Louis-Félix Veillette

Olivier Savoie

Dictionnaire des classes

Travail présenté à

Charles Jacob

31 mars 2022

Cegep régional de Lanaudière

Table des matières

[Vues : 3](#_Toc99098179)

[Contrôleur : 5](#_Toc99098180)

[Modèles 6](#_Toc99098181)

# Vues :

|  |  |  |  |
| --- | --- | --- | --- |
| FormMenu  Description : Formulaire du menu permettant de commencer une nouvelle partie, de voir le classement des joueurs et de quitter le jeu. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_controlleur | Echec | Variable permettant de remonter au controller pour le mettre à jour |
| *Méthodes* | | | |
| Public | btnNouvelle\_Click() | void | Événement du bouton « Nouvelle partie » qui actionne une nouvelle partie |
| Public | btnClassement\_Click() | Void | Événement du bouton  « Classement » qui affiche le classement |
| Public | FormMenu\_FormClosed() | Void | Événement lors de la fermeture de ce formulaire (Menu principal) qui ferme tout le jeu |
| Public | btnQuitter\_Click() | Void | Événement du clic du bouton « Quitter le jeu » |

|  |  |  |  |
| --- | --- | --- | --- |
| FormSelection  Description : Formulaire de la sélection des joueurs. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_controlleur | Echec | Variable permettant de remonter au controller pour le mettre à jour |
| *Méthodes* | | | |
| Public | btnCommencerClick() | void | Événement du clic du bouton « Commencer » qui débute la nouvelle partie |
| Public | FormSelection\_Load() | Void | Affiche le formulaire de sélection avec la liste des joueurs dans chaque ComboBox pour sélectionner les joueurs |
| Public | cmbJoueur1\_SelectedIndexChanged() | Void | Événement de la sélection du premier joueur pour jouer, qui sélectionne le premier joueur et l'enlève de la deuxième liste |
| Public | cmbJoueur2\_SelectedIndexChanged | Void | Événement de la sélection du deuxième joueur pour jouer, qui sélectionne le deuxième joueur et l'enlève de la première liste |

|  |  |  |  |
| --- | --- | --- | --- |
| FormClassement  Description : Formulaire du classement des joueurs. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_controlleur | Echec | Variable permettant de remonter au controlleur. |
| Privé | \_joueurs | List<Joueur> | Liste des joueurs créés |
| *Méthodes* | | | |
| Public | FormClassement\_Load() | void | Événement de l'affichage du formulaire du classement des joueurs |
| Public | btnAjouter\_Click () | Void | Ajoute le nouveau joueur au classement |
| Public | btnSupprimer\_Click() | Void | Supprime le joueur sélectionné du classement |

|  |  |  |  |
| --- | --- | --- | --- |
| FormPartie  Description : Formulaire de la partie où se trouve l’échiquier et le choix des joueurs pour la partie. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_controlleur | Echec | Variable permettant de remonter au controlleur. |
| Privé | \_indexPartie | int | Identifiant de la partie créée |
| Privé | clickInitial | Point | Coordonnées de la case initiale |
| Privé | clickDest | Point | Coordonnées de la case destination |
| *Méthodes* | | | |
| Private | btnAbandon\_Click() | void | Événement du clique du bouton « abandonner » pour terminer la partie |
| Private | btnNulle\_Click() | Void | Événement du clique du bouton « Nulle » pour terminer la partie |
| Private | pnlEchiquier\_Paint() | Void | Met à jour l’échiquier et positionne les images des pièces correctement sur l’échiquer ainsi que les messages sur les tours des joueurs. |
| Private | pnlEchiquier\_Click() | Void | Événement des cliques sur l'échiquier. Joue le coup et affiche les messages si possibles. |
| Public | peinturerEchiquier() | Void | Affiche les images des pièces sur l'échiquier sur leurs bons emplacements. |
| Public | finPartie() | void | Termine la partie dépendamment de la raison de cette dernière. |

|  |  |  |  |
| --- | --- | --- | --- |
| FormPromotion  Description : Formulaire qui s’affiche lors d’une promotion d’un pion. Affiche un choix de pièce qui pourra remplacer le pion | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_controlleur | Echec | Variable permettant de remonter au contrôleur. |
| Privé | \_indexPion | Int | Emplacement (index) où se trouve le pion qui s’apprête à être promu |
| *Méthodes* | | | |
| Private | pnl1\_Paint() | void | Dessine l'image de la reine dans le formulaire |
| Private | pnl2\_Paint() | Void | Dessine l'image du fou dans le formulaire |
| Private | pnl3\_Paint() | Void | Dessine l’image du cavalier dans le formulaire |
| Private | pnl4\_Paint() | Void | Dessine l’image de la tour dans le formulaire |
| Private | pnl1\_Click() | void | Événement du clic de l’image sur la reine, ce qui promouvoit le pion en reine |
| Private | pnl2\_Click() | Void | Événement du clic de l’image du fou, ce qui promouvoit le pion en fou |
| Private | pnl3\_Click() | Void | Événement du clic de l’image du cavalier, ce qui promouvoit le pion en cavalier |
| Private | pnl4\_Click() | Void | Événement du clic de l’image de la tour, ce qui promouvoit le pion en tour |

