**Rapport Projet Jeu TicTacToe**

**By Mohamed Yassine Ezzaouia IDS4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Partie I

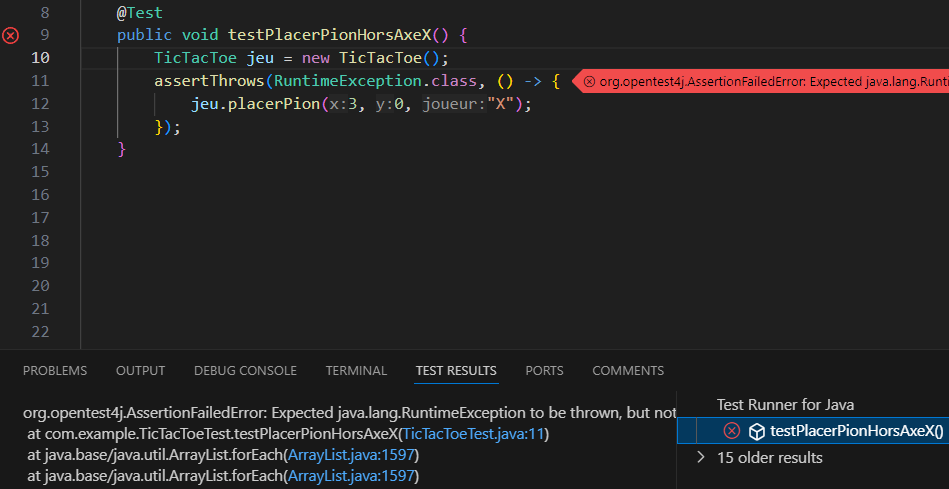
**Exigence 1**

Etape1 : Vérifier qu'un pion placé en dehors de l'axe des X génère une exception

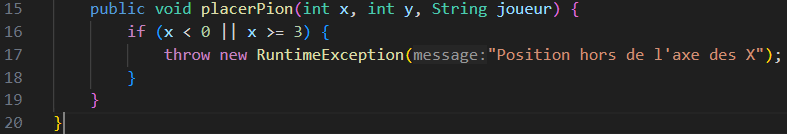
-On commence par définir la grille du jeu TicTacToe



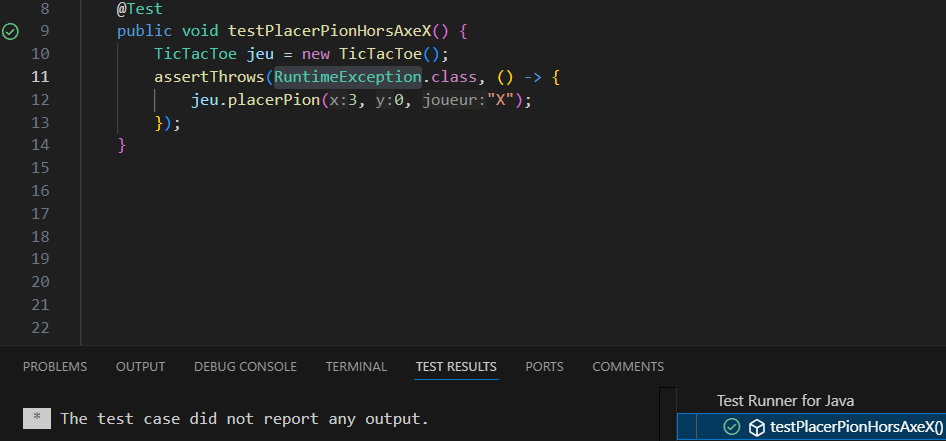
-Ensuite on commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

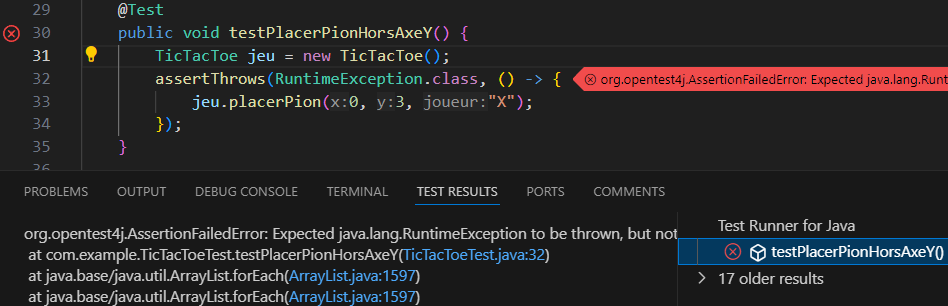


-On réexécute le test afin de vérifier le succès

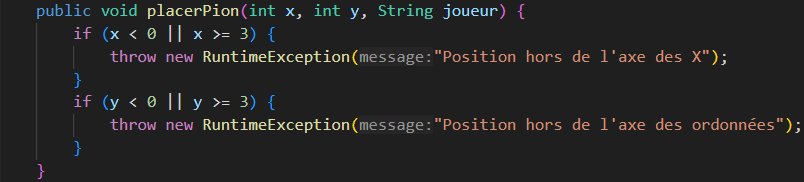


Etape2 : Vérifier qu'un pion placé en dehors de l'axe des ordonnées génère une exception

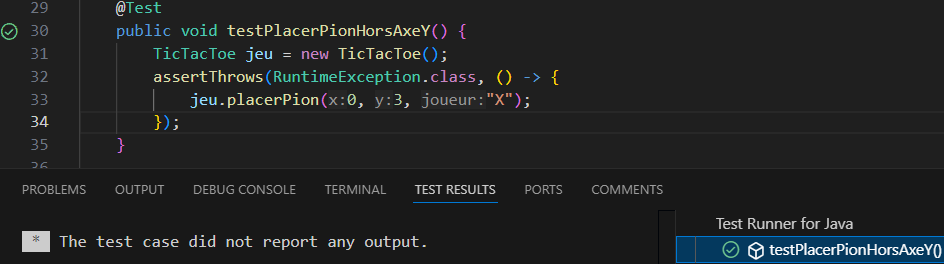
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test



-On réexécute le test afin de vérifier le succès

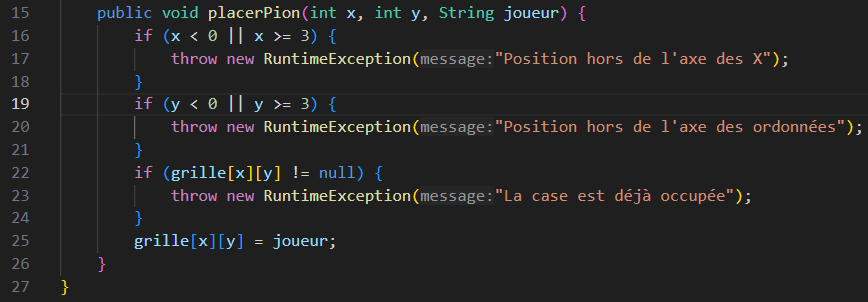


Etape3 : Vérifier qu'un pion placé sur une case déjà occupée génère une exception

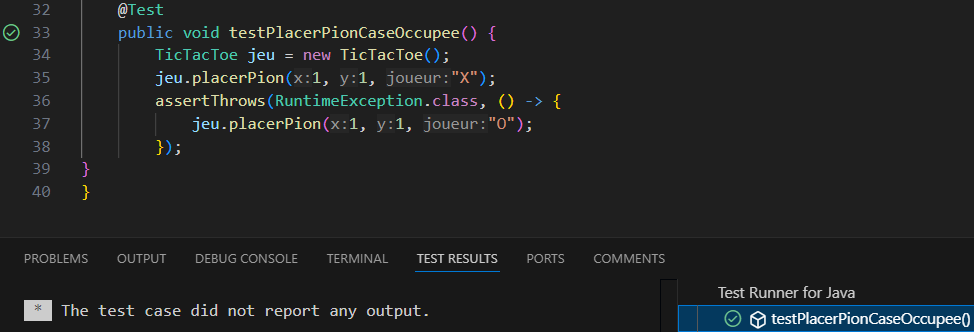
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test



