UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Projet de session - TP2

Jacques Berger

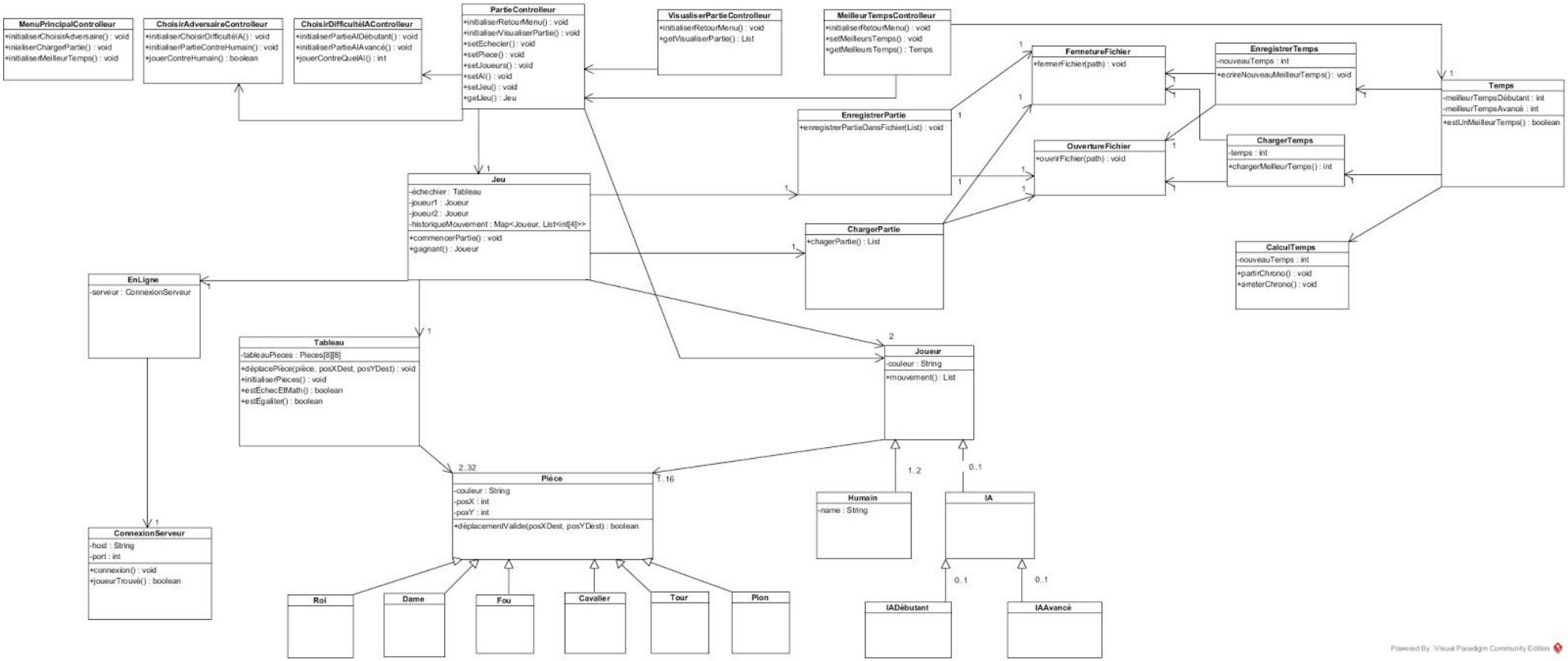
DANS LE CADRE DU COURS
Génie Logiciel: conception
INF5153
Groupe: 40

PAR
Jean-Michel Poirier (POIJ26089200)
Audrey Eugene (EUGA21589707)

9 novembre 2017

Diagramme de classes

Le diagramme de classe illustrant la conception détaillé d'un jeu d'échec. Un utilisateur peut jouer une partie d'échec contre un joueur humain en établissant une connexion à un serveur ou jouer contre un joueur artificiel localement. Le diagramme est composé des classes des couches: application, du domaine, services techniques et fondation.

