Architecture Logique du Logiciel

L'architecture du projet "La Prophétie de la Mare" repose sur une approche Orientée Objet stricte, favorisant la modularité et l'extensibilité. Le système est conçu autour de la séparation des responsabilités entre le moteur de jeu, l'affichage, et les données (le monde et les entités).

Voici la présentation des principales entités et de leurs interactions :

1. `Jeu`

La classe **`Jeu`** est le point d'entrée central. Elle agit comme le contrôleur principal (Game Controller). C'est elle qui :

- Initialise la partie (création du joueur, génération de la liste des salles).
- Gère la boucle principale (`boucleDeJeu`), qui maintient le jeu actif tant que la condition de victoire ou de défaite n'est pas atteinte.
- Orchestre les tours de jeu en demandant les actions au joueur et en déclenchant les événements des salles.

2. `InterfaceUtilisateur`

Afin de ne pas mélanger la logique métier avec la gestion des entrées/sorties, la classe **`InterfaceUtilisateur`** a été isolée.

- Elle est responsable de la capture des choix du joueur (via `Scanner`).
- Elle gère tout l'affichage textuel (narrations, état du joueur, menus).
- Cette séparation permet au reste du code de demander "Affiche un message" sans se soucier de la manière dont il est affiché.

3. `Salle`

Le monde est constitué de différents lieux, tous dérivés de la classe abstraite **`Salle`**.

- **Concept de base :** Chaque salle possède une description et un état (visitée ou non).
- **Spécialisation :** Grâce au mécanisme d'héritage, nous avons trois types de salles aux comportements distincts :
    - **`SalleTresor`** : Permet au joueur d'obtenir un objet (la prophétie).
    - **`SalleEnigme`** : Impose une épreuve intellectuelle au joueur.
    - **`SalleCombat`** : Déclenche une confrontation avec un ennemi.

- **Interaction :** La méthode clé est `explorer()`. Le moteur de jeu appelle cette méthode sans savoir dans quel type de salle il se trouve, et c'est la salle elle-même qui exécute sa logique spécifique (lancer un combat, poser une question, etc.).

4. `Entite`

Les protagonistes du jeu sont regroupés sous le concept abstrait d'**`Entite`**, qui définit les caractéristiques communes à tout être vivant (Nom, Points de Vie, méthode pour recevoir des dégâts).

- **`Joueur`** : Ajoute des spécificités liées au héros, comme la gestion de l'inventaire (potions) et la capacité de se soigner.
- **`Ennemi`** : Représente les adversaires (comme le Gobelin), caractérisés par une puissance d'attaque spécifique.
- Lors d'un combat, ces deux classes interagissent directement : le joueur attaque l'ennemi et vice-versa, en modifiant mutuellement leurs points de vie.

5. Ordre d'exécution

L'interaction générale suit le cycle suivant (visible sur le diagramme d'activité) :

1. **Choix :** Le `Jeu` demande une `Direction` via l'`InterfaceUtilisateur`.
2. **Déplacement :** Le joueur est déplacé vers la salle correspondante.
3. **Exploration :** La méthode `explorer()` de la salle courante est activée.
    - Si c'est un combat, la boucle de combat s'enclenche entre `Joueur` et `Ennemi`.
    - Si c'est une énigme, la boucle de questions/réponses s'active.
4. **Vérification :** Le jeu vérifie si le joueur est toujours vivant ou s'il a gagné, puis recommence le cycle.