# Diagramme de cas d'utilisation

```Mermaid
flowchart LR
    Joueur[🧍‍♂️ Joueur] --- Puissance([Jouer au jeu de Puissance 4])
```

# Scénario

## UC : Jouer au Puissance 4

Périmètre : Le jeu

Niveau : But utilisateur

Acteur principal : Joueur

Précondition : Aucune

Postconditions : Aucune

### Scénario nominal

1. L'utilisateur lance le programme
2. Le système demande le nom du joueur 1
3. Le joueur 1 répond
4. Le système enregistre le nom du joueur 1
5. Le système demande au joueur 2
6. Le joueur 2 répond
7. Le système enregistre le nom du joueur 2
8. Le système affiche le plateau ainsi que le nom du joueur qui doit jouer
9. Le joueur choisi dans quelle colonne il souhaite mettre un jeton
10. Le système vérifie si la partie est gagné
11. Le système affiche le vainqueur et demande si l'utilisateur souhaite refaire une partie ("y" or "n")
12. L'utilisateur répond "y" ou "n"
13. Le système retourne au point 8 du scénario nominal et incrémente le nombre de partie

### Extensions

3.a L'utilisateur répond à la question avec une chaîne vide

1. Retour au point 2 du scénario nominal

6.a L'utilisateur répond à la question avec une chaîne vide

1. Retour au point 5 du scénario nominal

9.a L'utilisateur choisit un numéro de colonne

1. Le système affiche un message d'erreur
2. Retour au point 8

9.b L'utilisateur choisit un numéro de colonne valide, mais la colonne est pleine

1. Le système affiche un message d'erreur
2. Retour au point 8

10.a La partie n'est pas finie

1. Le système change le joueur qui doit jouer
2. Retour au point 8 du scénario nominal

12.a l'utilisateur répond autre chose que 'y' ou 'n'

1. Le système affiche un message d'erreur
2. Retour au point 11 du scénario nominal

12.b L'utilisateur répond 'n'

1. Le système affiche le gagnant et le nombre de victoire du gagnant
   1. - Il n'y a pas de gagnant
      1. Le système affiche ex aequo et le nombre de victoire des joueurs (il est identique pour les deux)
      2. Retour au point 2 de l'extension 12.b
2. Fin du programme

### Extensions

4.a L'utilisateur répond à la question avec une chaîne vide

1. Retour au point 2 du scénario nominal

7.a L'utilisateur répond à la question avec une chaîne vide

1. Retour au point 5 du scénario nominal

10.a L'utilisateur choisit un numéro de colonne

1. Le système affiche un message d'erreur
2. Retour au point 8 du scénario nominal

10.b L'utilisateur choisit un numéro de colonne valide, mais la colonne est pleine

1. Le système affiche un message d'erreur
2. Retour au point 8 du scénario nominal

10.a La partie n'est pas finie

1. Le système change le joueur qui doit jouer
2. Retour au point 8 du scénario nominal

13.a l'utilisateur répond autre chose que 'y' ou 'n'

1. Le système affiche un message d'erreur
2. Retour au point 11 du scénario nominal

13.b L'utilisateur répond 'n'

1. Le système affiche le gagnant et le nombre de victoire du gagnant
   1. - Il n'y a pas de gagnant
      1. Le système affiche ex aequo et le nombre de victoire des joueurs (il est identique pour les deux)
      2. Retour au point 2 de l'extension 13.b
2. Fin du programme

# Modèle de Conception

classDiagram
    class main {

    }

    class Ihm {
        - Scanner scanner

        + Ihm()
        + demanderNomJoueur(int numJoueur) String
        + demanderJouerEncore() bool
        + afficherPlateau(String plateauString) void
        + afficherVictoire(String nomJoueur, int nbVictoires, int nbParties) void
        + afficherErreur(String message) void
        + demanderCoupP4(String nomJoueur) byte
    }

    class ControleurP4 {
        - int numeroJoueurCourant
        - int nbParties
        
        + ControleurP4(Ihm ihm)
        + jouer () void
        - commencerPartie() void
        - toursDeJeu() void
        - tourSuivant() void
        - finPartie() void
        + getNumeroJoueurCourant() int
        + getJoueurCourant() Joueur
        + getNomJoueurCourant() String
    }