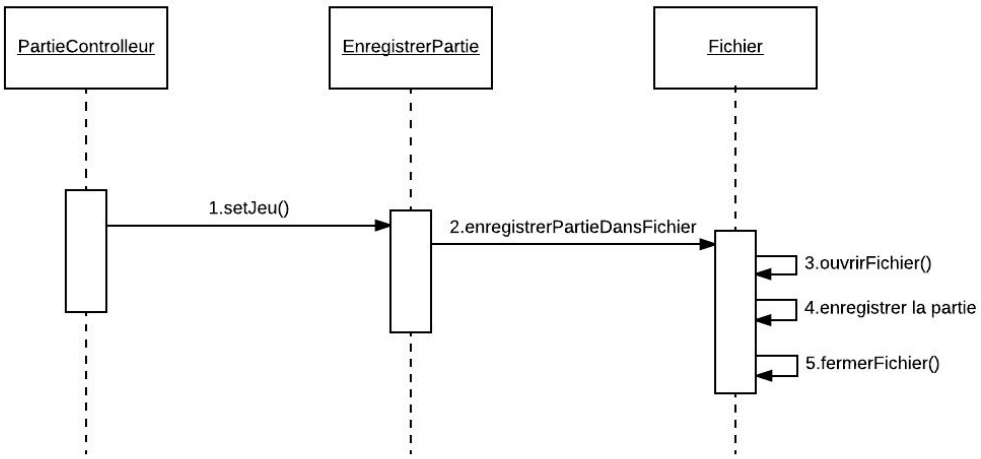

MenuPrincipalController
+initialiserChoisirAdversaire() : void
+inialiserChargerPartie() : void
+initialiserMeilleurTemps() : void
ChoisirAdversaireController
+initialiserChoisirDifficultéIA() : void
+initialiserPartieContreHumain() : void
+jouerContreHumain() : boolean
ChoisirDifficultéIAController
+initialiserPartieAIDébutant() : void
+initialiserPartieAIAvancé() : void
+jouerContreQuelAI() : int
PartieController
+initialiserRetourMenu() : void
+initialiserVisualiserPartie() : void
+setEchecier() : void
+setPiece() : void
+setJoueurs() : void
+setAI() : void
+setJeu() : void
+getJeu() : Jeu
VisualiserPartieController
+initialiserRetourMenu() : void
+getVisualiserPartie() : List
MeilleurTempsController
+initialiserRetourMenu() : void
+setMeilleursTemps() : void
+getMeilleursTemps() : Temps
FermetureFichier
+fermerFichier(path) : void
EnregistrerTemps
-nouveauTemps : int
+ecrireNouveauMeilleurTemps() : void
Temps
-meilleurTempsDébutant : int
-meilleurTempsAvancé : int
+estUnMeilleurTemps() : boolean
EnregistrerPartie
+enregistrerPartieDansFichier(List) : void
OuvertureFichier
+ouvrirFichier(path) : void
ChargerTemps
-temps : int
+chargerMeilleurTemps() : int
Jeu
-échechier : Tableau
-joueur1 : Joueur
-joueur2 : Joueur
-historiqueMouvement : Map<Joueur, List<int[4]>>
+commencerPartie() : void
+gagnant() : Joueur
ChargerPartie
+chagerPartie() : List
CalculTemps
-nouveauTemps : int
+partirChrono() : void
+arreterChrono() : void
EnLigne
-serveur : ConnexionServeur
Tableau
-tableauPieces : Pieces[8][8]
+déplacePièce(pièce, posXDest, posYDest) : void
+initialiserPieces() : void
+estÉchecEtMath() : boolean
+estÉgaliter() : boolean
Joueur
-couleur : String
+mouvement() : List
Pièce
-couleur : String
-posX : int
-posY : int
+déplacementValide(posXDest, posYDest) : boolean
Humain
-name : String
IA
ConnexionServeur
-host : String
-port : int
+connexion() : void
+joueurTrouvé() : boolean
Roi
Dame
Fou
Cavalier
Tour
Pion
IADébutant
IAAvancé
2..32
1..16
1..2
0..1
Powered By Visual Paradigm Community Edition

Diagrammes de séquences

Les diagrammes suivants modélisent les différentes fonctionnalités du système au travers des échanges des objets dans le temps.
Pour chacune des situations qui vont suivre, le diagramme associé présentera les messages échangés par l'utilisateur, le joueur artificiel et le système de jeu.

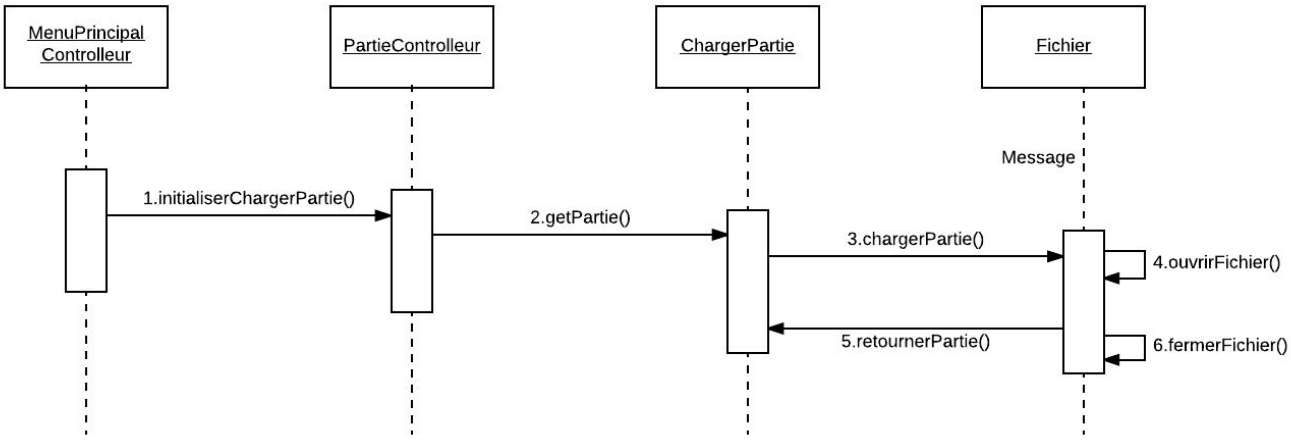
DS1 – Sauvegarder une partie

L’utilisateur souhaite arrêter la partie et voudrait pouvoir la reprendre par la suite. Lorsque l’utilisateur choisit un joueur artificiel comme adversaire et la partie est commencé, celui-ci à l’option de sauvegarder ça partie en cours. La partie se sauvegarde dans un document XML sur la machine de l’utilisateur.



