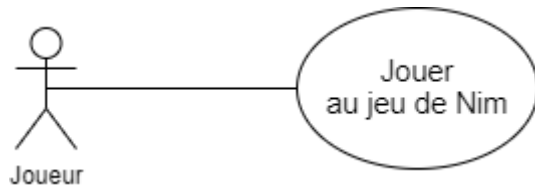


Itération 1

Diagramme de cas d'utilisation



Scénario

UC : Jouer au jeu de Nim

Périmètre : Le jeu

Niveau : But utilisateur

Acteur principal : Joueur

Pré-condition : Aucune

Post-conditions : Aucune

Scénario nominal

1. L'utilisateur lance le programme
2. Le système demande à l'utilisateur le nombre de tas qu'il souhaite
3. L'utilisateur répond à la question
4. Le système enregistre le nombre de tas
5. Le système demande le nom du joueur 1
6. Le joueur 1 répond
7. Le système enregistre le nom du joueur 1
8. Le système demande au joueur 2
9. Le joueur 2 répond
10. Le système enregistre le nom du joueur 2
11. Le système affiche les tas ainsi que le nom du joueur qui doit jouer
12. Le joueur choisi dans quel tas il souhaite prendre un certain nombre d'allumette
13. Le système vérifie si la partie est gagnée
14. Le système affiche le vainqueur et demande si l'utilisateur souhaite refaire une partie ("y" or "n")
15. L'utilisateur répond "y" ou "n"
16. Le système retourne au point 11 du scénario nominal et incrémente le classement entre les deux joueurs

Extensions

4. a Le joueur répond autre chose ou un nombre inférieur à 1

1. Le système affiche un message d'erreur "le nombre de tas doit être un entier supérieur à 1"
2. Retour au point 2 du scénario nominal

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

1. Le système affiche un message d'erreur "Le nom ne peut pas être vide"
2. Retour au point 5 du scénario nominal

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

1. Le système affiche un message d'erreur "Le nom ne peut pas être vide"
2. Retour au point 8 du scénario nominal

13. a Le joueur n'a pas entré une réponse au format « m n »

1. Le système affiche un message d'erreur "Le format de la réponse doit être « m n »"
2. Retour au point 11 du scénario nominal avec le même joueur

13. b Le joueur a demandé un tas qui n'existe pas

1. Le système affiche un message d'erreur "Tas inexistant"
2. Retour au point 11 du scénario nominal avec le même joueur

13. c Le joueur veut prendre un nombre d'allumette supérieur au nombre contenue dans le tas

1. Le système affiche un message d'erreur "Pas assez d'allumette dans ce tas"
2. Retour au point 11 du scénario nominal avec le même joueur