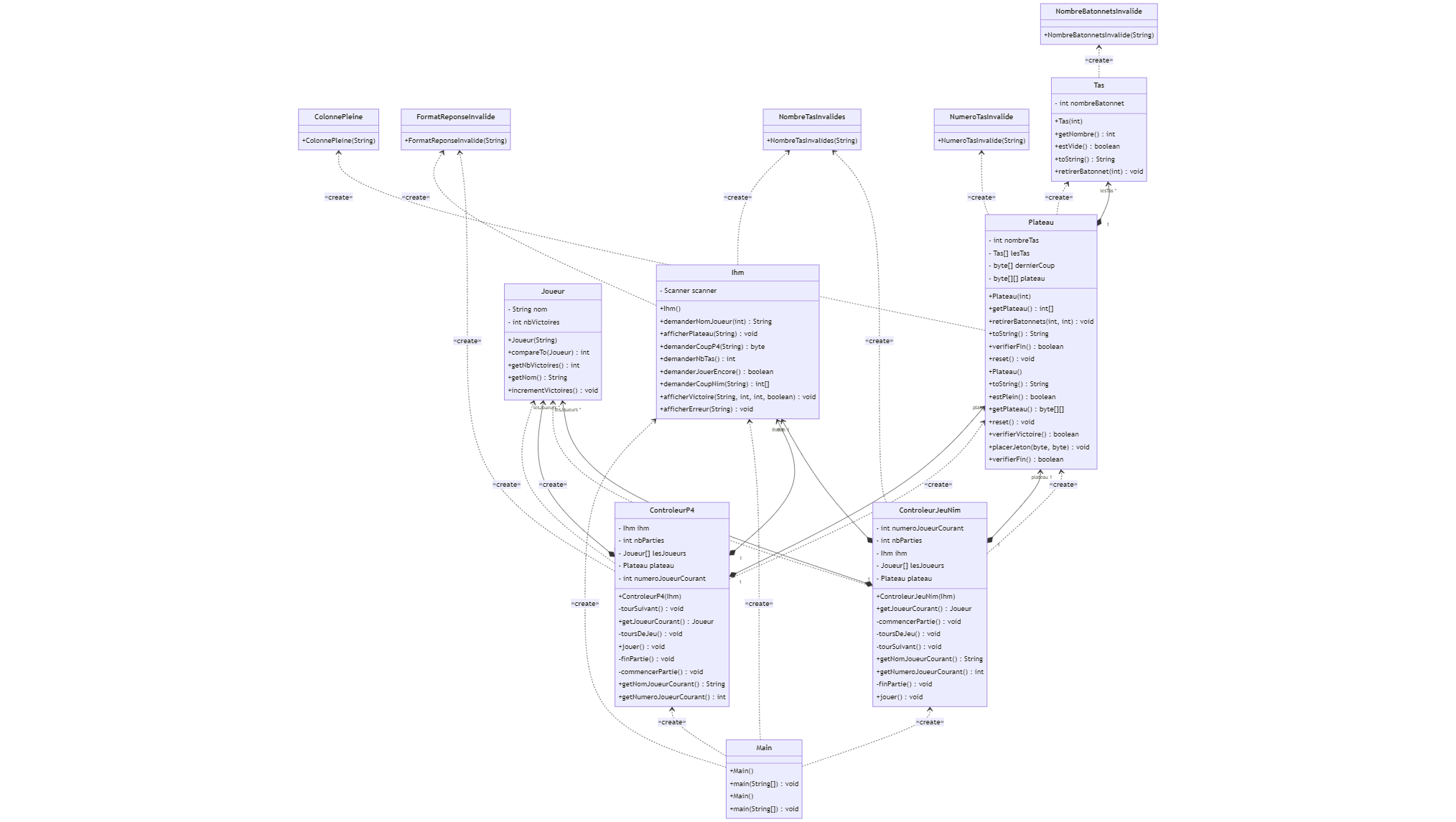
    class Joueur {
        - String nom
        - int nbVictoires