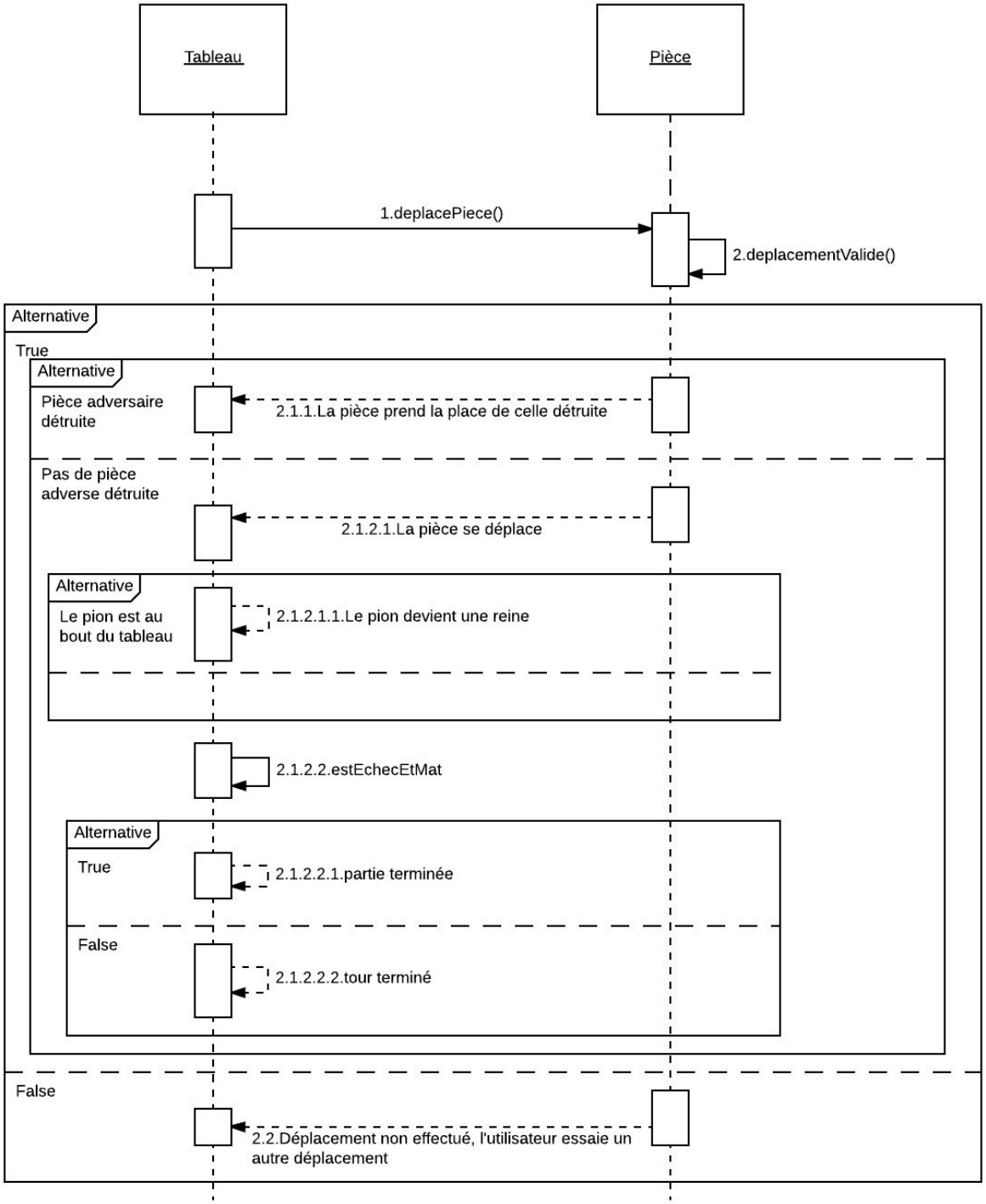
DS2 – Charger une partie

L'utilisateur souhaite reprendre une partie sauvegardé. Dans le menu principal, l'utilisateur à l'option de charger une partie qui a été précédemment sauvegardé dans un fichier XML de sa machine local.



DS3 – Jouer un coup

L'utilisateur et l'adversaire (un joueur humain ou artificiel) jouent des coups jusqu'à ce que la partie soit nulle ou terminé (c'est-à-dire lorsqu'un des deux rois à été éliminé).

