|  |  |
| --- | --- |
| CLASS TOURNOI |  |
| Méthode | Valid\_nb\_tour |
| Input | Demande entrer nb\_tour. 4 par défaut |
| Actions | Test conformité avec if et methode isnumeric, sinon erreur. |
| Output | Si conforme, ajout à \_\_init\_\_ |
| Méthode | Liste\_tour |
| Input | Id\_Tour  Tuple issu de match |
| Actions | Créer une liste au format suivant [Id\_Tour, [tous les tuples issus de Class Match]]  Usage de la methode .append pour ajouter les tuple à chaque tour, et pour ajouter une nouvelle itération de la liste principale à chaque nouveau tour |
| Output | Envoyer la liste de tuple du dernier tour terminé vers l’attribution des points et classement |
| Méthode | Attribution\_points et maj classement |
| Input | Tuple issus de liste\_tour (Class Tournoi) /self.  Classement à actualiser /self. |
| Actions | Pour chaque tuple, pour chaque liste, attribuer point au joueur formant la paire [Joueur, point] |
| Output | Tableau  Envoyer à Class Tour |

|  |  |
| --- | --- |
| CLASS JOUEUR |  |
| Méthode | Valid\_id |
| Input | Demande entrer id Joueur |
| Actions | Test conformité avec if et conditions multiple, sinon erreur. *(à changer par regex)* |
| Output | Si conforme, ajout à \_\_init\_\_ |
| Méthode | Valid\_nom |
| Input | Demande entrer nom Joueur |
| Actions | Test conformité avec if et methode isalpha, sinon erreur. |
| Output | Si conforme, ajout à \_\_init\_\_ |
| Méthode | Valid\_prenom |
| Input | Demande entrer prénom Joueur |
| Actions | Test conformité avec if et methode isalpha, sinon erreur. |
| Output | Si conforme, ajout à \_\_init\_\_ |
| Méthode | Valid\_date\_naissance |
| Input | Demande entrer date naissance Joueur |
| Actions | Test conformité avec if et regex ? (format à énoncer dans l’affichage de l’input), sinon erreur. |
| Output | Si conforme, ajout à \_\_init\_\_ |

|  |  |
| --- | --- |
| CLASS TOUR |  |
| Méthode | Generation\_paire |
| Input | Classement actualisé venant de Class Tournoi  Id\_tour venant de Class Tournoi  Listes de id\_match déjà joués |
| Actions | *Pour id\_Tour = 1*  *Shuffle liste après transformation*  Option 1 :  Tranformation classement en liste de joueur ordonnée à partir du classement  2 boucles for pour tester la liste avec elle-même et créer les paires  Conditions : valeur1 != valeur2 AND id\_match\_test not in id\_match\_joue  Si ok, paire créer et valeur .pop de la liste  *Problème de décalage dans la boucle avec le pop…*  Option 2 :  Transformation d’une liste de liste à 2 éléments à partir du classement [Joueur, flag]. Le flag est un indicateur pour indiquer si le joueur à un paire attribuer ou non. 0 le joueur est dispo, 1 le joueur est déjà attribué  2 boucles for pour tester la liste avec elle-même et créer les paires  Conditions : valeur1 != valeur2 AND id\_match\_test not in id\_match\_joue AND flag == 0  Si ok, paire créée et flag = 1 |
| Output | Paires à envoyer à Class Match |
| Méthode | Id\_match\_joue *(problème, méthode qui dépasse le simple tour)* |
| Input | Paire issu du tour |
| Actions | Concatener Paire[0] + Paire[1] et Paire[1] + Paire[0], afin de créer des id\_match interdit  Et .append dans une liste qui sera envoyé pour la génération\_des\_paires |
| Output | Envoyer à Class Tour |

|  |  |
| --- | --- |
| CLASS MATCH |  |
| Méthode | attribution\_couleur |
| Input | paire de joueur : Liste [J1, J2] |
| Actions | Attribution random + if |
| Output | print Joueur + couleurs |
| Méthode | Input\_score |
| Input | paire de joueur : Liste [J1, J2]  demande entrer point obtenu *(à changer en gagné, perdu, égalité)* |
| Actions | Vérification si point ok, sinon erreur |
| Output | Tuple ([J1, pointJ1], [J2, pointJ2])  Envoyer à Class Match, dans compléter liste\_tour |