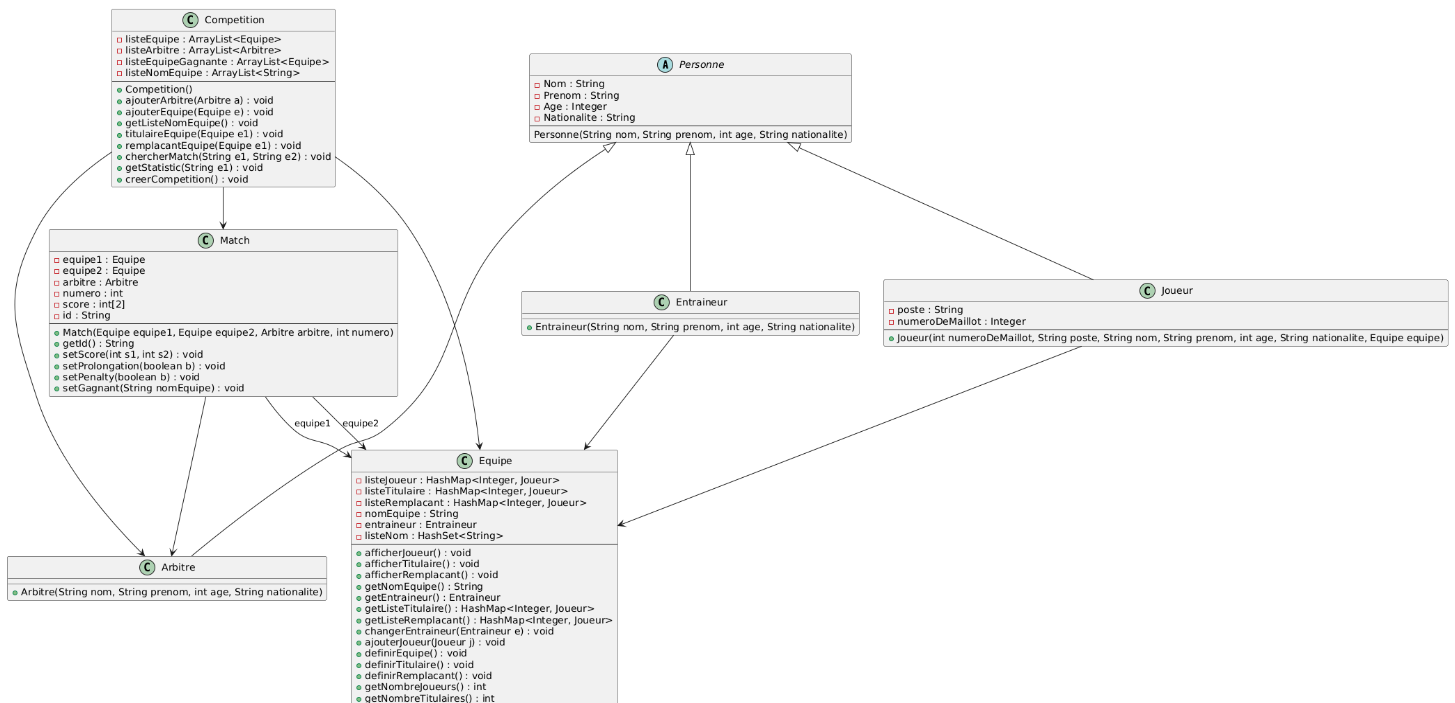
**Rapport de Projet : Gestion d'un Tournoi de Football**

**1. Présentation du Diagramme de Classes**

Le diagramme de classes fourni illustre la structure de notre projet, conçu pour organiser et gérer un tournoi de football de manière efficace. Les principales classes et leurs relations sont :

* **Personne** (classe mère) : Définissant les attributs communs (nom, prénom, âge, nationalité), elle est héritée par :
  + Arbitre : Responsable des matchs.
  + Entraineur : Responsable d'une équipe.
  + Joueur : Membre d'une équipe avec des propriétés spécifiques comme le numéro de maillot et le poste.
* **Equipe** : Gère les joueurs, les titulaires, les remplaçants, et l'entraîneur associé. Elle inclut des méthodes pour gérer la composition de l'équipe.
* **Match** : Représente un match entre deux équipes, avec un arbitre assigné, et des fonctionnalités pour gérer les scores, les prolongations, les pénalités, et le gagnant.
* **Competition** : Coordonne les équipes, arbitres et matchs. Elle permet l'organisation de matchs et la gestion globale de la compétition.



**2. Implémentation des Fonctionnalités**

**Gestion des Données avec une Base de Données**

Pour améliorer la gestion des informations et rendre le programme évolutif, nous avons intégré une base de données relationnelle. Voici le diagramme :

Une image contenant texte, capture d’écran, diagramme, Parallèle

Description générée automatiquement

Voici les principales tables conçues :

1. **Table Arbitre** :
   * Contient les informations des arbitres : ID, nom, prénom, nationalité.
2. **Table Joueur** :
   * Stocke les joueurs avec leurs propriétés : ID, nom, prénom, nationalité, numéro de maillot, poste, et l'ID de l'équipe associée.
3. **Table Entraineur** :
   * Enregistre les entraîneurs : ID, nom, prénom, nationalité.
4. **Table Equipe** :
   * Regroupe les équipes avec leurs informations : Nom d'équipe, et l'ID de l'entraîneur.
5. **Table Game** :
   * Enregistre les matchs : ID, ID des deux équipes participantes, ID de l'arbitre, numero de tours, scores des équipes, et les informations sur le gagnant.
6. **Table Jouedans** :
   * Lie chaque joueur à son équipe. Contient : ID de Equipe et ID du Joueur

**Interactions avec la Base de Données**

* Chaque classe correspond à une table dans la base de données.
* Les méthodes d'ajout, modification et suppression dans les classes sont reliées à des requêtes SQL.
* Exemple : Lorsqu'un joueur est ajouté à une équipe, cela se reflète dans la table Joueurs.

**3. Difficultés Rencontrées**

**Échec dans la Partie Graphique**

Nous avons tenté de développer une interface utilisateur graphique pour rendre le programme plus interactif. Cependant, faute de temps, cette partie n’a pas pu être achevée. Nous avons utilisé **JavaSwing** pour créer des interfaces de base, mais plusieurs défis ont ralenti notre progression :

* La gestion des événements (par exemple, les clics pour ajouter des joueurs).
* La synchronisation entre l'interface graphique et la base de données.
* Le manque de temps pour déboguer et finaliser les éléments graphiques.

**Complexité des Relations dans la Base de Données**

* La gestion des relations entre les différentes tables (comme les clés étrangères entre Equipes et Joueurs) a nécessité une planification minutieuse pour éviter les erreurs d’intégrité.

**Manque de Temps**

* L'ajout de fonctionnalités supplémentaires, comme une gestion des statistiques avancées ou une visualisation graphique des résultats, a dû être reporté faute de temps.

**4. Les bibliothèques utilisées :**

**java.io.File / java.util.ArrayList / java.util.Scanner / java.io.FileInputStream / java.io.IOException; java.util.Properties / java.sql.Connection / java.sql.DriverManager / java.sql.PreparedStatement; java.sql.ResultSet/ java.util.Collections / java.sql.SQLException;**

**5. Les IDE utilisées :**

* **Apache NetBeans IDE 23**
* **VSC (Visual Studios Code)**
* **Eclipse IDE**
* **MySQL WorkBench 8.0 CE**