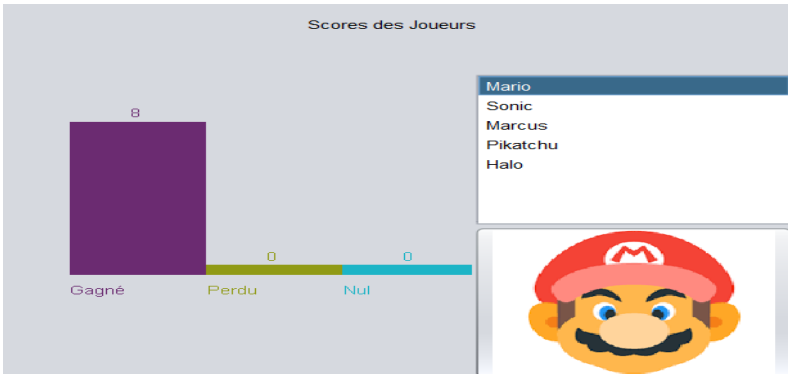
- Sélectionner le nom d'un joueur dans la liste (dans une jList).
- Afficher la photo de chaque joueur sur un bouton de type jButton en cliquant sur son nom dans la liste.
- Montrer également le nombre de parties gagnées, le nombre de parties perdues et le nombre de matchs nuls sous forme d'histogramme (dont chaque rectangle a une couleur differente) sur un panneau de type jPanel.

Dans la prochaine section, je parlerai du déroulement d'une partie de jeu.

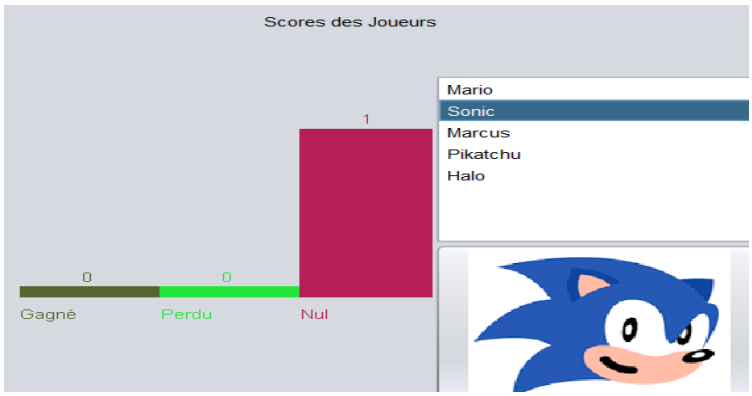
III. Démonstration du fonctionnement de la boîte de dialogue

Pour voir le fonctionnement de la boîte de dialogue on regardera les anciens scores de deux joueurs, puis on fera un match entre les deux pour voir l'évolution de leurs scores. La partie se déroulera entre Mario et Sonic.