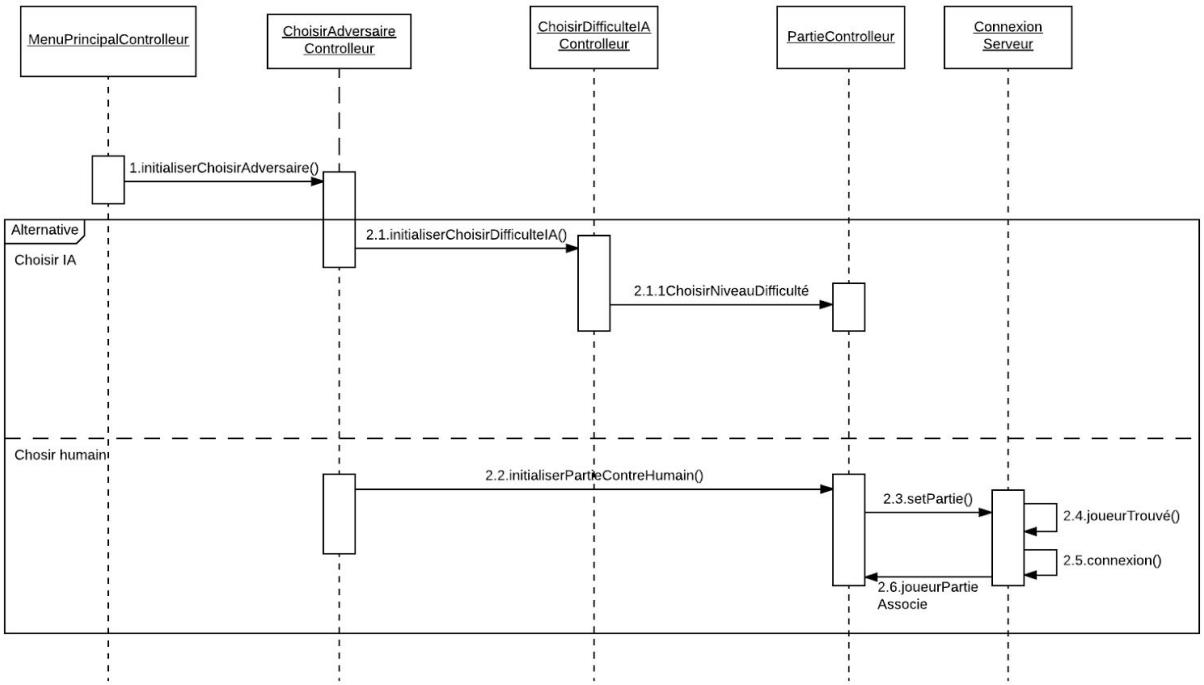


DS4 – Lancer une partie

L’utilisateur peut choisir un niveau de difficulté débutant ou avancé. Ce cas s’applique seulement lorsque l’utilisateur décide de jouer contre un joueur artificiel. Une fois le niveau de difficulté choisie, la partie commence.

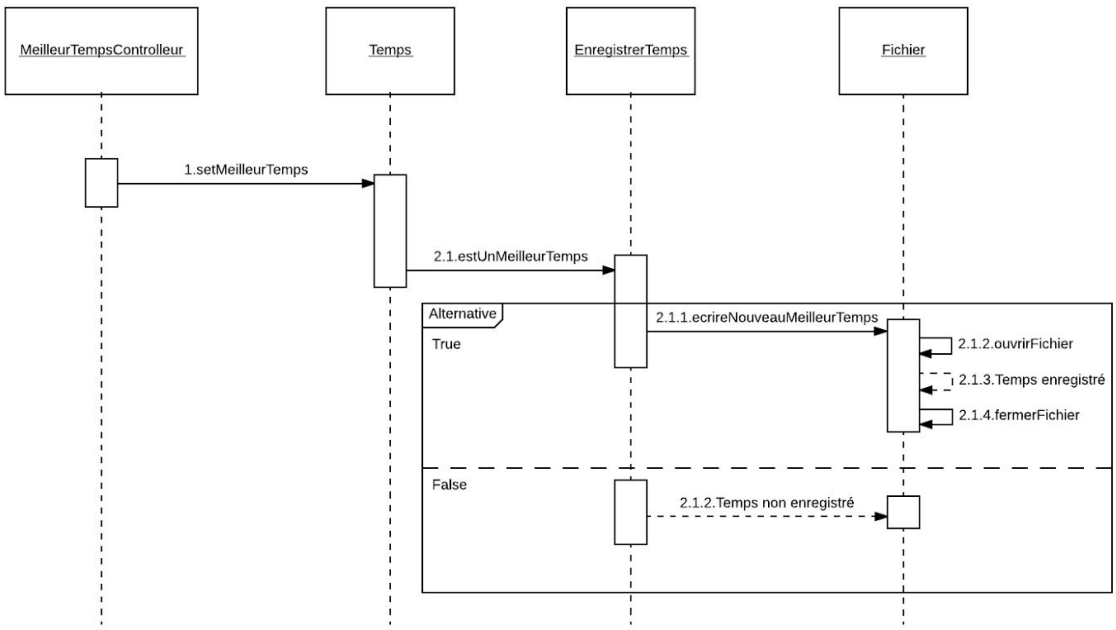
DS5 – Choisir un adversaire

L'utilisateur a le choix de jouer une partie contre un joueur humain ou un joueur artificiel. Dans le cas que l'utilisateur décide de joueur contre un humain, une connexion au serveur web est établie. Puis, un adversaire humain est affecté à l'utilisateur et la partie commence. Dans le cas que l'utilisateur décide de jouer contre un joueur artificiel, un menu va s'afficher à l'utilisateur pour le choix de difficulté (DS4). Une fois la difficulté choisie, la partie peut commencer.