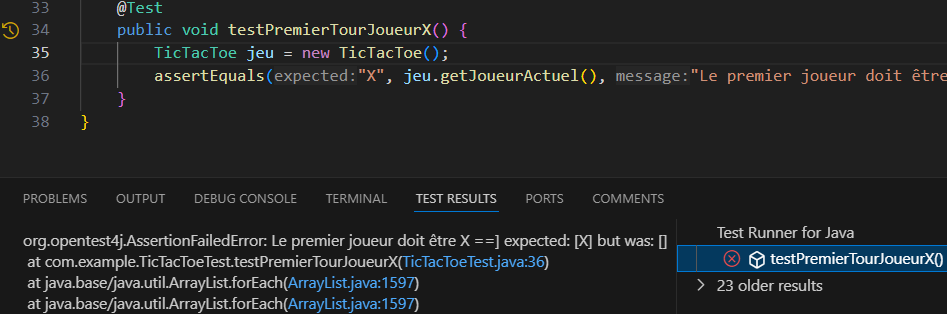
-On réexécute le test afin de vérifier le succès



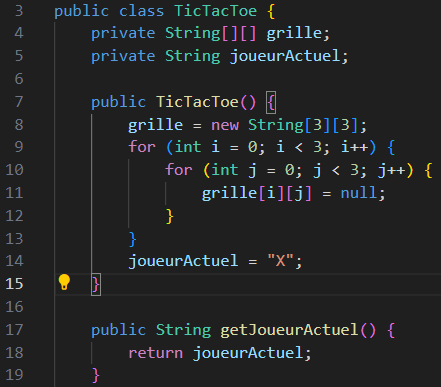
**Exigence 2**

Etape1 : Vérifier que le premier tour est joué par le joueur X

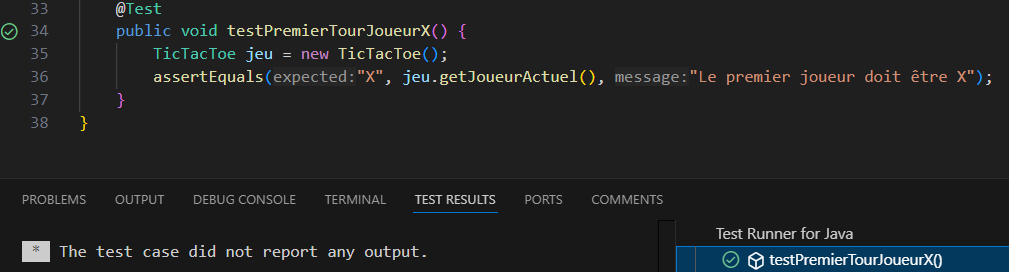
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

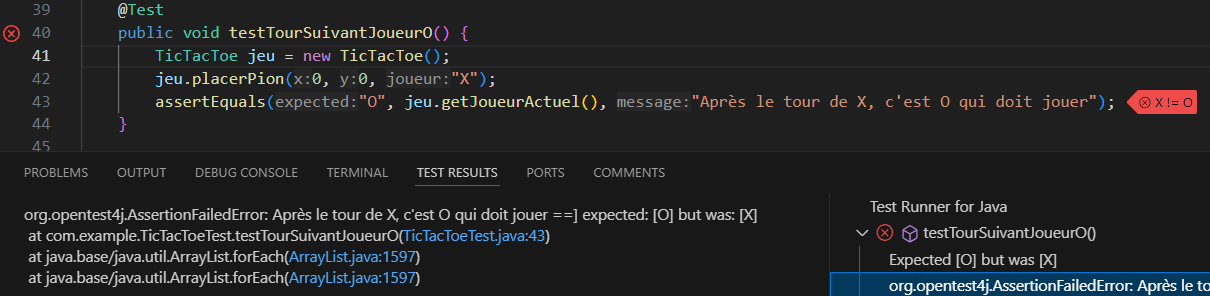


-On réexécute le test afin de vérifier le succès



Etape2 : Vérifier que si le dernier tour a été joué par X, le tour suivant doit être joué par O

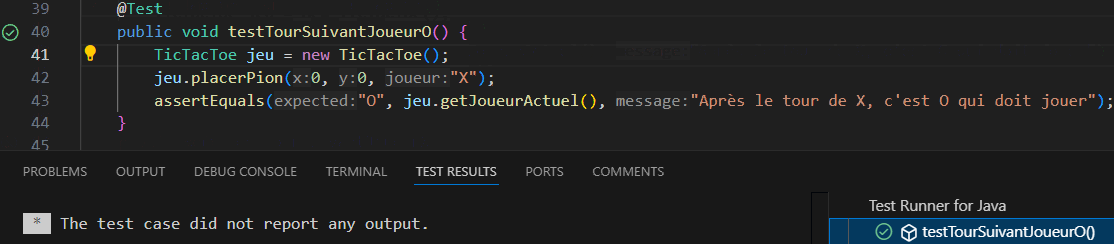
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

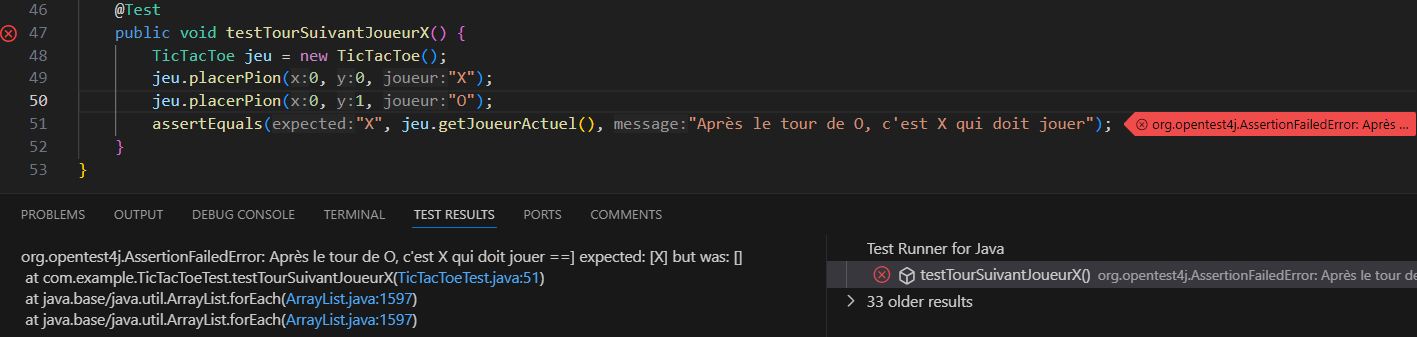


-On réexécute le test afin de vérifier le succès



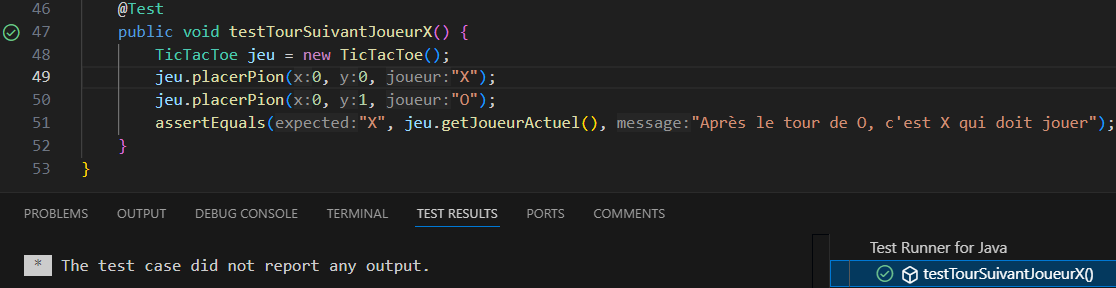
Etape3: Vérifier que si le dernier tour a été joué par O, le tour suivant doit être joué par X