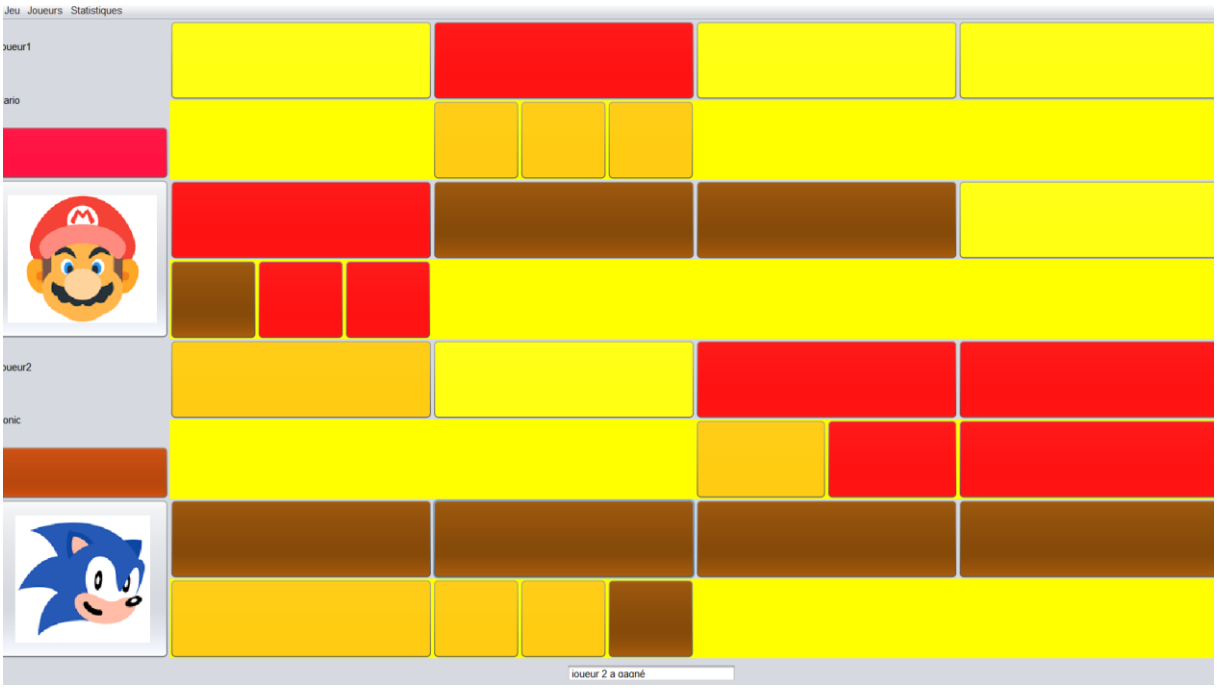
Sur l'image ci-dessous on voit qu'on a cliqué sur le joueur Mario et on voit sa photo ainsi que son score des parties déjà jouées sous forme d'histogramme : son score est 8 il n'a aucune partie perdue, ni un match nul.



Et là on a Sonic qui n'a aucune partie gagnée, aucune perdu et un match nul.

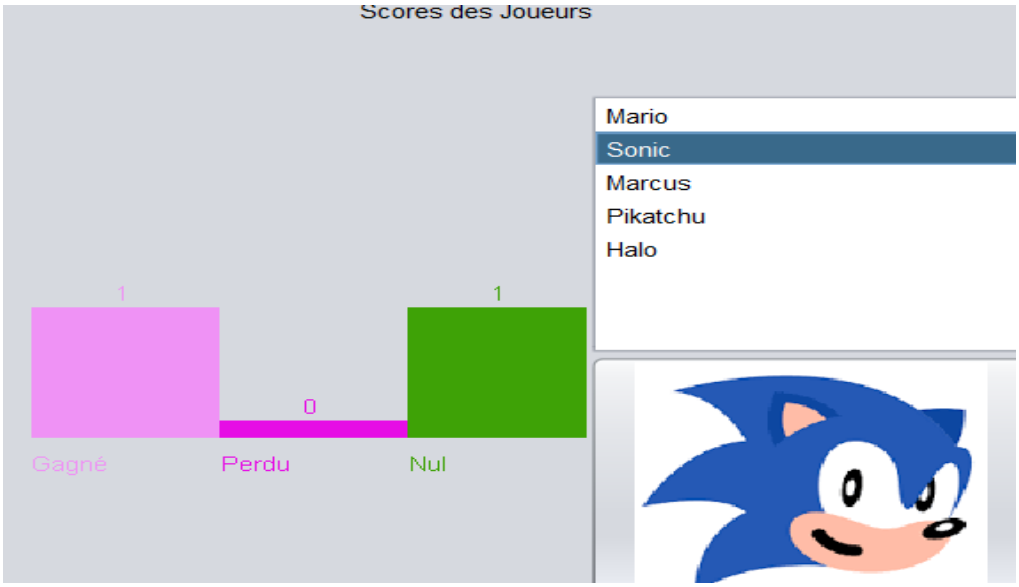


Maintenant on va faire un match entre les deux pour voir comment évolue leurs différents scores.