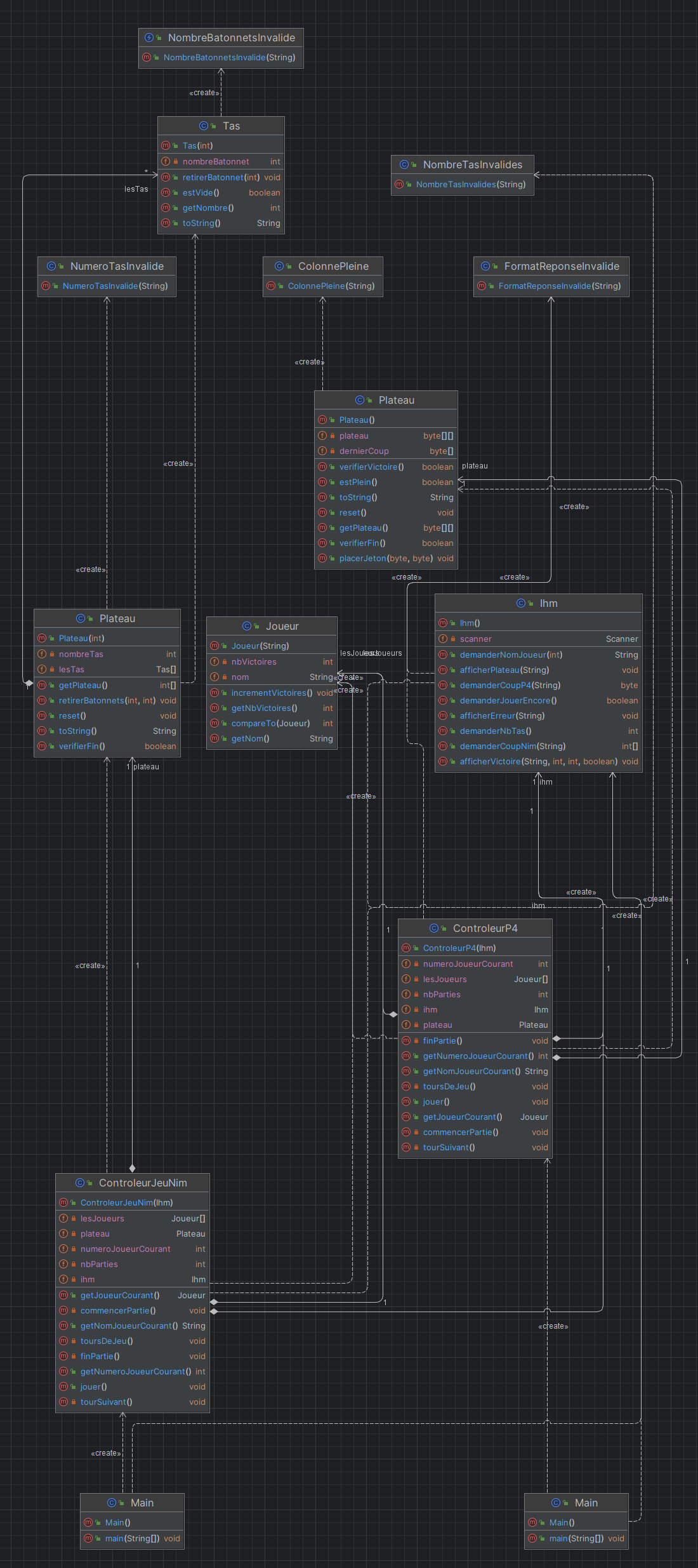
        + Joueur(String nom)
        + getNom() String
        + getNbVictoires() int
        + incrementVictoires() void
        + compareTo(Joueur autreJoueur) int
    }

    class Plateau {
        - byte[][] plateau
        - byte[] dernierCoup
                
        + Plateau ()
        + reset() void
        + verifierFin() bool
        + verifierVictoire() bool
        + estPlein() bool
        + getPlateau() byte[][]
        + placerJeton(byte colonne, byte joueur) void
        + toString() String
    }