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test(sans implémentation du code de l’étape précédente)



-On implémente le code de l’étape précédente pour faire passer le test

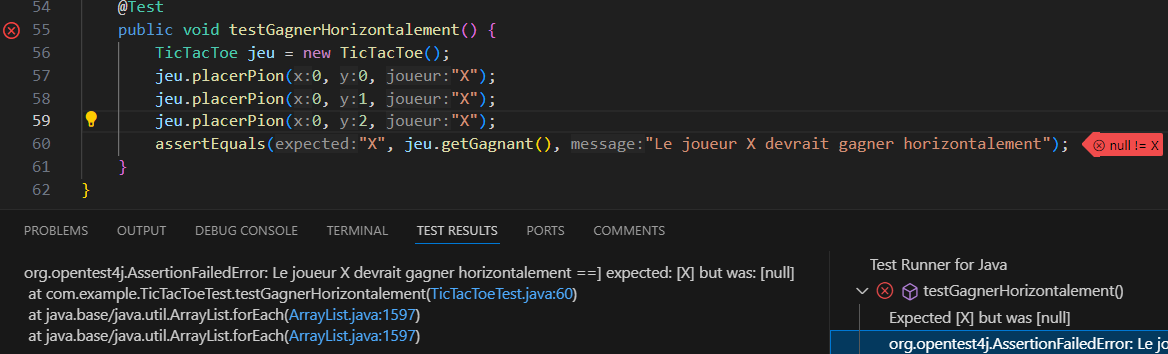
-On réexécute le test afin de vérifier le succès



**Exigence 3**

Etape1 : Vérifier qu'un joueur gagne en ayant une ligne horizontale complète

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

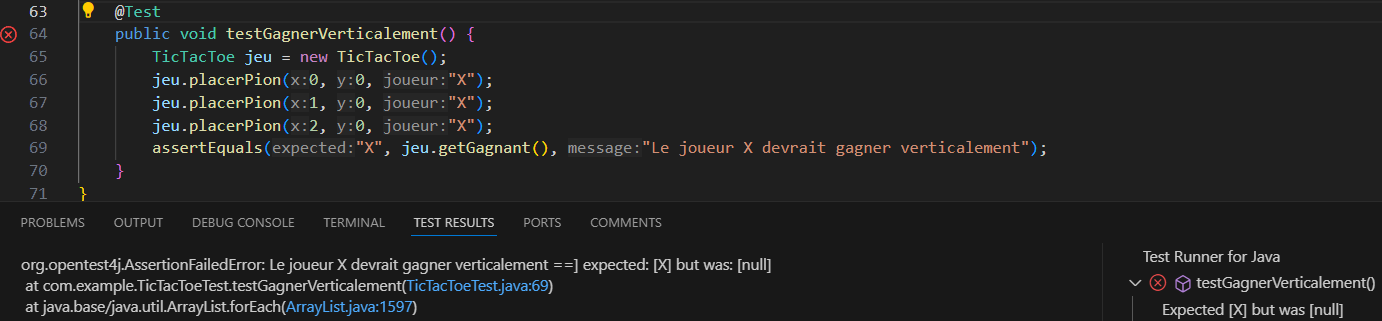


-On réexécute le test afin de vérifier le succès

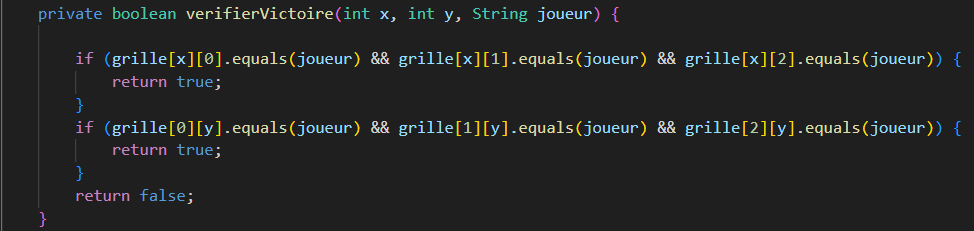


Etape2 : Vérifier qu'un joueur gagne en ayant une ligne verticale complète

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

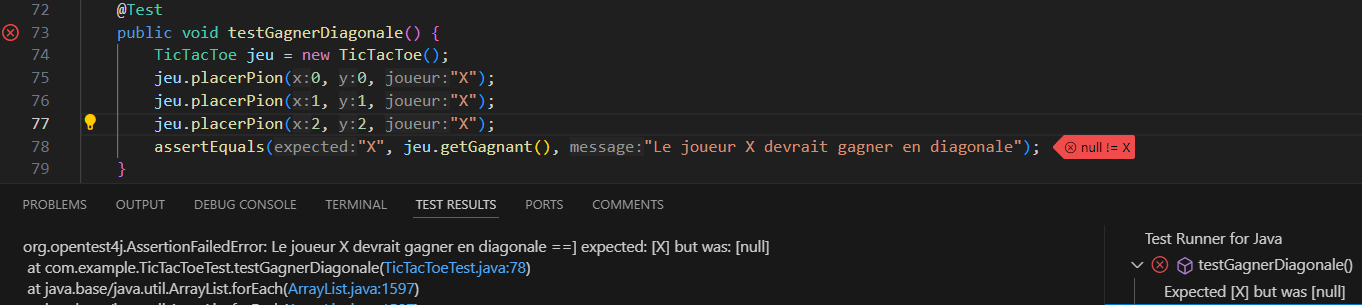


-On réexécute le test afin de vérifier le succès

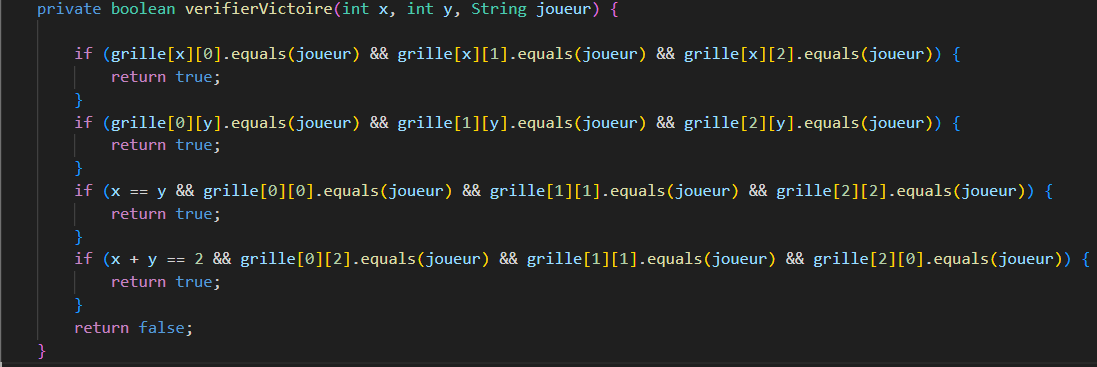


Etape3 : Vérifier qu'un joueur gagne en ayant une diagonale complète

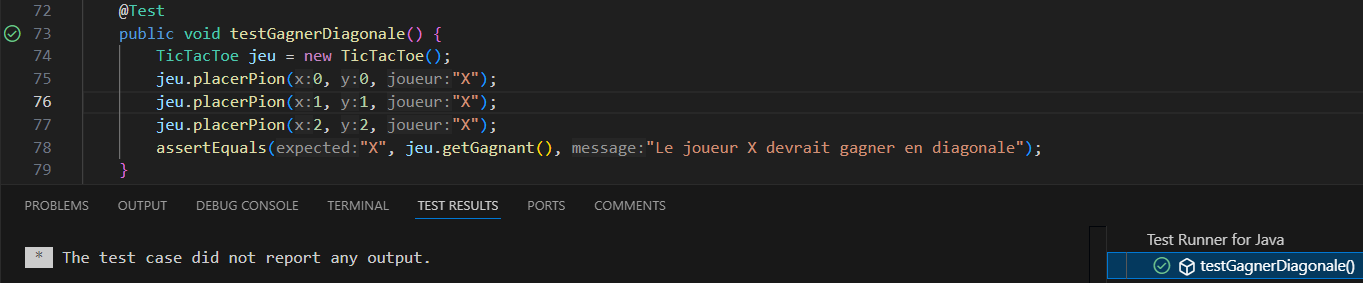
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test