13. d La partie n'est pas fini

1. Retour au point 11 du scénario nominal avec l'autre joueur

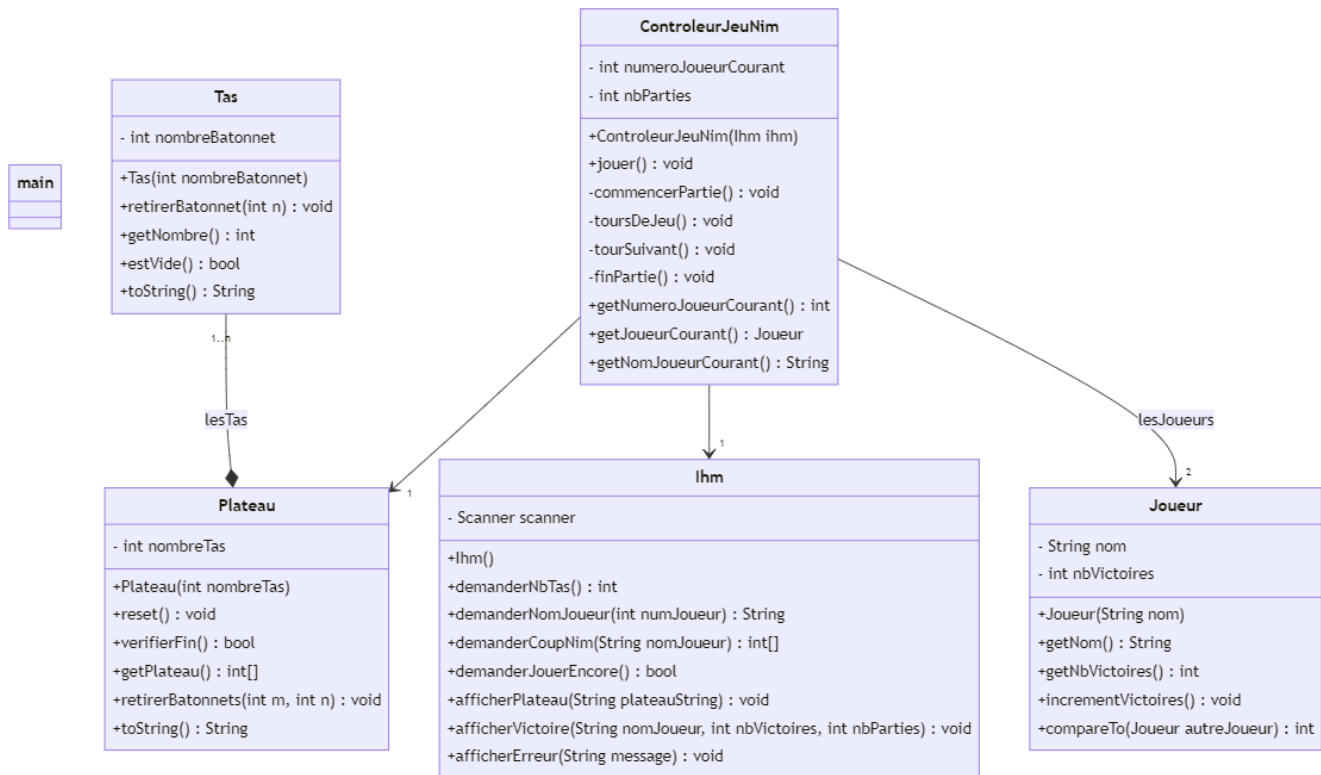
16. a L'utilisateur a répondu autre chose que y ou n