    ControleurP4 --> "1" Plateau
    ControleurP4 --> "1" Ihm
    ControleurP4 --> "2" Joueur : lesJoueurs

Version intelij





# Répartition des tâches

14/03/2024 (Publication du sujet)

Nathan, Dany, Agathe : review du sujet et brainstorming (check de la victoire etc)

16/03/2024

Nathan : Diagramme de classe, Cas d'utilisation

Dany : review du Diagramme

Dany : Développement

Agathe : Développement

Dany : Review, documentation, fix

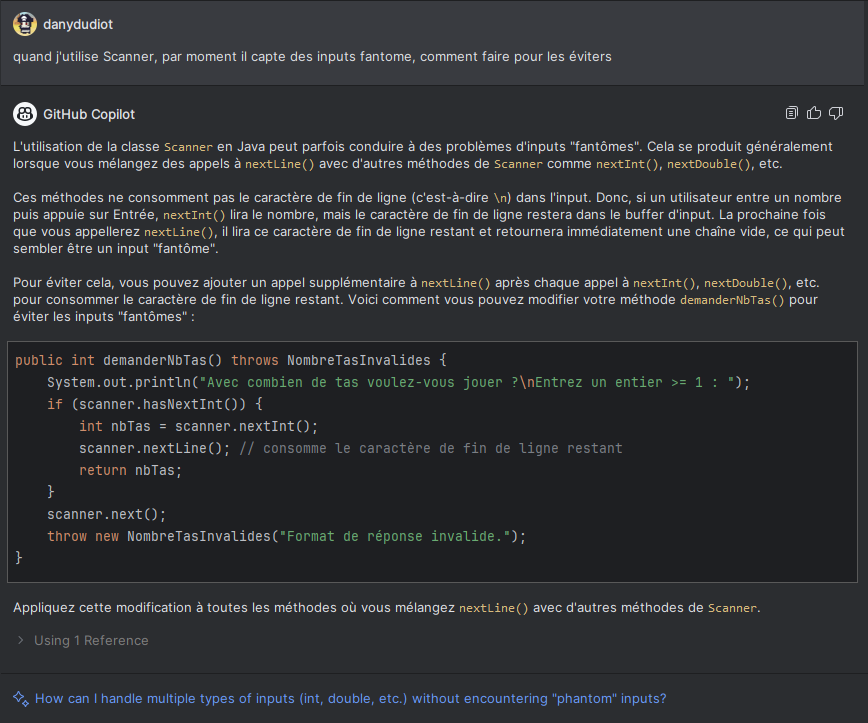
Dany : mise en place de test avec JUnit

# Utilisation de l'IA

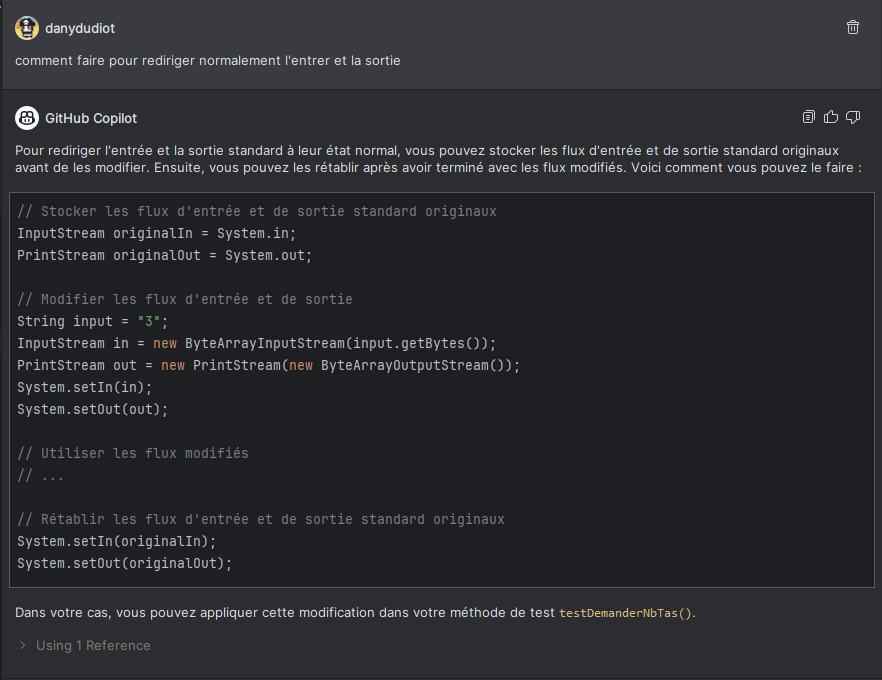