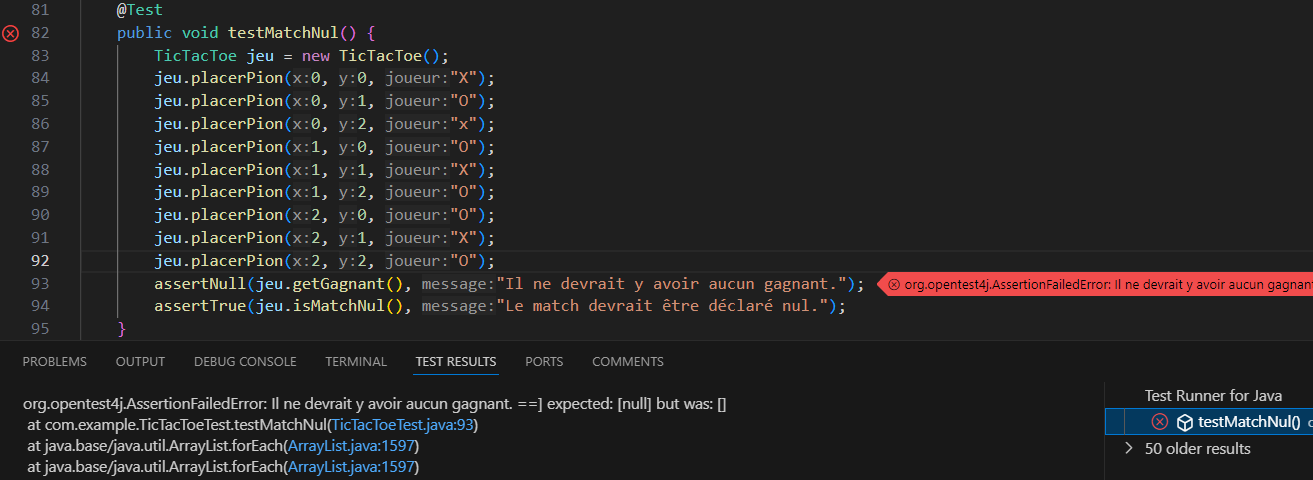
-On réexécute le test afin de vérifier le succès



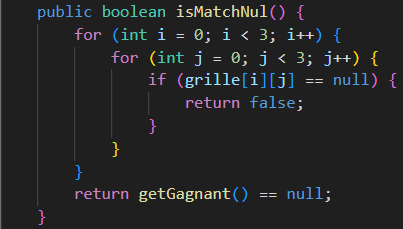
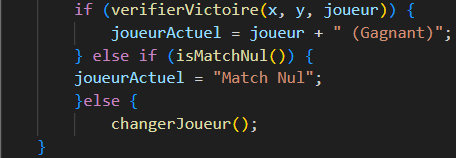
**Exigence 4**

Etape1 : Vérifier que le match est nul lorsque toutes les cases sont remplies sans qu'aucun joueur n'ait gagné

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



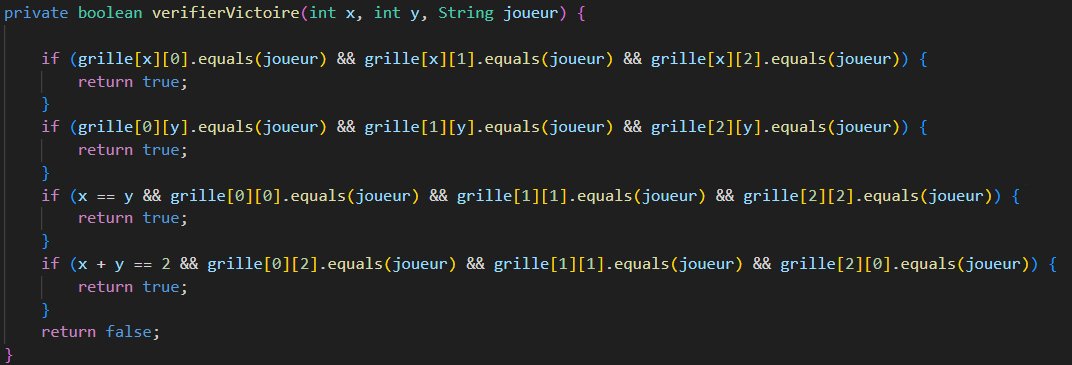
-On implémente le code nécessaire pour faire passer le test



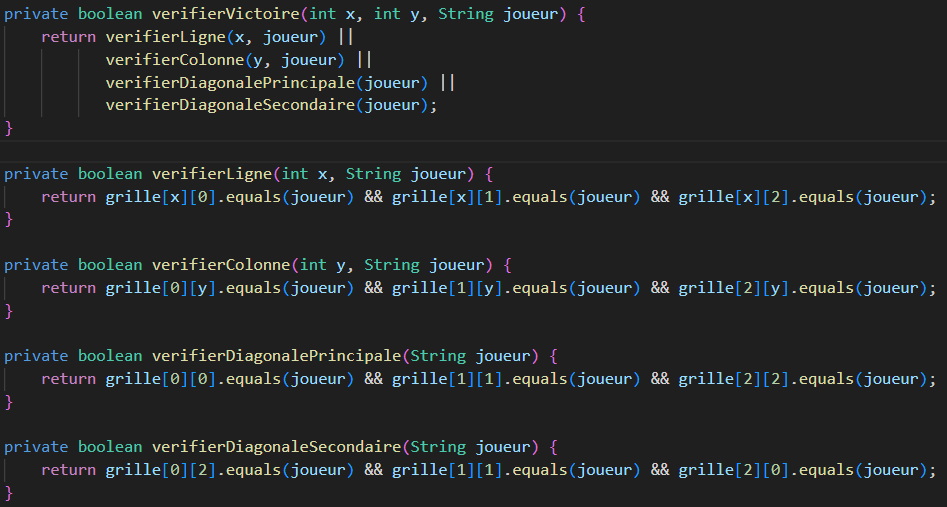
-On réexécute le test afin de vérifier le succès



**POUR LE REFACTORING DU CODE :**

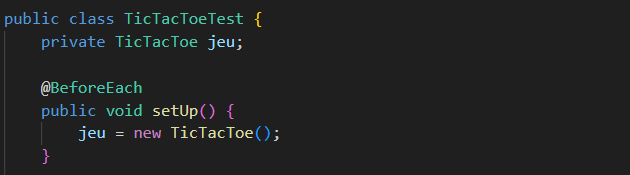


-dans la méthode verifierVictoire on peut extraire la vérification des lignes, colonnes et diagonales comme suit :