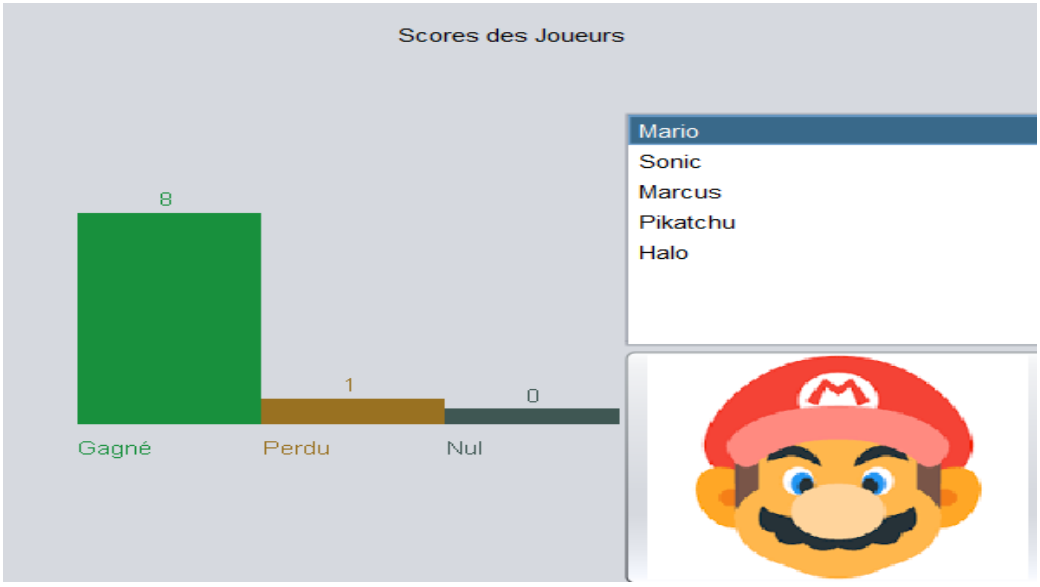


Après le match entre les deux on voit que c'est le joueur 2 qui a gagné c'est-à-dire Sonic.

Donc voilà son score après la mise à jour : une partie gagnée et un match nul.



Et là nous avons celui de Mario : 8 partie gagnées et bien-sûr une partie perdue, et aucun match nul.



IV. Description des informations gérées par la boîte

```
//Attributs
private LesJoueurs lj;
private Joueur j;
```

La boîte de dialogue reçoit un objet de la classe principale :

- private LesJoueurs lj : qui est la liste de tous les joueurs, ici qui nous sert à remplir la liste par les noms de ces joueurs et aussi dans la sélection d'un joueur de cette liste.

  private Joueur j : attribut utilisé par la boîte pour sélectionner un joueur dans la liste, afficher sa photo et représenter ses statistiques sur l'histogramme via les méthodes getNbpartiesGagnees() , getNbpartiesPerdues() , getNbpartiesNul() qui sont de la classe Joueur.

La boîte ne renvoie rien à la classe principale.

V. Structures de données utilisées

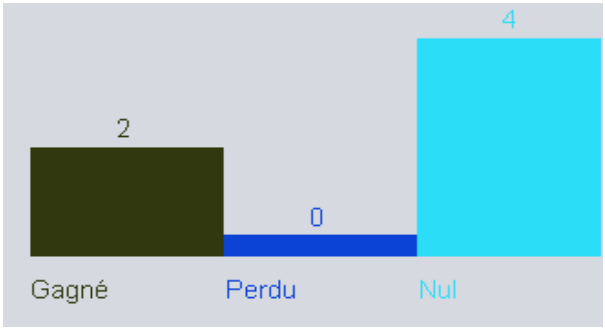
Dans la boîte de dialogue j'ai eu à utiliser deux tableaux au niveau de ma méthode dessineHisto :