# Contrôleur :

|  |  |  |  |
| --- | --- | --- | --- |
| Echec  Description : Le jeu d’échec permettant de jouer plusieurs parties et de montrer les joueurs. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_formMenu | FormMenu | Menu principal du jeu |
| Privé | \_formPartie | FormPartie | Formulaire d’une partie |
| Privé | \_formSelection | FormSelection | Formulaire de sélections des deux joueurs qui joueront la partie |
| Privé | \_formPromotion | FormPromotion | Formulaire de promotion des pions |
| Privé | \_unePartie | Partie | L’une des parties d’échec étant commencé |
| Privé | \_listeJoueur | List<Joueur> | Liste des joueurs du jeu |
| Privé | \_listePartie | List<Partie> | Liste des parties étant commencées |
| Privé | \_listeFormPartie | List<FormPartie> | Liste des formulaires des parties étant commencées |
| *Méthodes* | | | |
| Public | nouvellePartie() | Void | Ouvre l formulaire de sélection de joueurs pour la nouvelle partie |
| Public | commencerPartie(string joueur1, string joueur2) | Void | Commence la nouvelle partie et active et garde en mémoire les joueurs qui vont s'affronter |
| Public | jouerCoup (FormPartie monForm, int x1, int y1, int x2, int y2) | Tuple<bool, int> | Vérifie si le coup est valide et fait si possible le déplacement ainsi que la promotion s'il y a lieu. Retourne vrai et un entier représentant que le coup est valide si le coup est valide |
| Public | changerPion(char piece, int indexPion) | Void | Change le pion qui est promu par la pièce que le joueur aura sélectionnée. |
| Public | Victoire\_Abandon(FormPartie monForm) | Void | Abandon de la partie |
| Public | uneNulle(FormPartie monForm) | Void | Partie nulle |
| Public | tour(FormPartie monForm) | Int | Indique le tour du joueur dans la bonne partie. Retourne le tour du joueur. |
| Public | Tour() |  |  |

# Modèles

|  |  |  |  |
| --- | --- | --- | --- |
| Partie  Description : Partie d’échec ayant les informations de la partie. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_indexInitial | Int | Index de départ où se trouve la pièce cliquée dans l’échiquier |
| Privé | \_indexDestination | Int | Index de destination où se trouvera la pièce après son mouvement |
| Privé | \_plateau | Plateau | Plateau du jeu avec toutes les 64 cases. |
| Privé | \_indexPartie | int | ID de la partie |
| Privé | \_parent | Echec | Variable permettant d’accéder au contrôleur Echec |
| Privé | \_nbCoup | int | Compte du nombre de coups joués. |
| Privé | \_coupDepuisPionJoué | Int | Compte le nb de coups depuis le dernier mouvement d’un pion pour prévenir un null |
| Privé | \_joueur1 | Joueur | Le joueur 1 |
| Privé | \_joueur2 | Joueur | Le joueur 2 |
| Privé | \_listeEchiquier | List<string> | Liste permettant de mémoriser les dernières positions des pièces. |
| *Méthodes* | | | |
| Public | determinerCase(int x1, int y1, int, x2, int y2) | void | Détermine l’index initial de la pièce et l’index d’arrivée |
| Public | verifDeplacementPossible(int indexInitial, int indexDestination) | Tuple<bool, string> | Retourne vrai sans message d’erreur si le déplacement fonctionne |
| Public | faireDeplacement() | Void | Déplace la pièce dans l’échiquier |
| Public | verifLesEchecs | Tuple<bool, string> | Retourne vrai et un message s’il y a un échec, un échec et mat ou un échec et pat. |
| Public | verifListeEchiquier() | Bool | Retourne vrai si la liste des échiquiers se répètent 3 fois |
| Public | viderEchiquier() | Void | Vide la liste d’échiquier |
| Public | verifFinMatchSpecial | Tuple<bool, string> | Retourne vrai et un message si il y a une null spéciale |
| Public | Verif50Coup | Bool | Retourne vrai si cela fait 50 coups qu’un pion n’a pas bougé ou qu’une pièce n’a pas été mangé. |

