Plan de Test

Objectif :

Tester le bon fonctionnement de la classe Partie et de ses méthodes associées dans l'application Puissance 4.

Test du constructeur par défaut

Contexte : Ce test vérifie que lorsque la classe Partie est instanciée avec son constructeur par défaut, tous ses attributs sont correctement initialisés. Cela garantit que l'objet Partie commence dans un état cohérent avant toute interaction.

Description du Test :

- Créer une instance de la classe Partie sans arguments.
- Vérifie que le plateau de jeu n'est pas nul.
- Vérifie que le mode de jeu par défaut est false.
- Vérifie que les compteurs de victoires pour les joueurs rouge et jaune sont initialisés à 0.

Test de la modification du mode de jeu

Contexte : Ce test s'assure que la méthode **ModifierMode** fonctionne correctement en modifiant le mode de jeu. Il est essentiel pour permettre aux joueurs de basculer entre différents modes de jeu.

Description du Test :

- Créer une instance de la classe Partie.
- Appelle la méthode **ModifierMode** avec la valeur true.
- Vérifie que la propriété modeDeJeu de l'objet Partie est mise à jour à true.

Test de la modification des dimensions du plateau

Contexte : Ce test valide que la méthode **ModifierPlateau** permet de changer les dimensions du plateau de jeu. Cela permet d'adapter le jeu à différentes configurations.

Description du Test :

- Créer une instance de la classe Partie.
- Modifie les dimensions du plateau en appelant **ModifierPlateau** avec les valeurs 10 pour la hauteur et 8 pour la largeur.
- Vérifie que la hauteur et la largeur du plateau sont correctement mises à jour.

Test de la vérification si une colonne est pleine

Contexte : Ce test vérifie le comportement de la méthode **colonnePleine**, qui détermine si une colonne du plateau est remplie jusqu'à sa capacité maximale. Cela est crucial pour le fonctionnement du jeu, car il empêche les joueurs de placer des jetons dans une colonne pleine.

Description du Test :

- Créer une instance de la classe Partie.
- Remplit la première colonne du plateau jusqu'à sa hauteur maximale.
- Appelle la méthode **colonnePleine** pour la première colonne et vérifie qu'elle retourne true.

Test de la méthode MinMax

Contexte : Ce test évalue le comportement de la méthode **MinMax**, qui est souvent utilisée dans les algorithmes de jeu pour déterminer le meilleur coup à jouer en simulant différents scénarios. Le test s'assure que la méthode retourne un score négatif si la situation est défavorable pour le joueur actuel.

Description du Test :

- Crée une instance de la classe Partie.
- Utilise une grille simulée (GrilleMock) pour tester le comportement de la méthode **MinMax**.
- Appelle **MinMax** avec une profondeur de 3 et le joueur actuel étant true.
- Vérifie que le résultat de **MinMax** est négatif, indiquant une évaluation défavorable.

Test de la fermeture des fenêtres

Contexte : Ce test s'assure que toutes les fenêtres ouvertes dans l'application peuvent être correctement fermées. Cela est important pour la gestion des interfaces utilisateur et pour garantir une expérience utilisateur fluide.

Description du Test :

- Créer une instance de la classe Partie.
- Simule l'ouverture de plusieurs fenêtres (par exemple, "Paramètres", "Aide", "Menu").
- Ferme toutes les fenêtres simulées.
- Vérifie que la liste des fenêtres ouvertes est vide après la fermeture.