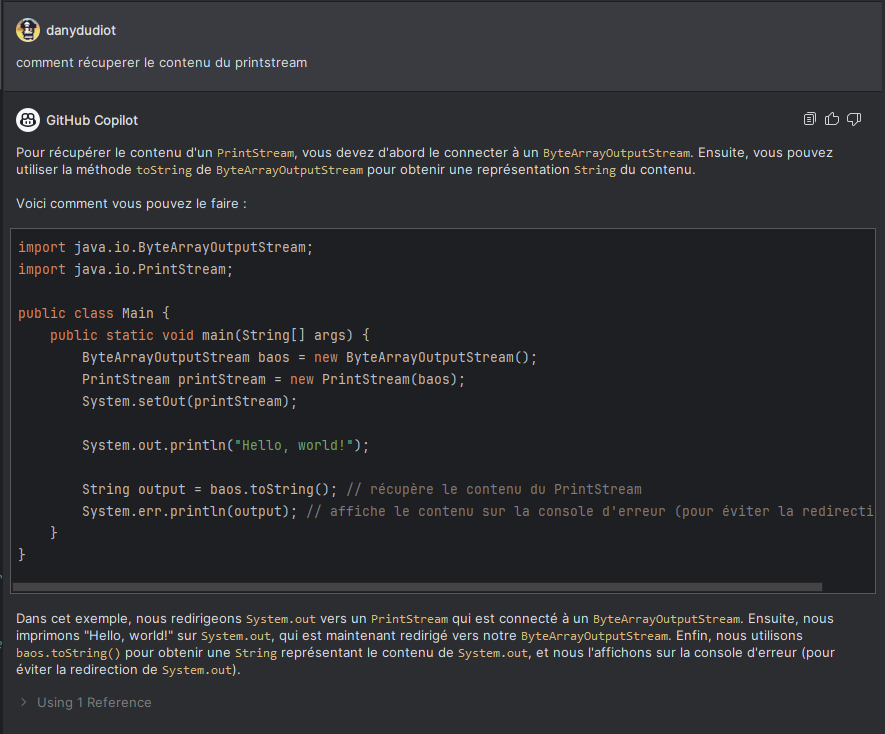
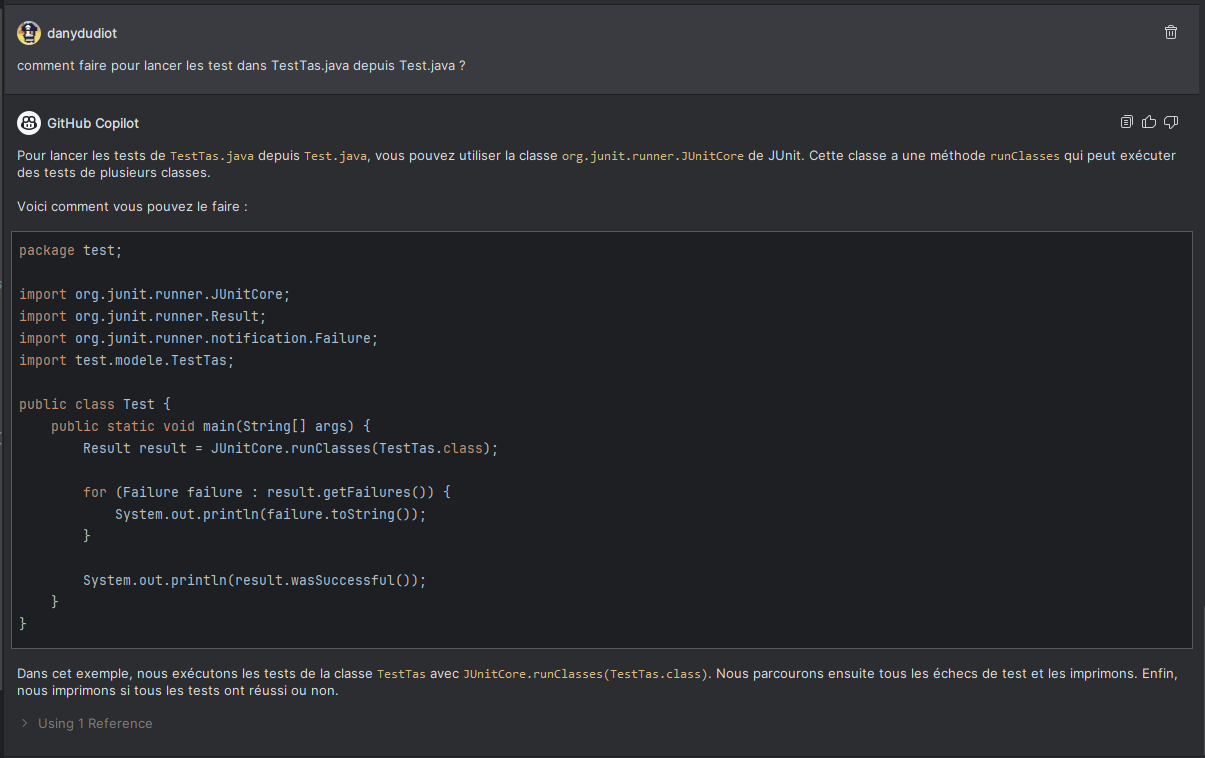
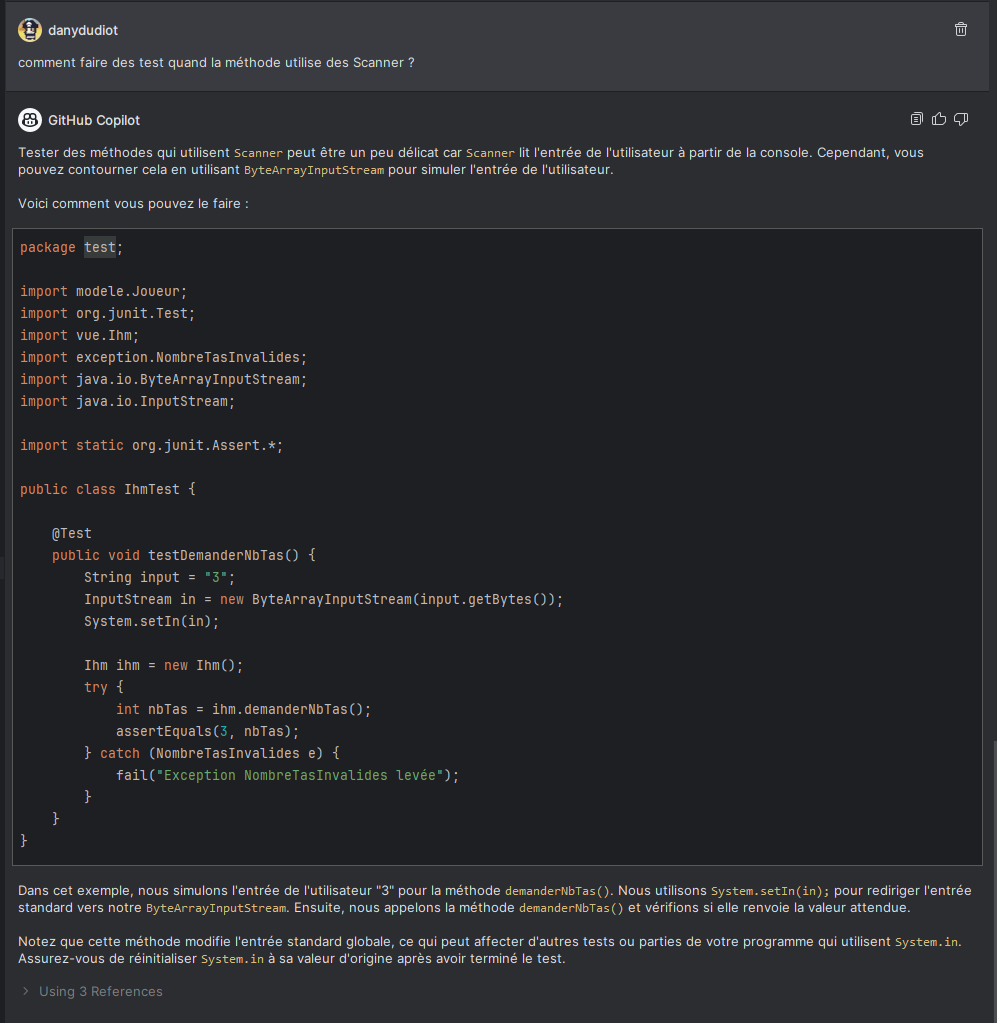
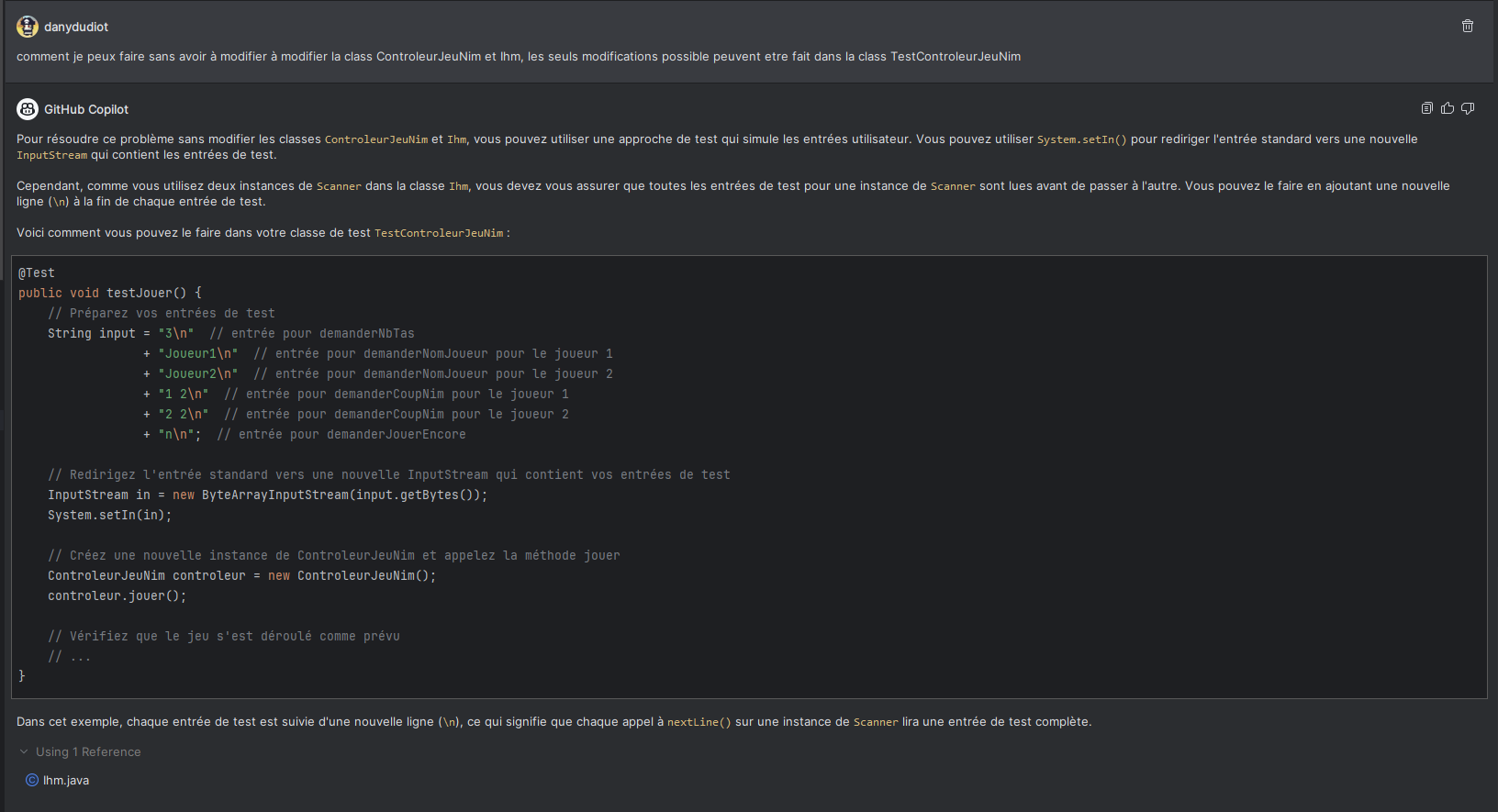