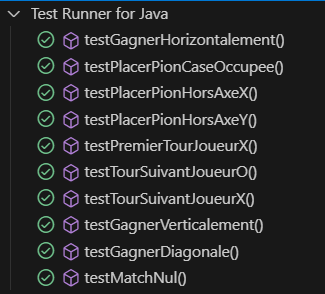


-Pour la classe test on peut aussi déclarer jeu comme une variable d'instance de la classe de test pour qu’elle

Soit créer avant l’exécution de chaque test

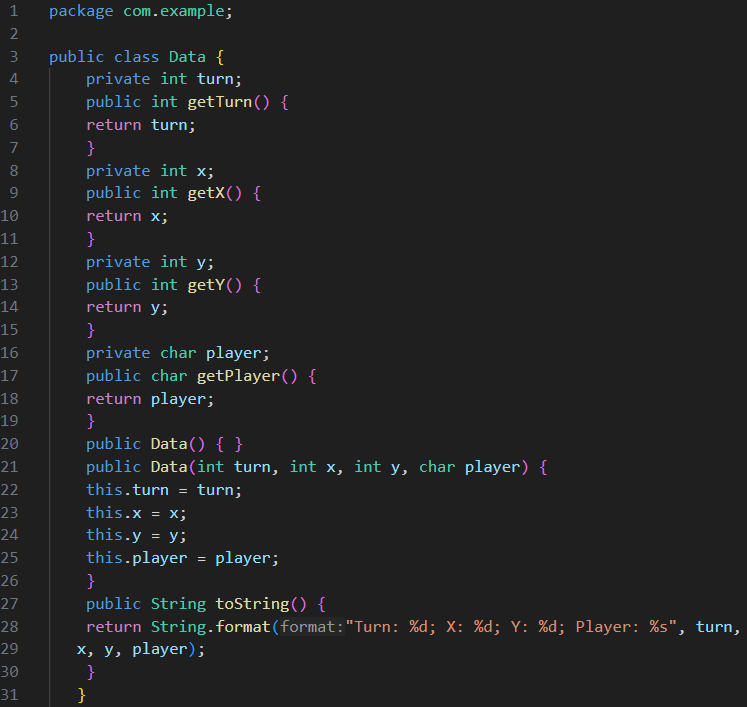


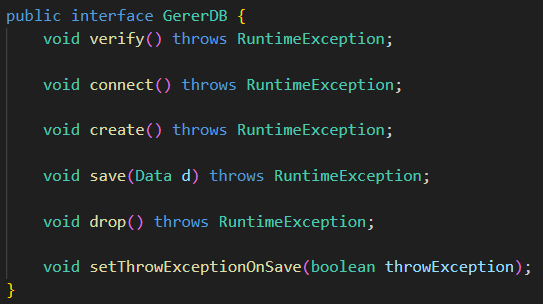
-enfin on réexécute tous les tests pour vérifier qu’il sont tous passer avec succès

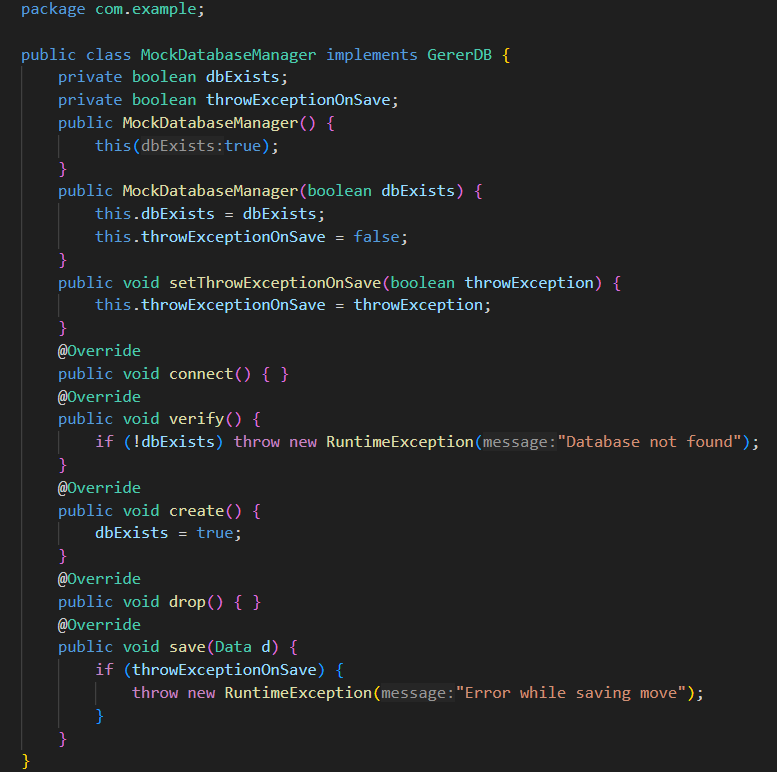


Partie 2

-Tout d’abord on doit créer la classe Data, l’interface GererBD et une classe MockDatabaseManager qui va implémenter l’interface comme suit :



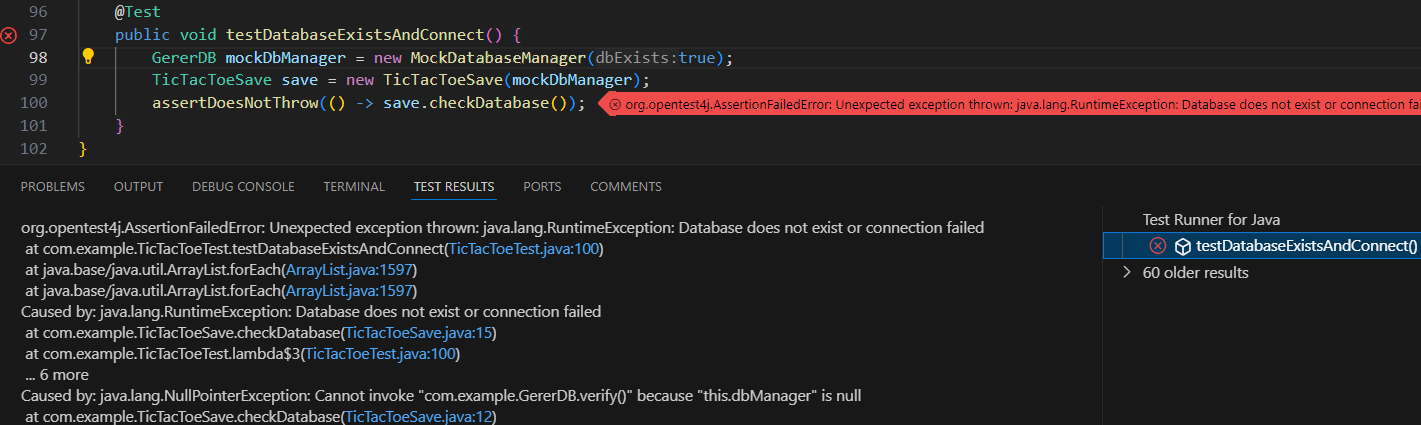




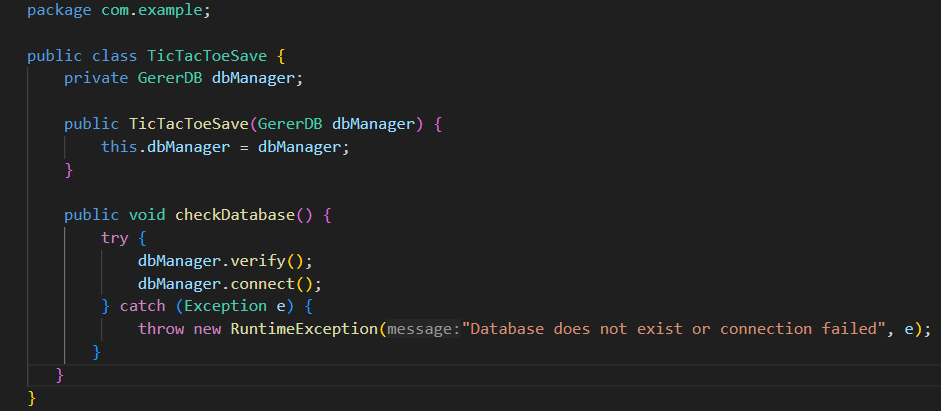
**Exigence 1 :** **Enregistrer un mouvement**

