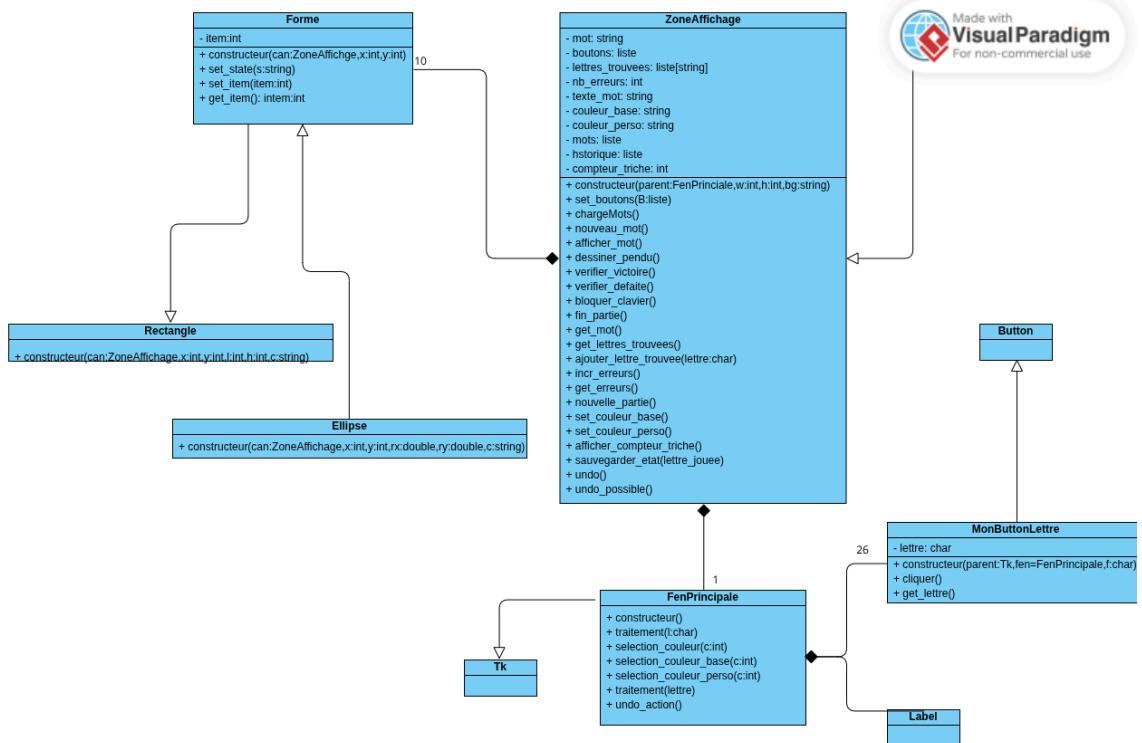


# TD5-INF-TC2 Jeu du pendu "Amélioration du jeu"

Adèle Marcon - groupe C1a - Intervenant: Derrode Lamia

```
In [7]: from IPython.display import Image
Image(url="TD5.png", width=1275, height=1066)
```

Out[7]:



## 1. Ajout d'une méthode de personnalisation des couleurs

Dans ZoneAffichage

Ajout de deux fonctions permettant de changer la couleur de la base (potence) et du bonhomme pendu

```
In [ ]: def set_couleur_base(self, couleur):
    self.__couleur_base = couleur

def set_couleur_perso(self, couleur):
    self.__couleur_perso = couleur
```

## Dans FenPrincipale

Création des 3 boutons permettant la modification de la couleur de l'arrière plan, de la potence et du bonhomme

```
In [ ]: boutonCouleur = Button(barreOutils, text="Couleur fond")
boutonCouleur.pack(side=LEFT, padx=5, pady=5)

boutonCouleurbase = Button(barreOutils, text="Couleur base")
boutonCouleurbase.pack(side=LEFT, padx=5, pady=5)

boutonCouleurpendu = Button(barreOutils, text="Couleur pendu")
boutonCouleurpendu.pack(side=LEFT, padx=5, pady=5)

boutonCouleur.config(command=self.selection_couleur)
boutonCouleurbase.config(command=self.selection_couleur_base)
boutonCouleurpendu.config(command=self.selection_couleur_perso)
```