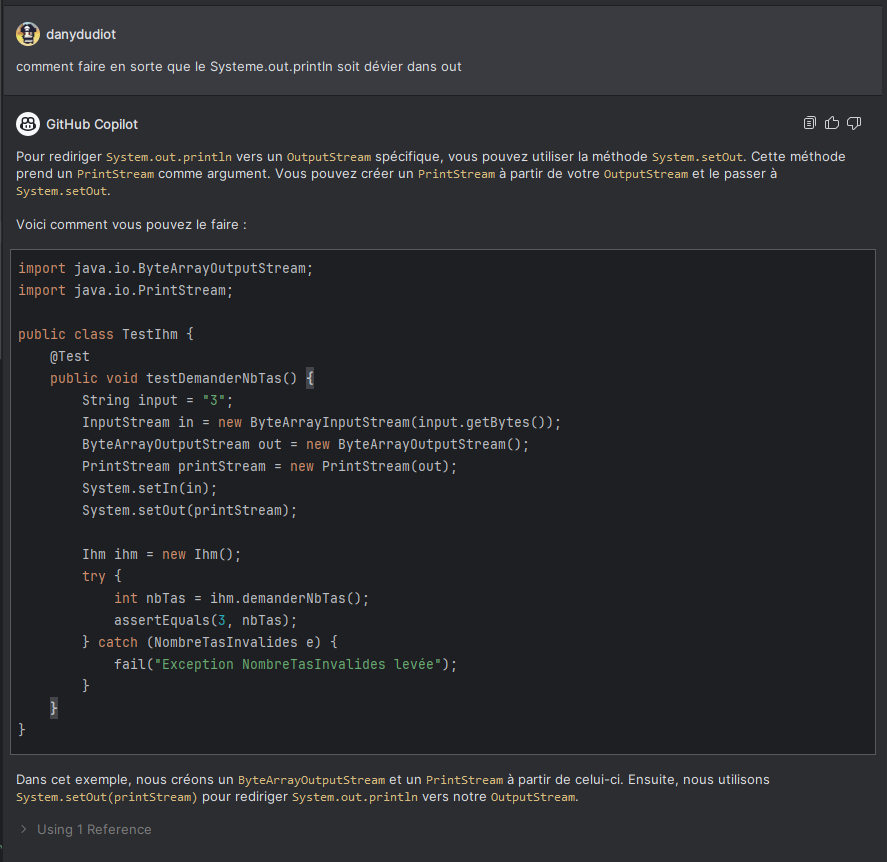
Nathan : Aucun Usage

Agathe : Aucun Usage

Dany : Copilot (screen dans img)



Utilisation de l’IA pour les tests

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