1. Le système affiche un message d'erreur "Répondre avec "y" ou "n""
2. Retour au point 14 du scénario nominal

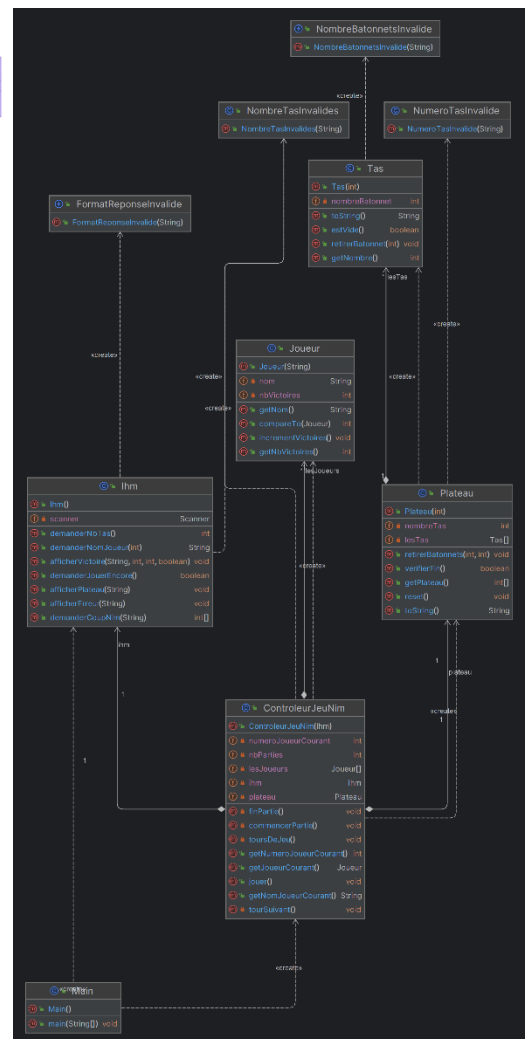
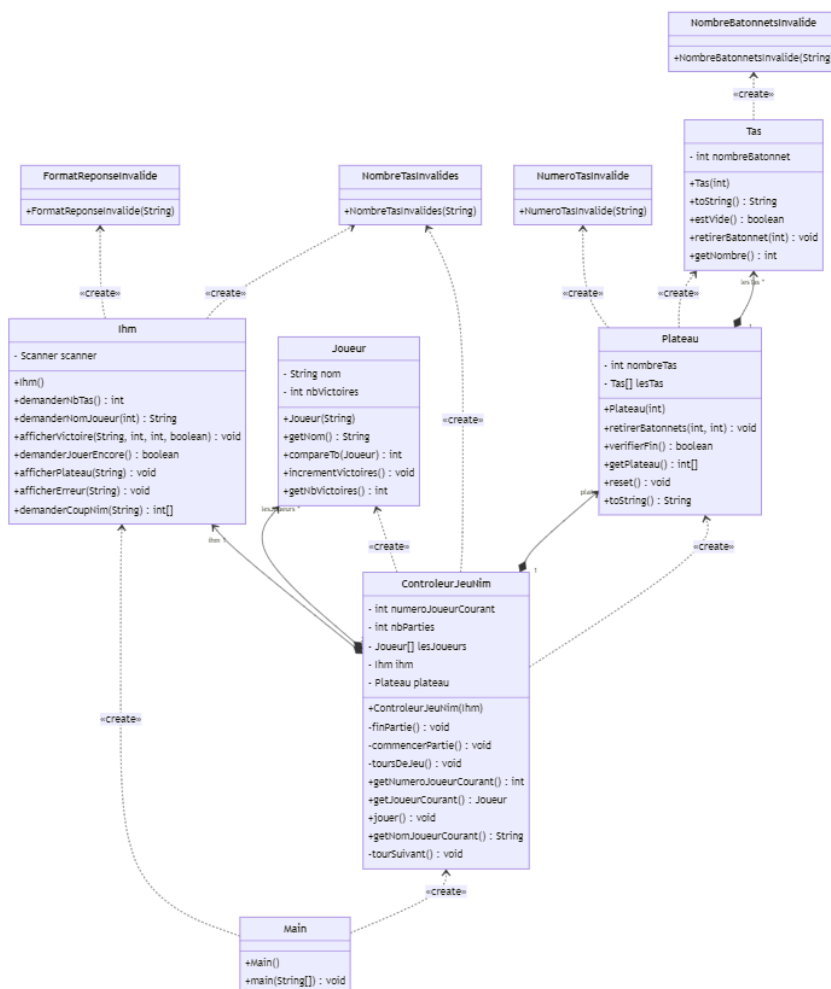
16. b L'utilisateur a choisi d'arrêter de jouer

1. Le système affiche le classement entre les deux joueurs
2. Le système attend une action de l'utilisateur pour arrêter le programme

Diagrammes de Classes



Version IntelliJ



Répartition des tâches

Publication du sujet

Dany : Premier jet du diagramme des UC et scénario

Nathan : Review et premier jet du diagramme de classe

Premier CM avec des questions (19/02/2024)

Dany : Review et amélioration du diagramme de classe

Agathe, Nathan, Dany : Review des diagrammes

Agathe : Finalisation du diagramme de Classes

Nathan : Développement du code

Nathan et Dany : Debug

Dany : Essaie de création d'une UI mais pas réussi

Début des vacances

Fin des vacances

Dany : Petit fix avant le TP

Dany, Nathan, Agathe : Fix durant le TP


Utilisation de l'IA

Nathan : Génération de Javadoc sur les fonctions simples (ex set-get, afficherErreur etc.)