- *int[] valeurs = {gagne, perdu, nul};* qui est un tableau d'entiers, gagne, perdu et nul sont des variable de type entier qui stockent le nombre de parties gagnées, perdu et matchs nuls d'un joueur donné.
  J'ai fais le choix de ce tableau pour que chaque variable récupère le nombre de partie lui correspondant (voir image en dessous) .

```
// On récupère les vraies valeurs directement du joueur
int gagne = j.getNbpartiesGagnees();
int perdu = j.getNbpartiesPerdues();
int nul = j.getNbpartiesNul();
```

Après on se sert de ces valeur pour mettre sur les différents rectangles de l'histogramme.(voir l'instruction ci-dessous et l'image que ça donne).

```
// Numéro juste au-dessus de la barre
g.drawString("" + valeurs[i],
        decalageLarg + i * largRect + largRect / 2 - 5,
        hautMax + decalageHaut - h - 5);
```



- *String[] labels = {"Gagné", "Perdu", "Nul"};* qui est aussi un tableau de type chaine de caractère qui décrit ce que représente chaque barre de l'histogramme.

```
int[] valeurs = {gagne, perdu, nul};
String[] labels = {"Gagné", "Perdu", "Nul"};
```

(Voir l'illustration de l'histogramme ci-dessus)

- *DefaultListModel* : utilisé pour afficher dynamiquement les pseudos des joueurs dans une JLIST, on a déclaré une variable de type DefaultListModel on l'a initialisé avec l'appel du constructeur, on applique ce model à notre Liste après on parcourt notre liste tout en récupérant chaque joueur a l'indice i et en l'ajoutant à notre model (qui lui est appliqué à la liste donc on ajoute les pseudo à la liste).
  Une liste a toujours besoin d'un model pour recevoir des donnée c'est pourquoi on fait de cette manière.

```
public void initListesJoueurs() { //methode qui remplie la jlist par le nom des joueurs
    DefaultListModel mod = new DefaultListModel();
    ListJoueurs.setModel(mod);
    for (int i = 0; i < this.lj.getNbJoueurs(); i++) {

        mod.addElement(this.lj.getJoueur(i).getPseudo());
    }

}
```

VI. Constructeur et Méthodes appelées

public ScoresDlg(java.awt.Frame parent, boolean modal, LesJoueurs lj)

Dans le constructeur de la boîte on a :

super(parent, modal);

L'appel du constructeur de la classe parent qui a besoin de deux paramètres parent(qui est la fenêtre qui va gérer la boîte de dialogue) et modale(mode d'ouverture de la boîte) .

initComponents();

Qui fait la création de l'interface par l'IDE

this.lj = lj;

Initialisation de la liste des joueurs(le deuxième lj c'est celui en paramètre on le mets en paramètre parce qu'il contiendra la liste des joueurs qu'on doit récupérer).

this.j = this.lj.getJoueur(0);

On initialise l'attribut de type Joueur en lui donnant le premier joueur de la liste.

initListesJoueurs();

Qui est l'appel de la méthode qui remplie la JLIST par les nom des joueurs

afficheJoueur() ;

Appel de la méthode qui affiche les informations des joueurs sur les composants de l'interface.

PDroit.setPreferredSize(new Dimension(200, 200));

Méthode appelée sur le panneau à droite pour fixer sa dimension pour ne pas que sa taille change en fonction de la photo affichée

VII. Analyse des fonctionnalités

Dans cette boîte de dialogue on a qu'un gestionnaire d'évènement qui gère le clique sur un nom dans la liste des joueur :

Le clic d'un nom dans cette liste conduit à l'affichage de la photo du joueur à qui appartient ce nom sur un bouton et de son score(nombre de parties gagnées, perdues et matchs nuls) dans la partie graphique de l'interface.