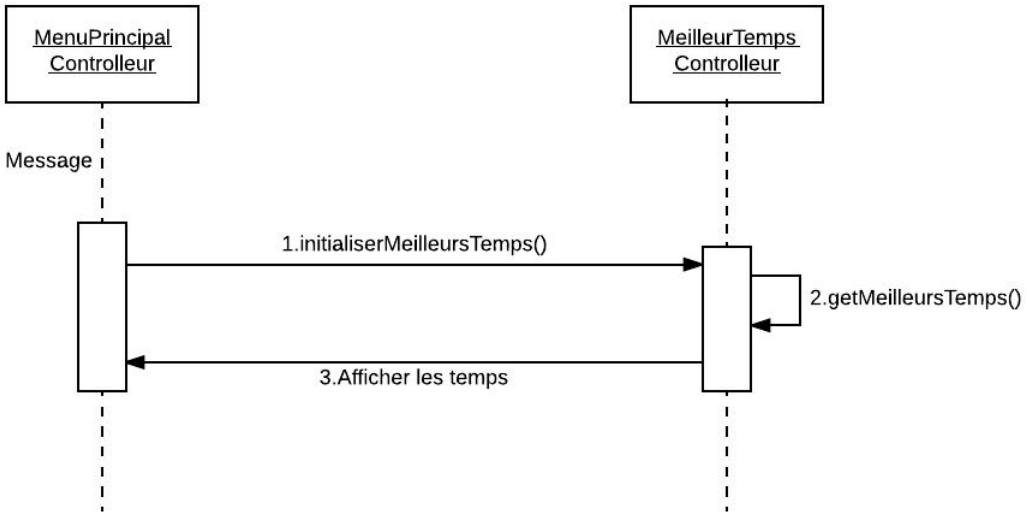
DS6 - Enregistrer un nouveau temps

L’utilisateur a eu un nouveau record de temps dans un niveau de difficulté contre un joueur artificiel. Le nouveau record de temps, le nom de l’utilisateur et la difficulté du joueur artificiel vaincu est enregistré dans un fichier sur la machine de l'utilisateur.



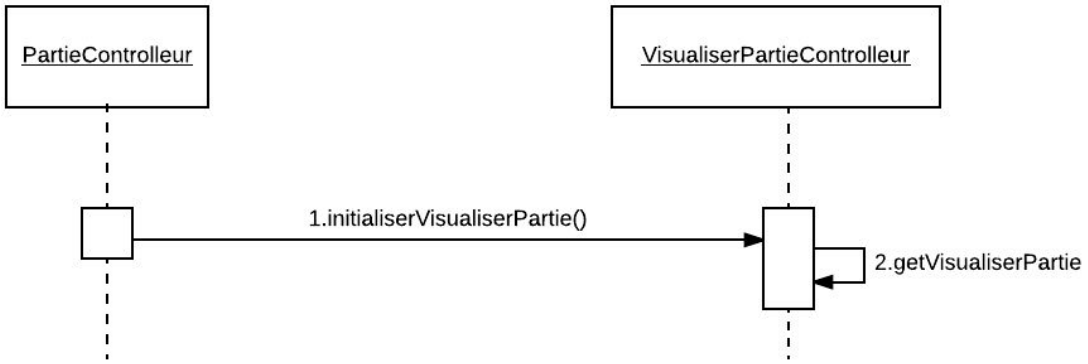
DS7 - Consulter les meilleurs temps

L’utilisateur veut voir les meilleurs temps contre le joueur artificiel. Cette page va afficher à l’utilisateur le nom du joueur, son meilleur temps et le niveau de difficulté du joueur artificiel vaincu.



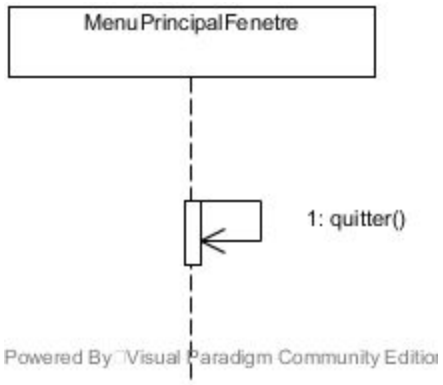
DS8 - Visualiser une partie

À la fin d'une partie, l'utilisateur veut revoir la partie dans son intégralité. Une fois la visualisation activé, un tableau d’échec d’une partie vierge est affiché. Puis, après chaque fois que l’utilisateur appuie sur le bouton pour afficher le coup suivant, le tableau d’échec se modifie avec les mouvements précédemment enregistrés. Une fois la partie en mode visualisation est terminé, le mode visualisation est quitté.



DS9 - Quitter le jeu

Lorsque l’utilisateur se trouve au menu principal, il a la possibilité de quitter le jeu.



DS10 - Retourner au menu principal

Lorsque l’utilisateur se trouve sur la fenêtre de la partie d’échec, la fenêtre pour visualiser une partie ou la fenêtre pour consulter les meilleurs scores. L’utilisateur peut retourner au menu principal.