(<https://chat.openai.com/share/2db9f292-a089-4935-a584-acebe06836a9>)

Dany : usage de Github Copilot pour la découverte de JavaFX

Quelque exemple :

 danydudiot

peut tu m'expliquer comment fonctionne jfx

 GitHub Copilot

JavaFX est une bibliothèque logicielle pour Java qui est utilisée pour créer des applications de bureau qui peuvent fonctionner sur une variété de dispositifs. JavaFX est destiné à remplacer Swing en tant que bibliothèque standard de GUI pour Java SE.

Voici comment cela fonctionne généralement :

- Création de la scène et de la fenêtre (Stage)** : En JavaFX, la classe `Stage` représente la fenêtre de l'application. Une application JavaFX peut avoir plusieurs fenêtres, mais il doit y avoir une fenêtre principale à partir de laquelle les autres fenêtres sont lancées.
- Création de conteneurs (Containers)** : Les conteneurs sont utilisés pour contenir des composants, tels que des boutons, des champs de texte, des étiquettes, etc. Les conteneurs peuvent être imbriqués pour créer des interfaces utilisateur complexes.
- Ajout de composants (Nodes)** : Les composants sont ajoutés aux conteneurs. Ces composants peuvent être des éléments d'interface utilisateur tels que des boutons, des images, des tableaux, etc.
- Gestion des événements** : JavaFX fournit une API riche pour gérer les événements. Par exemple, vous pouvez définir ce qui se passe lorsqu'un utilisateur clique sur un bouton, déplace la souris, appuie sur une touche du clavier, etc.
- Stylisation avec CSS** : JavaFX supporte le CSS pour styliser les applications, ce qui permet de séparer le design de la logique de l'application.

Voici un exemple de code simple pour une application JavaFX :


```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.stage.Stage;

public class HelloWorld extends Application {
    @Override
    public void start(Stage stage) {
        Label helloWorldLabel = new Label("Hello, World!");
        Scene scene = new Scene(helloWorldLabel, 200, 100);
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Dans cet exemple, nous créons une fenêtre avec un label qui dit "Hello, World!".

comment retourner null


 GitHub Copilot

In Java, you cannot return `null` for a method that is supposed to return a primitive type like `int`. If you want to return `null`, you should change the return type of your method to `Integer` which is the wrapper class for `int`. Here's how you can do it:


```
private Integer parser (TextField reponse){
    String rep = reponse.getText();
    try{
        int nbTas = Integer.parseInt(rep);
        if (nbTas <= 0){
            demanderNbTas();
        }
        else{
            return nbTas;
        }
    }
    catch (Exception e){
        System.out.println("erreur dans le parsing");
    }
    return null;
}
```

In this code, I changed the return type of the `parser` method to `Integer` and returned `null` in the catch block.

> Using 1 Reference

 danydudiot

comment avoir javafx

 GitHub Copilot

Pour obtenir JavaFX, vous pouvez le télécharger à partir du site officiel de Gluon. Voici les étapes à suivre :

- Allez sur le site de Gluon à l'adresse <https://gluonhq.com/products/javafx/>.
- Cliquez sur le bouton "Download" sous "JavaFX SDK".
- Choisissez la version qui correspond à votre système d'exploitation et à votre version de Java.
- Une fois le téléchargement terminé, extrayez le fichier zip dans un emplacement de votre choix.
- Ajoutez le chemin du dossier `lib` dans le dossier extrait à votre variable d'environnement `PATH`.

Dans IntelliJ IDEA, vous pouvez ajouter JavaFX à votre projet en suivant ces étapes :

- Allez dans `File > Project Structure > Libraries`.
- Cliquez sur le signe plus (+) et sélectionnez Java.
- Naviguez jusqu'au dossier `lib` dans le dossier JavaFX que vous avez extrait et cliquez sur `OK`.
- Cliquez sur `Apply` puis sur `OK`.

N'oubliez pas d'ajouter les modules JavaFX nécessaires à votre fichier `module-info.java` si vous en utilisez un.

> Using 1 Reference

Agathe : Aucun usage