Création des fonctions permettant à l'utilisateur de choisir sa couleur

```
In [ ]: def selection_couleur(self):
    couleur = colorchooser.askcolor()[1]
    if couleur:
        self.configure(bg=couleur)

def selection_couleur_base(self):
    couleur = colorchooser.askcolor()[1]
    if couleur:
        self.__canevas.set_couleur_base(couleur)

def selection_couleur_perso(self):
    couleur = colorchooser.askcolor()[1]
    if couleur:
        self.__canevas.set_couleur_perso(couleur)
```

## 2. Ajout d'une méthode de triche

### Dans ZoneAffichage

Ajout d'un compteur de triche

```
In [ ]: def afficher_compteur_triche(self):
    compteur = f"Alors comme ça vous trichez ? Je compte: {self.__compte}"

    self.__compteur = self.create_text(600,800, text = compteur, font=(
```

Ajout d'une fonction pour sauvegarder l'état de la partie pour pouvoir revenir en arrière

```
In [ ]: def sauvegarder_etat(self, lettre_jouee):
    # Sauvegarde l'état actuel du jeu dans l'historique
```

```

etat = {
    'lettres_trouvees': copy.deepcopy(self.__lettres_trouvees),
    'erreurs': self.__erreurs,
    'lettre_jouee': lettre_jouee
}
self.__historique.append(etat)

```

## Ajout de la fonction undo pour faire les différentes actions en cas de triche

Mise à jour du compteur, récupération du dernier état grâce à l'historique, mise à jour de l'état actuel pour qu'il corresponde à l'état précédent, mise à jour de l'interface avec l'état actuel (dont une mise à jour du compteur de triche), dessin du pendu et récupération de la lettre annulée pour la dégriser

```
In [ ]: def undo(self):
    # Augmenter le compteur de triche
    self.__compteur_triche += 1

    # Retirer le dernier état
    dernier_etat = self.__historique.pop()

    # Restaurer l'état précédent
    self.__lettres_trouvees = dernier_etat['lettres_trouvees']
    self.__erreurs = dernier_etat['erreurs']

    # Redessiner l'interface
    self.delete("all")
    self.afficher_mot()
    self.afficher_compteur_triche()

    # Redessiner le pendu
    for i in range(1, self.__erreurs + 1):
        temp_erreurs = self.__erreurs
        self.__erreurs = i
        self.dessiner_pendu()
        self.__erreurs = temp_erreurs

    # Retourner la lettre qui a été annulée pour réactiver le bouton lors de l'undo
    return dernier_etat['lettre_jouee']
```

Fonction permettant de vérifier si l'action undo est possible (si un coup a déjà été joué)

```
In [ ]: def undo_possible(self):
    return len(self.__historique)
```

## Dans FenPrincipale

### Ajout du bouton Undo et son action

```
In [ ]: boutonUndo = Button(barreOutils, text="Undo (Triche!)")
boutonUndo.pack(side=LEFT, padx=5, pady=5)

boutonUndo.config(command=self.undo_action)
```

Fonction permettant de vérifier si au moins un coup a déjà été joué + utilise undo pour récupérer la lettre annulée et la dégriser

```
In [ ]: def undo_action(self):
    if not (self.__canevas.undo_possible() > 0):
        print("Impossible de revenir en arrière : aucun coup joué")
        return

    # Effectuer le undo et récupérer la lettre annulée
    lettre_annulee = self.__canevas.undo()

    if lettre_annulee:
        # Réactiver le bouton de la lettre annulée
        index = ord(lettre_annulee) - ord('A')
        self.__B[index].config(state=NORMAL)
        print(f"Coup annulé : lettre '{lettre_annulee}' réactivée")
```