Test 1 : Vérifier si la base de données tic-tac-toe existe et se connecter

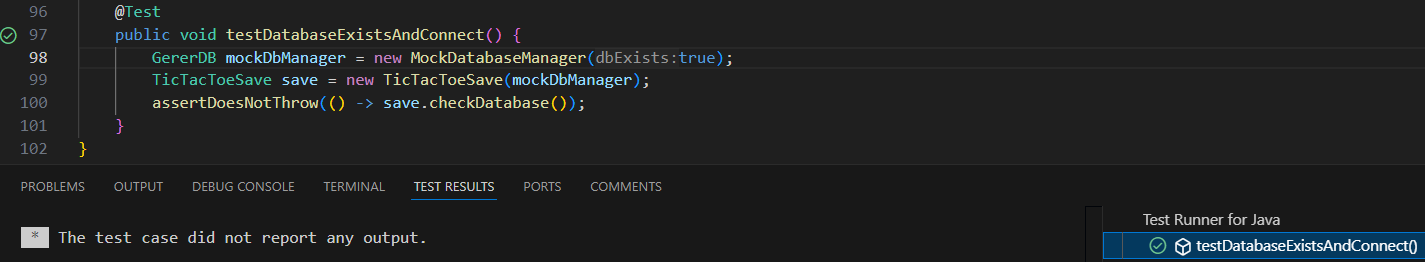
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

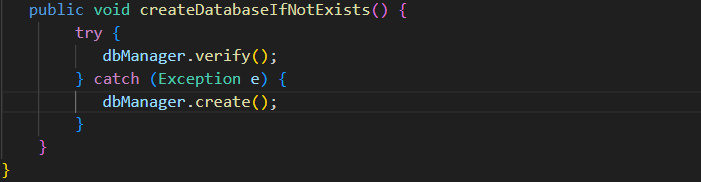


-On réexécute le test afin de vérifier le succès

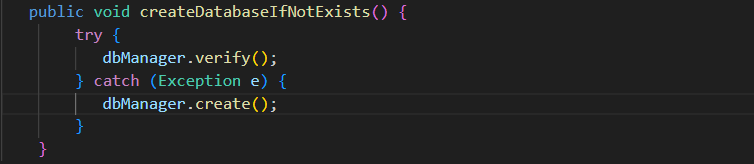


Test 2 : Vérifier si la base de données tic-tac-toe n’existe pas, elle est créée

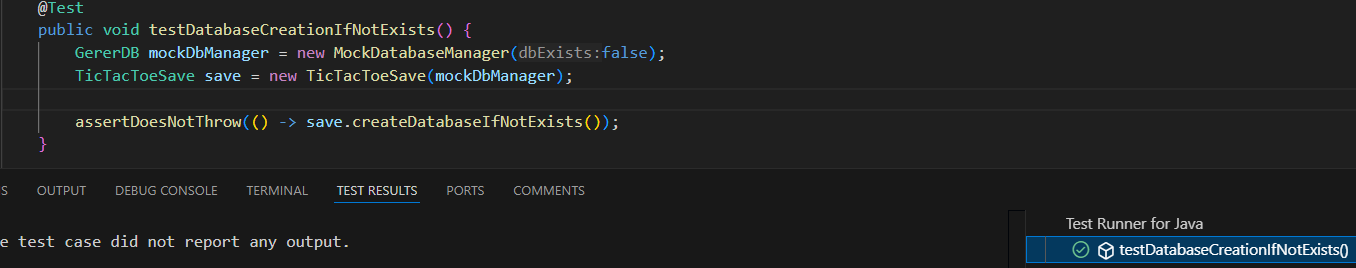
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test