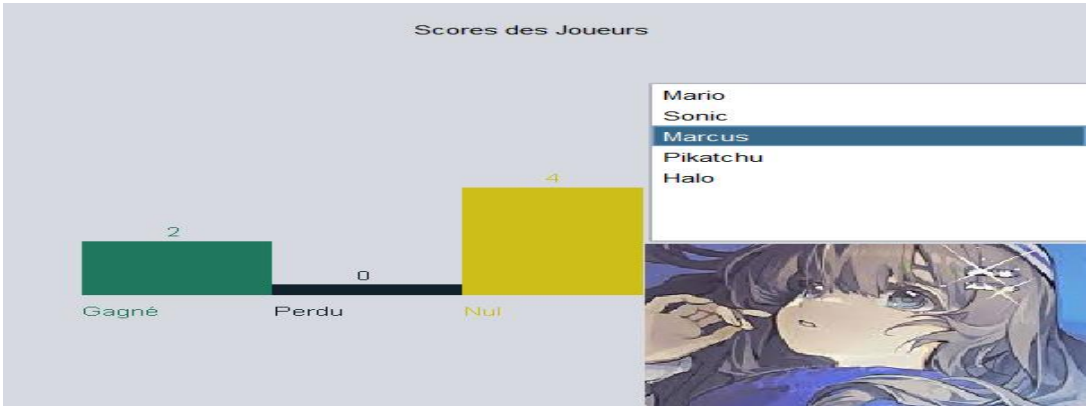
```
private void ListJoueursMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    int ind = ListJoueurs.getSelectedIndex();
    this.j = this.lj.getJoueur(ind);
    afficheJoueur();
    this.repaint();

}
```

Dans cette méthode on récupère l'index de l'élément sélectionné de la JLIST on le stocke dans une variable, après on utilise l'index récupéré pour **obtenir un joueur** à cette position via la méthode getJoueur(int ind) sur l'objet lj(liste des joueurs), et l'assigne à l'attribut j (le joueur actuellement sélectionné).

Et la méthode afficheJoueur() met à jour l'interface en affichant la photo du joueur sélectionné.

La méthode repaint() est appelé sur la boîte pour redessiner l'histogramme sur le panneau en appelant la méthode paint.

Exemple : Après clique sur le joueur Marcus voilà ce que nous affiche l'interface.



VIII. Description de l'affichage graphique

La méthode **dessineHisto**() a pour but de représenter graphiquement, sous forme d'un histogramme, les résultats d'un joueur : le nombre de parties gagnées, perdues et nulles.

```
Graphics g = PCentre.getGraphics();

int largeurMax = this.getWidth() / 2;
int decalageLarg = 50;
int decalageHaut = 50;
int largRect = largeurMax / 3;
int hautMax = this.PCentre.getHeight() / 2;
```

Là au début(sur la première ligne) on récupère l'outil qui va nous aider pour dessiner sur le composant PCentre, cet outil est un objet graphique qui nous permet de dessiner des formes (fillRect,drawArc…) et aussi d'écrire du texte avec drawString.

Après (sur ligne 2 et 6) on prend la moitié de la fenêtre qui est la largeur maximale utilisé pour l'histogramme et aussi la moitié de la hauteur du panneau comme hauteur maximale des barres de l'histogramme.

Sur (ligne 3 et 4) ce sont les marges pour espacer le dessin du bord.

Et sur (ligne 5) largeur d'une seule barre (1/3 de **largeurMax**, car il y a 3 barres). Voir image ci-dessus

```
int max = Math.max(1, gagne + perdu + nul); /

for (int i = 0; i < 3; i++) {
    int h;
    if (valeurs[i] == 0) {
        h = 10;
    } else {
        h = hautMax * valeurs[i] / max;
    }

    int r = (int) (Math.random() * 255);
    int v = (int) (Math.random() * 255);
    int b = (int) (Math.random() * 255);
    g.setColor(new Color(r, v, b));
```

max = Math.max(1, …) évite une division par zéro si toutes les valeurs sont nulles.

Dans la boucle, la hauteur de chaque barre est calculée proportionnellement à la valeur, mais un minimum de 10 est utilisé si la valeur est 0 (pour que la barre reste visible).