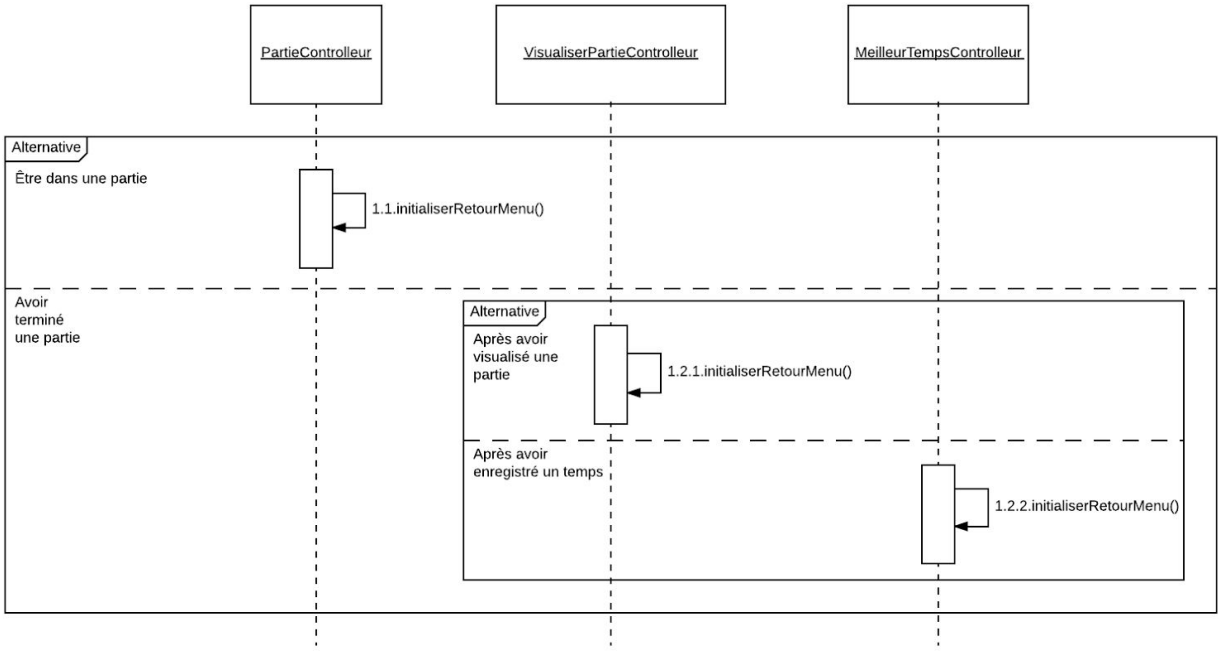
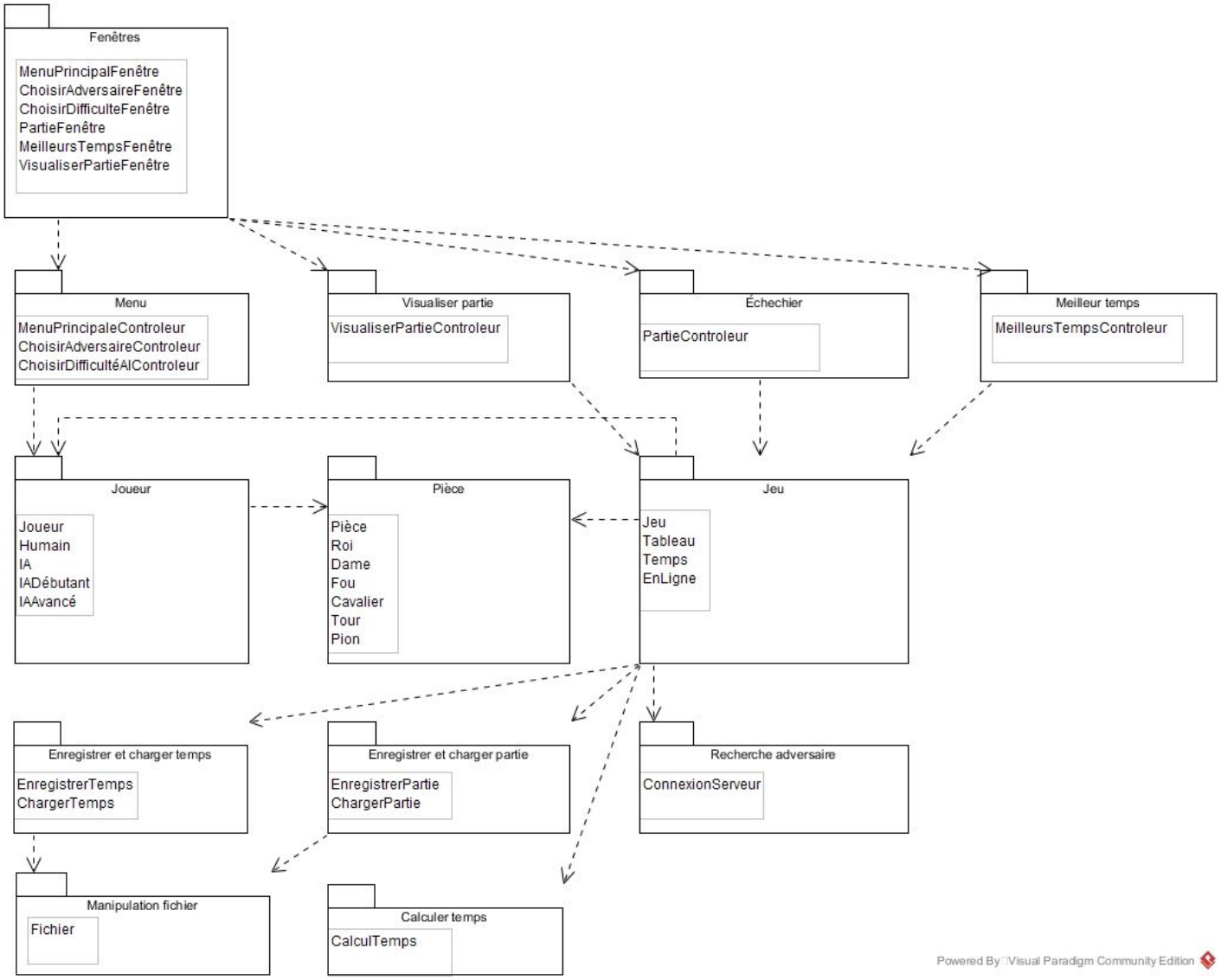


Diagramme de packages du système

Le diagramme suivant établit les packages du systèmes et leurs liens. Pour le développement du logiciel, nous utilisons l'architecture en couche détaillées. Chaque couche du diagramme de package représente en ordre: la présentation, l'application, le domaine, les services techniques et la fondation. Les classes sont indiquées pour chaque package.



Les patrons GRASP

Chaque responsabilité/action des classes sont justifiée par un pattern GRASP. Cette justification est documentée pour chaque méthode publique et pour chaque création des instances d'une classe.