-On réexécute le test afin de vérifier le succès

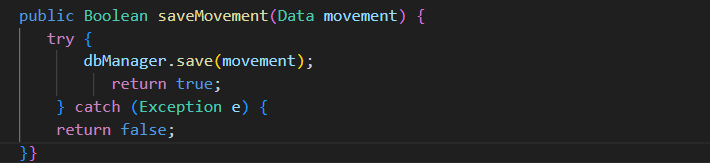


Test 3 : Enregistrer les données d’un mouvement dans la BD

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

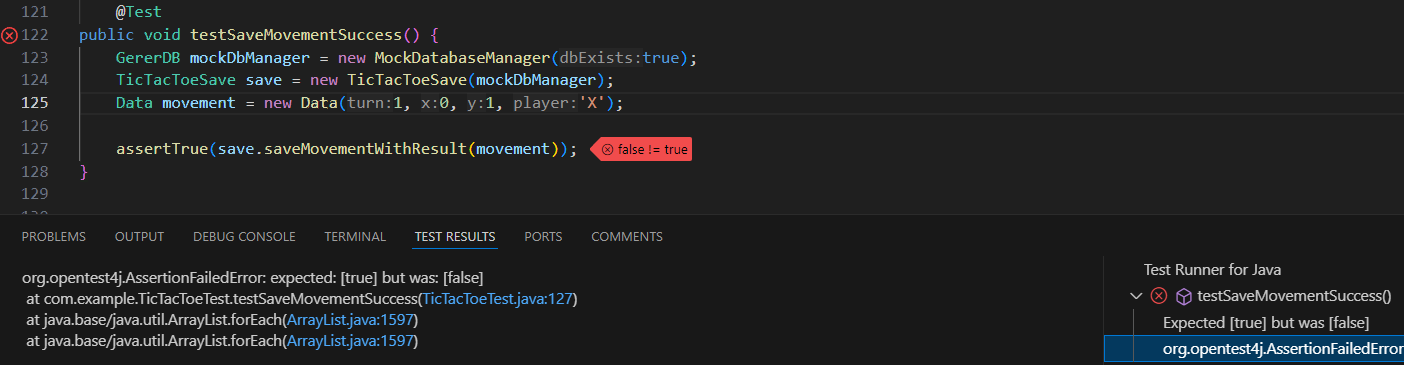


-On réexécute le test afin de vérifier le succès

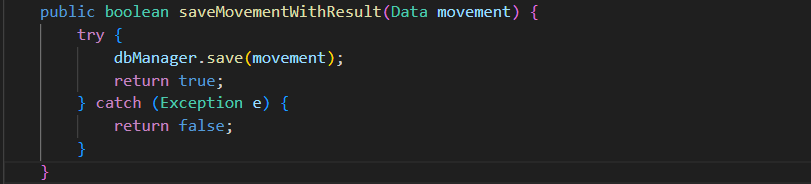


Test 4 : Retourner true si l’enregistrement du mouvement est réussi

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

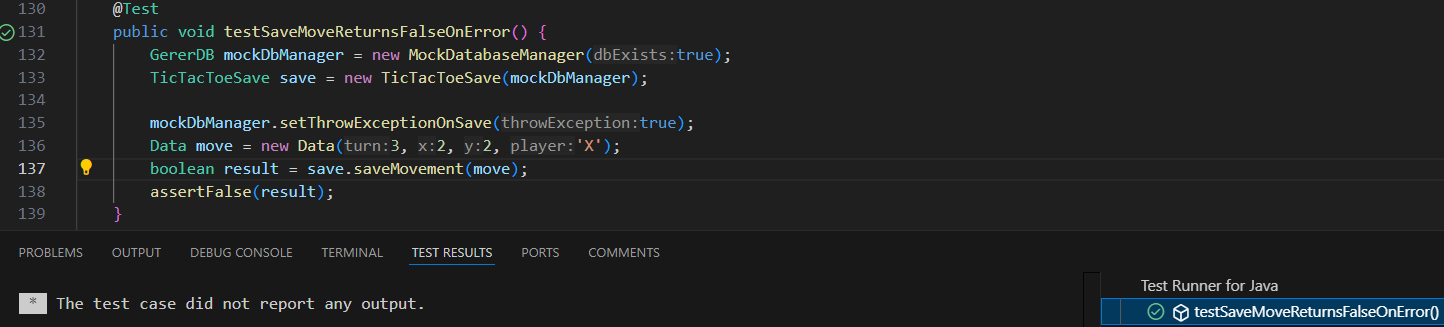


-On réexécute le test afin de vérifier le succès



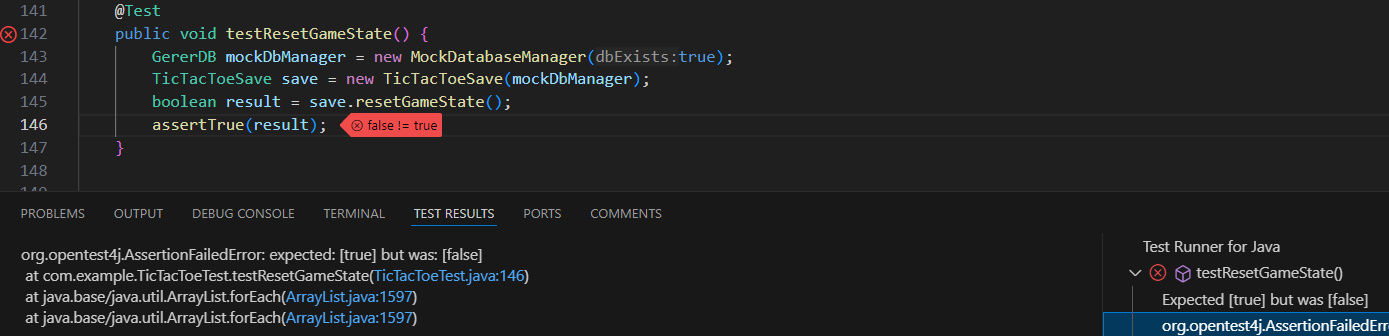
Test 5 : Retourner false si une erreur se produit lors de l’enregistrement

-On commence par écrire le test et l’exécuter ce qui nous indique son succès car il utilise la même méthode du test 4

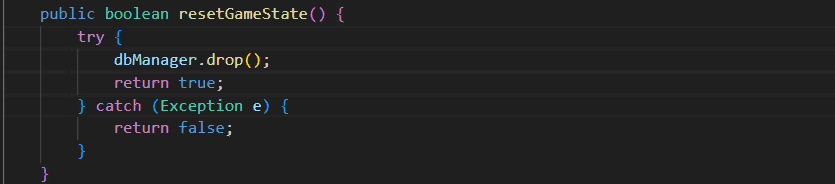


Test 6 : Initialiser l’état du jeu entre deux parties

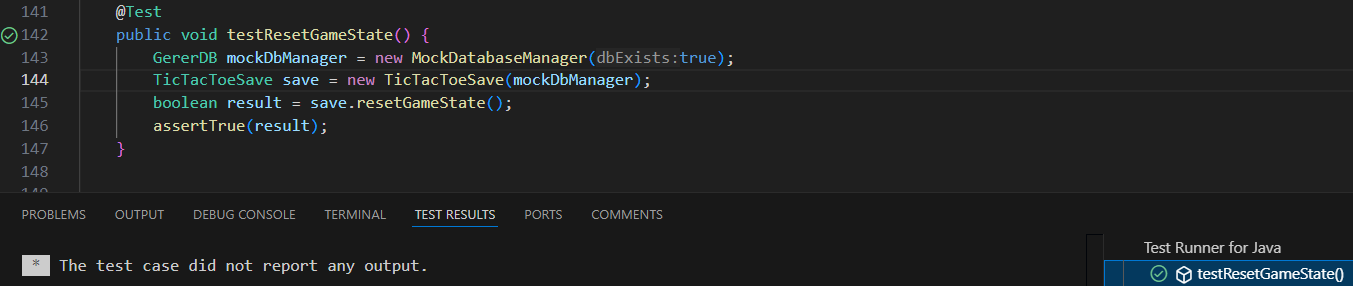
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



-On implémente le code nécessaire pour faire passer le test

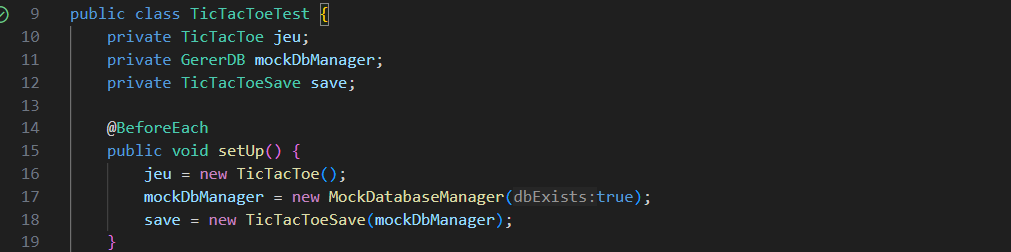


-On réexécute le test afin de vérifier le succès



**POUR LE REFACTORING DU CODE :**

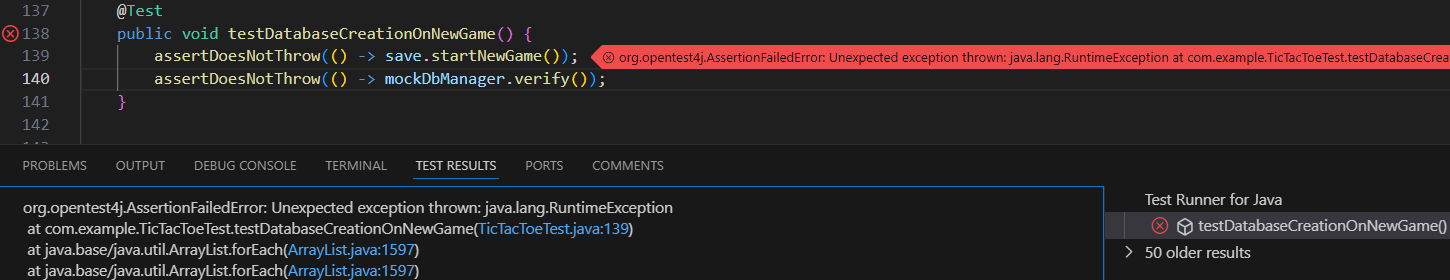
-Pour la classe test on peut aussi déclarer mockDbManager et save comme deux variable d'instance de la classe de test pour qu’elles soient créées avant l’exécution de chaque test



**Exigence 2 :** **Enregistrer chaque tour**

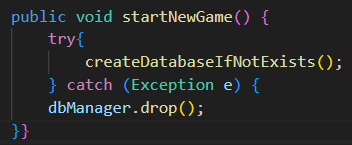
Test 1 : Vérifier la création de la base de données lors de lancement d’une nouvelle partie