|  |  |  |  |
| --- | --- | --- | --- |
| Plateau  Description : Plateau (échiquier) de la partie | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_echiquier | List<Case> | Échiquier avec ses 64 cases |
| Privé | \_partie | Partie | La partie où se trouve le plateau |
| *Méthodes* | | | |
| Public | verifierSiPiece(int indexInitial) | Tuple<bool, string> | Retourne vrai sans message d’erreur si la case cliquée détient une pièce |
| Public | maPiece() | Tuple<bool, string> | Retourne vrai sans message d’erreur si la couleur de la pièce où la case a été cliquée est la bonne couleur |
| Public | verifTrajectoire(int indexInitial, int indexDestination) | Tuple<enum, Case[]> | Retourne une énumération qui montre si la trajectoire fonctionne avec certains critères et un tableau de cases qui montre les cases qui passe par la trajectoire. |
| Public | estCollision(int indexDestination) | Tuple<bool, string> | Vérifie si la pièce entre en collision avec une autre pièce sans en avoir la permission lors de sa trajectoire. Retourne vrai avec un message d’erreur s’il y a une collision durant la trajectoire. |
| Public | verifCouleurDesti(int indexDesti) | Tuple<bool, string> | Retourne vrai et un message d’erreur si la pièce de destination est de la même couleur que la pièce de départ |
| Public | metEnEchecAllié(int index) | Tuple<bool, string> | Retourne vrai et un message d’erreur si le mouvement de la pièce cause l’échec sur son propre roi. |
| Public | verifPromoPion(int indexDestination) | bool | Retourne vrai lorsqu’un pion joue s’il peut faire une promotion. Retourne vrai si le pion peut être promu. |
| Public | verifEchec(int indexInitial, int indexDestination) | Tuple<bool, string | Retourne vrai et un message si le coup cause un échec à l’ennemi |
| Public | verifEchecMat(int indexInitial, int indexDestination) | Tuple<bool, string | Retourne vrai et un message si le coup cause un échec et mat à l’adversaire |
| Public | verifPat() | Tuple<bool, string> | Retourne vrai et un message si le coup cause une partie nulle |

|  |  |  |  |
| --- | --- | --- | --- |
| Case  Description : Case du plateau du jeu pouvant détenir une pièce | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_piece | Piece | Pièce se trouve sur cette case. Peut être null |
| Privé | \_plateau | Plateau | Plateau où se trouve la case |
| *Méthodes* | | | |
| Public | Vide() | bool | Retourne vrai si la case est vide |
| Public | savoirCouleurPiece() | Couleur | Retourne la couleur de la pièce |

|  |  |  |  |
| --- | --- | --- | --- |
| Pieces  Description : Pièces du jeu d’échec | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_nom | string | Nom de la pièce |
| Privé | \_couleur | Enum Couleur | Couleur de la pièce |
| Privé | \_parent | Echec | Variable permettant d’accéder au contrôleur Echec |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | estEssentiel() | bool | Vérifie si c’est un roi |
| Public | toString() | String | Retourne la pièce en chainede charactères |

|  |  |  |  |
| --- | --- | --- | --- |
| Tour  Description : La pièce tour | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_aBougé | bool | Devient faux lorsqu’elle bouge |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |
| Public | estEssentiel() | bool | Vérifie si c’est un roi |

|  |  |  |  |
| --- | --- | --- | --- |
| Reine  Description : La pièce reine | | | |
| *Méthodes* | | | |
| Public | estEssentiel() | bool | Vérifie si c’est un roi |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |

|  |  |  |  |
| --- | --- | --- | --- |
| Fou  Description : La pièce du fou | | | |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | estEssentiel() | bool | Vérifie si c’est un roi |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |

|  |  |  |  |
| --- | --- | --- | --- |
| Cavalier  Description : La pièce du cavalier | | | |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | estEssentiel() | bool | Vérifie si c’est un roi |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |

|  |  |  |  |
| --- | --- | --- | --- |
| Pion  Description : La pièce du pion. | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_aBougé | bool | Devient faux lorsqu’elle bouge |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |
| Public | verifPrisePassant(int index) | Bool | Retourne vrai si le pion peut subir une prise en passant. |

|  |  |  |  |
| --- | --- | --- | --- |
| Roi  Description : La pièce du roi | | | |
| *Variables* | | | |
| Permissions | **Nom** | **Type** | **Description** |
| Privé | \_aBougé | bool | Devient faux lorsqu’elle bouge |
| *Méthodes* | | | |
| Public | Regles(int initiale, int destination, Plateau plateau) | Tuple<bool, string | Retourne vrai sans message d’erreur si la trajectoire entre le point initial est et la destination respecte les règles de la pièce qui devra être joué. |
| Public | toString() | String | Retourne une chaine de charactère adapté à la pièce |