Couche application

Package Menu

- MenuPrincipaleControleur

  Méthode : initialiserChoisirAdversaire()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu principal pour aller au menu de choix d'un adversaire. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : inialiserChargerPartie()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu principal pour aller directement à la fenêtre d'une partie. Aucun adversaire sera sélectionné et un fichier XML représentant d'une partie antérieur sauvegardé sera demandé. Si aucun fichier n'est fournis l'utilisateur reste sur le menu principal. Si le fichier sélectionné est valide, la partie reprend. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : initialiserMeilleurTemps()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu principal pour aller au menu de choix d'un adversaire. Le contrôleur effectue la transition entre les deux fenêtres.

- ChoisirAdversaireControleur

  Méthode : initialiserChoisirDifficultéIA()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu choisir un adversaire pour aller au menu de choix d'une difficulté. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : initialiserPartieContreHumain()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu choisir un adversaire pour aller à la fenêtre d'une partie. Le contrôleur effectue la transition entre les deux fenêtres.

Méthode : jouerContreHumain()
Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode retourne vrai si l'utilisateur désire de jouer contre un humain et faux s'il désire jouer contre une IA. Le contrôleur effectue la transformation des données de la couche présentation.

- ChoisirDifficultéAIControleur

  Méthode : initialiserPartieIA()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir du menu choisir une difficulté pour IA pour aller à la fenêtre d'une partie. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : jouerContreQuelAI()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode retourne un entier propre à la difficulté de l'AI choisie par l'utilisateur. Le contrôleur effectue la transformation des données de la couche présentation.

Package Échechier

- PartieControleur

  Méthode : initialiserRetourMenu()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir de la fenêtre d'une partie en cours pour aller au menu principal. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : initialiserVisualiserPartie()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir de la fenêtre d'une partie en cours pour aller visualiser la dernière partie terminé. Si aucune partie n'est terminé la méthode est désactivé. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : setEchecier()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet d'initialiser un échechier dans le but d'être utilisé par la couche présentation. Il y a deux types d'initialisation, l'initialisation des pièces de l'échechier par défaut et l'initialisation faite à partir d'une partie chargé. Le contrôleur a reçu les commandes de la présentation et manipule les objets (Tableau) du domaine.

  Méthode : setPiece()

Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode permet de bouger une pièce d'échec avec les coordonnées saisie en entré à la couche présentation. Le contrôleur a reçu les commandes de la présentation et manipule les objets (Piece) du domaine.

Méthode : setJoueur()
Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode permet de saisir un nom à l'utilisateur (Joueur) et instancier un adversaire humain à une partie (Jeu). Le contrôleur a reçu les commandes de la présentation et manipule les objets (Humain) du domaine.

Méthode : setAI()
Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode permet d'instancier un adversaire IA à une partie (Jeu). Le contrôleur a reçu les commandes de la présentation et manipule les objets (IA) du domaine.

Méthode : setJeu()
Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode permet d'instancier une partie avec un échechier et deux joueurs (Humain ou IA). Le contrôleur a reçu les commandes de la présentation et manipule les objets (Joueur) du domaine.

Méthode : getJeu()
Justification : Utilise le patron de GRASP contrôleur.
Documentation : Cette méthode permet de retourner les données d'une partie (Jeu). Le contrôleur retourne les données transformé par les objets du domaine.

Package Visualiser une partie

- VisualiserPartieControleur

  Méthode : initialiserRetourMenu()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir de la fenêtre visualiser une partie pour aller au menu principal. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : getVisualiserPartie()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de retourner les données de la dernière partie jouer. Le contrôleur retourne les données transformé par les objets du domaine.

Package Meilleurs temps

- MeilleursTempsControleur

  Méthode : initialiserRetourMenu()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de partir de la fenêtre meilleurs temps pour aller au menu principal. Le contrôleur effectue la transition entre les deux fenêtres.

  Méthode : setMeilleursTemps()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de modifier les données des meilleurs temps lorsqu'une partie est terminé et que l'utilisateur a battu un meilleur temps. Le contrôleur a reçu les commandes de la présentation et manipule l'objet (Temps) du domaine.

  Méthode : getMeilleursTemps()
  Justification : Utilise le patron de GRASP contrôleur.
  Documentation : Cette méthode permet de retourner le nom, le meilleur temps et la difficulté de l'IA associé pour tous les niveaux de difficultés. Le contrôleur retourne les données transformées par l'objet (Temps) du domaine.

Couche du domaine

Package Joueur

- Joueur

  Méthode : mouvement()
  Justification : Utilise le patron GRASP polymorphisme.
  Documentation : Cette méthode à des comportements connexes pour les classes (Humain, IADébutant et IAAvancé). La méthode affecte un nouveau comportement à l'endroit de sa hiérarchie pour chaque type de joueur.

Package Pièce

- Pièce

  Méthode : déplacementValide()
  Justification : Utilise le patron GRASP polymorphisme.

Documentation : Cette méthode à des comportements connexes pour les classes (Roi, Dame, Pion, Fou, Cavalier et Tour). La méthode affecte un nouveau comportement à l'endroit de sa hiérarchie pour chaque pièce d'échec.

Package Jeu

- Jeu

  Instance de classe : Tableau
  Justification : Utilise le patron de GRASP créateur.
  Documentation : L'instance de Tableau détient de l'informations nécessaire pour instancier une partie (Jeu).

  Instance de classe : Joueur
  Justification : Utilise le patron de GRASP créateur.
  Documentation : Les instances de Joueur détiennent de l'informations nécessaire pour instancier une partie (Jeu).