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test

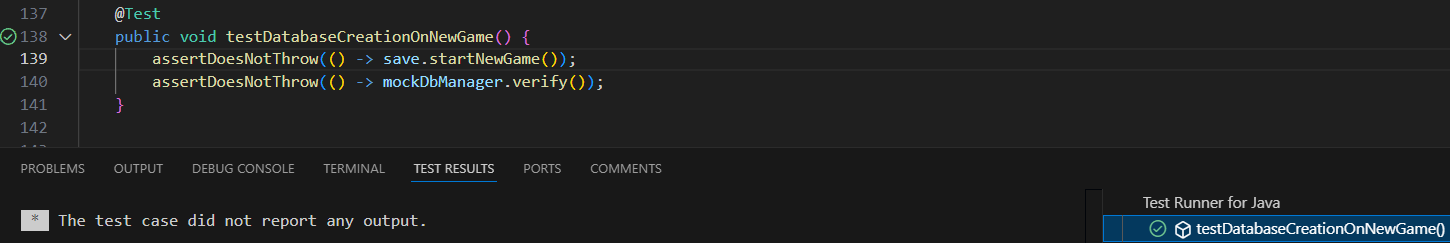


-On implémente le code nécessaire pour faire passer le test

Dans TicTacToeSave



-On réexécute le test afin de vérifier le succès



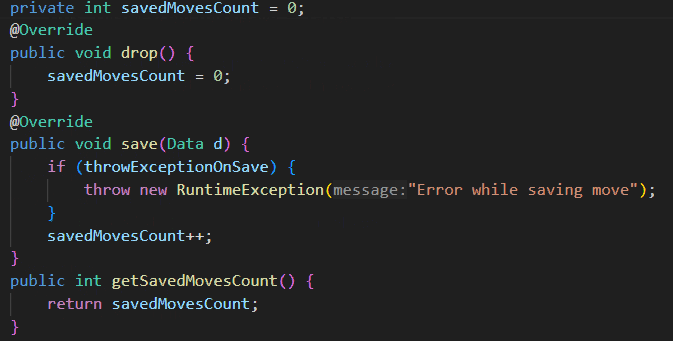
Test 2 : À chaque coup du jeu, les données sont sauvegardées dans la base de données

-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test

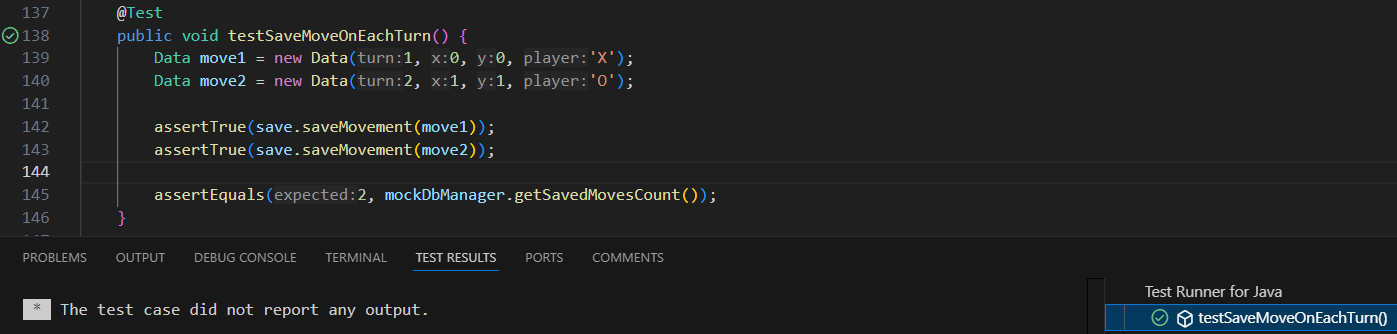


-On implémente le code nécessaire pour faire passer le test

dans MockDatabaseManager

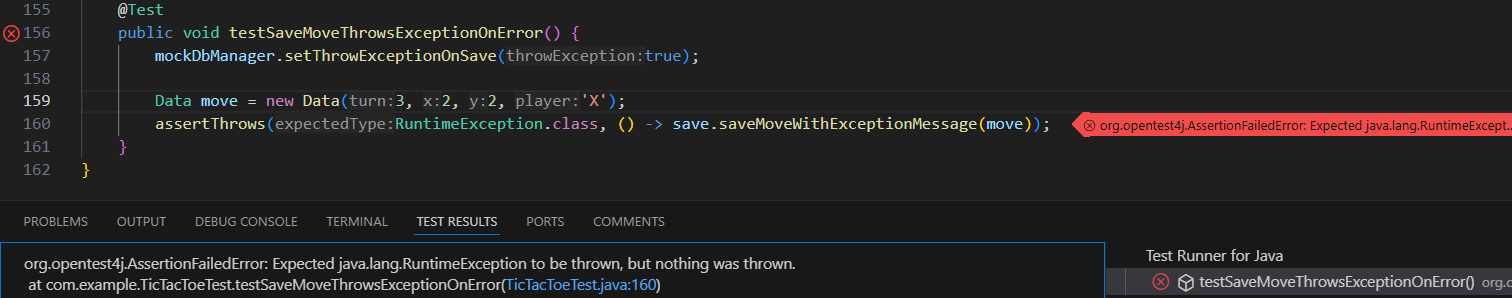


-On réexécute le test afin de vérifier le succès

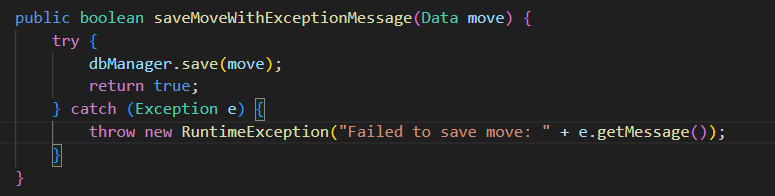


Test 3 : En cas de problème de d’enregistrement des données, générer une exception

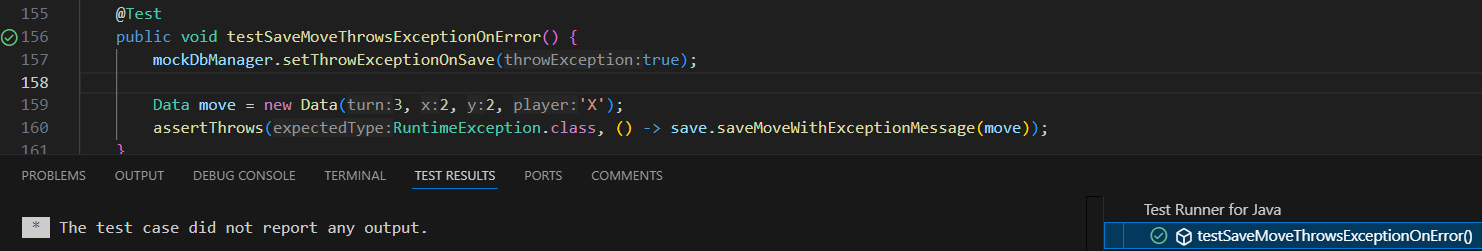
-On commence par écrire le test et l’exécuter ce qui nous indique l’échoue du test



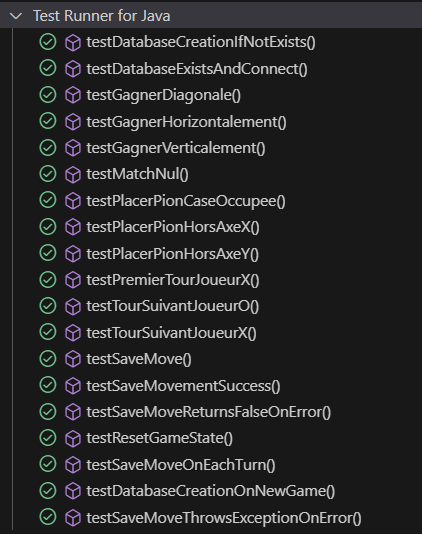
-On implémente le code nécessaire pour faire passer le test



-On réexécute le test afin de vérifier le succès



Enfin on exécute tous les tests :



**POUR LE REFACTORING DU CODE :**

**On enlève la méthode saveMovementWithResult car la méthode saveMovement fait la même chose**