Une couleur aléatoire est générée pour chaque barre avec des valeurs RGB entre 0 et 255.

g.setColor(...) applique cette couleur au dessin.(voir image ci-dessus)

```
// Numéro juste au-dessus de la barre
g.drawString("" + valeurs[i],
        decalageLarg + i * largRect + largRect / 2 - 5,
        hautMax + decalageHaut - h - 5);

// Label en dessous
g.drawString(labels[i],
        decalageLarg + i * largRect,
        hautMax + decalageHaut + 20);

// Dessin de la barre
int x = decalageLarg + i * largRect;
int y = hautMax + decalageHaut - h;
g.fillRect(x, y, largRect, h);
```

- Pour la première instruction c'est la méthode qui écrit les numéros au dessus des barres de l'histogramme :

decalageLarg + i * largRect : position horizontale (x) de la barre numéro i.

+ largRect / 2 - 5 : centre le texte au milieu de la barre, avec un petit décalage pour l'alignement.

hautMax + decalageHaut - h - 5 : position verticale (y), juste au-dessus de la barre (on monte un peu avec le -5).

- Pour la deuxième c'est pour le texte en dessous des barres de l'histogramme :

labels[i] : nom de la catégorie correspondant à la barre (ex : "Perdu").

decalageLarg + i * largRect : position horizontale (x) de la barre.

hautMax + decalageHaut + 20 : position verticale (y) en dessous de la barre (on descend un peu avec le +20).

- Pour la troisième c'est elle qui dessine les barres :

x = decalageLarg + i * largRect : position horizontale de la barre numéro i.

y = hautMax + decalageHaut - h : position verticale du haut de la barre, on monte de h pixels.

g.fillRect(x, y, largRect, h) : dessine un rectangle plein (la barre) à partir de (x, y), avec :

largRect : largeur de la barre,

h : hauteur calculée en fonction de la valeur.

```
public void paint(Graphics g) {
    super.paint(g);

    dessineHisto();

}
```

La méthode *dessineHisto*() est appelée dans la méthode *paint(Graphics g),* qui est responsable de dessiner l'interface graphique.

Dans cette méthode *paint*, on commence par appeler la methode *paint* de la classe parente avec *super.paint(g),* ce qui permet de nettoyer et préparer correctement l'affichage.

Ensuite, on appelle *dessineHisto*() pour tracer l'histogramme sur le panneau en utilisant le contexte graphique passé en paramètre (g).

IX. Description de l'appel de cette boîte

```
private void MScoreActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    ScoresDlg diag = new ScoresDlg(this, true, listeJ);
    diag.setVisible(true);
}
```

L'appel de la boîte se fait dans la classe principale c'est-à-dire dans la classe P4P, en faisant le gestionnaire de clique sur le sous menu scores du menu statistique.

Dans cette méthode on crée une variable qui a le nom de la boîte de dialogue comme type on fait appel a son constructeur via le mot new et comme paramètre on le donne this qui est la JFRAME dans laquelle nous somme c'est pourquoi le this après on donne le mode d'ouverture (true = bloquant) puis on lui fourni la liste des joueurs.

Puis on met le setVsible à true pour que la boîte s'ouvre quand on clique sur le sous menu scores de la fenêtre principale.

X. Conclusion

Ce projet m'a permis de mobiliser et d'approfondir un ensemble de compétences techniques et méthodologiques. J'ai renforcé ma maîtrise de Java Swing, notamment dans l'utilisation de composants comme JList, JPanel, JButton et la gestion des événements avec les écouteurs. J'ai aussi appris à intégrer des éléments graphiques dynamiques dans une interface pour représenter visuellement des données de manière claire et lisible. Cette approche m'a permis d'allier précision technique et souci de l'ergonomie dans la présentation des résultats. Ce travail m'a également conduit à approfondir plusieurs notions abordées en cours, en les appliquant à un cas concret et fonctionnel.