  Méthode : commencerPartie()
  Justification : Utilise le patron GRASP faible couplage.
  Documentation : La classe Jeu a la responsabilité de créer et manipuler le jeu complet d'échec. Elle contient tous les informations nécessaire pour manipuler une partie d'échec. La méthode commencé partie peut remplir sa responsabilité de commencer une partie avec les informations de la classe.

  Méthode : gagnant()
  Justification : Utilise le patron GRASP faible couplage.
  Documentation : La classe Jeu a la responsabilité de créer et manipuler le jeu complet d'échec. Elle contient tous les informations nécessaire pour manipuler une partie d'échec. La méthode gagnant peut remplir sa responsabilité de trouver le gagnant avec les informations de la classe.

- Tableau

  Instance de classe : Piece
  Justification : Utilise le patron GRASP créateur.
  Documentation : La classe Tableau contient des pièces d'échec.

  Méthode : déplacePièce()
  Justification : Utilise le patron GRASP expert en information.
  Documentation : L'objet Tableau contient l'information de tous les objets Pièce pour manipuler l'échechier. La méthode déplacer une pièce peut remplir sa responsabilité de manipuler des pièces sur l'échechier.

  Méthode : initialiserPieces()

Justification : Utilise le patron GRASP expert en information.
Documentation : L'objet Tableau contient l'information de tous les objets Pièce pour manipuler l'échechier. La méthode initialiser des pièces peut remplir sa responsabilité de manipuler des pièces sur l'échechier.

Méthode : estÉchecEtMath()
Justification : Utilise le patron GRASP expert en information.
Documentation : L'objet Tableau contient l'information de tous les objets Pièce pour manipuler l'échechier. La méthode est échec et math peut remplir sa responsabilité d'effectuer une validation des pièces (Roi) sur l'échechier.

Méthode : estÉgalité()
Justification : Utilise le patron GRASP expert en information.
Documentation : L'objet Tableau contient l'information de tous les objets Pièce pour manipuler l'échechier. La méthode est égalité peut remplir sa responsabilité d'effectuer une validation des pièces (Roi) sur l'échechier.

- Temps

  Méthode : estUnMeilleurTemps()
  Justification : Utilise le patron GRASP expert en information.
  Documentation : L'objet Temps contient les informations nécessaire pour manipuler les données sur les meilleurs temps. Donc, la méthode peut remplir sa responsabilité de manipuler des données sur les meilleurs temps.

- EnLigne

  Instance de classe : ConnexionServeur
  Justification : Utilise le patron GRASP créateur.
  Documentation : L' instance de ConnexsionServeur détient de l'informations nécessaire pour instancier une partie (EnLigne).

Couche services techniques

Package Enregistrer et charger temps

- EnregistrerTemps

  Méthode : écrireNouveauMeilleurTemps()
  Justification : Utilise le patron GRASP forte cohésion.
  Documentation : La méthode écrire le meilleur temps d'un utilisateur qui a battu l'IA pour une certaine difficulté est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

- ChargerTemps

  Méthode : chargerMeilleurTemps()
  Justification : Utilise le patron GRASP forte cohésion.
  Documentation : La méthode chargé le meilleur temps d'un utilisateur qui a battu l'IA pour une certaine difficulté est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

Package Enregistrer et charger partie

- EnregistrerPartie

  Méthode : enregistrerPartieDansFichier()
  Justification : Utilise le patron GRASP forte cohésion.
  Documentation : La méthode enregistrer une partie dans un fichier est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

- ChargerPartie

  Méthode : chargerPartie()
  Justification : Utilise le patron GRASP forte cohésion.
  Documentation : La méthode charger une partie sauvegardé ultérieurement est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

Package Recherche adversaire

- ConnexionServeur

  Méthode : connexion()
  Justification : Utilise le patron GRASP contrôleur.
  Documentation : La méthode connexion consiste au programme à établir une connexion au serveur est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

  Méthode : joueurTrouvé()
  Justification : Utilise le patron GRASP contrôleur.
  Documentation : La méthode joueur trouvé qui consiste à déterminer si un adversaire humain a été trouvé est dans une classe qui effectue qu'une seule responsabilité. Ce qui rend la classe simple et compréhensible tout en gardant un faible couplage.

Couche fondation

Package Manipulation fichier

- Fichier

  Méthode : ouvrirFichier()
  Justification : Utilise le patron GRASP fabrication pure.
  Documentation : Cette méthode génériques est dans une classe qui n'est pas lier aux classes du domaine. Cette classe préserve sa cohésion et est donc totalement réutilisable.

  Méthode : fermeFichier()
  Justification : Utilise le patron GRASP fabrication pure.
  Documentation : Cette méthode génériques est dans une classe qui n'est pas lier aux classes du domaine. Cette classe préserve sa cohésion et est donc totalement réutilisable.

Package Calculer temps

- CalculTemps

  Méthode : partirChrono()
  Justification : Utilise le patron GRASP fabrication pure.
  Documentation : La méthode pour partir un chronomètre est générique et est dans une classe qui n'est pas lier aux classes du domaine. Cette classe préserve sa cohésion et est donc totalement réutilisable.

  Méthode : arreterChrono()
  Justification : Utilise le patron GRASP fabrication pure.
  Documentation : La méthode arrêter un chronomètre est générique et est dans une classe qui n'est pas lier aux classes du domaine. Cette classe préserve sa cohésion et est donc totalement réutilisable.

Glossaire

- IA : Une intelligence artificielle.
- Menu: Un menu est une fenêtre constituée seulement de boutons.