TD 11 : Patterns Strategie et Decorateur

Objectifs pédagogiques : patterns *Strategie*, *Decorateur*

11.1 Soccer Game

Employé dans une entreprise de jeux vidéos, vous avez la charge de concevoir et déployer l'implémentation des actions que peuvent réaliser des personnages évoluant dans un jeu de football. Le cahier des charges est le suivant :

- Un match fait se rencontrer deux équipes (l'une à domicile, l'autre à l'extérieur), chacune comportant 11 joueurs.
- Un joueur possède un nom et une distance au but (entre 0 et 100m).
- Un joueur peut être un attaquant, un centre, un ailier gauche, un ailier droit, un défenseur ou un gardien de but.
- Tous les joueurs sauf les attaquants peuvent faire une passe à un autre joueur de leur équipe à condition qu'ils aient le ballon. La passe est réussie avec une probabilité de 80%. Si la passe est ratée, le ballon est récupéré par un joueur de l'équipe adverse (tiré au hasard).
- Les attaquants, les centres et les ailiers peuvent tirer au but à condition qu'ils aient le ballon. Le but est marqué selon une probabilité de 50%. Après un tir, le ballon est récupéré par le gardien de but de l'équipe adverse.
- Les défenseurs, les centres et les ailiers peuvent réaliser des tacles sur des joueurs de l'équipe adverse. Le tacle est réussi selon une probabilité de 50%. Si le joueur taclé possédait le ballon, le ballon est alors récupéré par le joueur à l'origine du tacle.

⇒ **Proposez un diagramme UML** pour cette application, **puis donnez**, de manière informelle, **la manière dont les actions des joueurs seront implémentées**.

Après réflexion, vous souhaitez modifier votre programme pour que les probabilités de réussite des tirs des ailiers ne soient plus fixée à 50% mais dépendent également de leur distance au but (par exemple probabilité de réussite = (100 - distance au but) / 2). On vous demande également de modifier la probabilité de réussite des passes pour qu'elle ne soit plus que de 70%.

⇒ Énumérez les zones de code qui doivent être modifiées puis, en pensant aux éventuelles autres évolutions que l'on pourra vous demander, déduisez-en les limites de l'organisation utilisée à la question précédente.

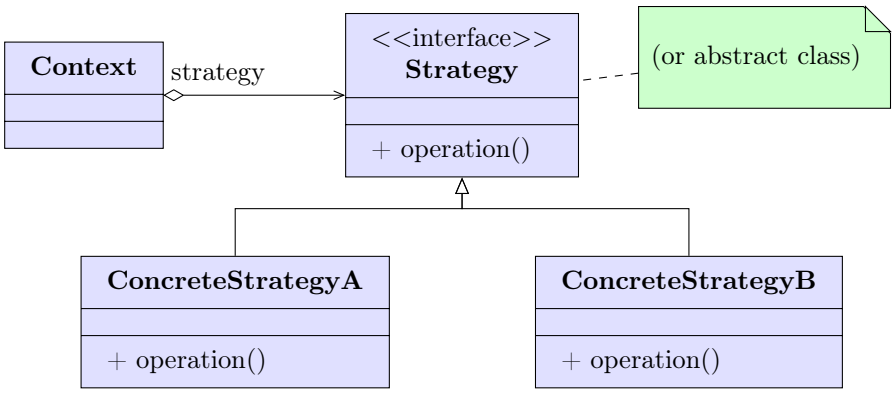

Figure 2: Le diagramme UML du DP *Strategy*.

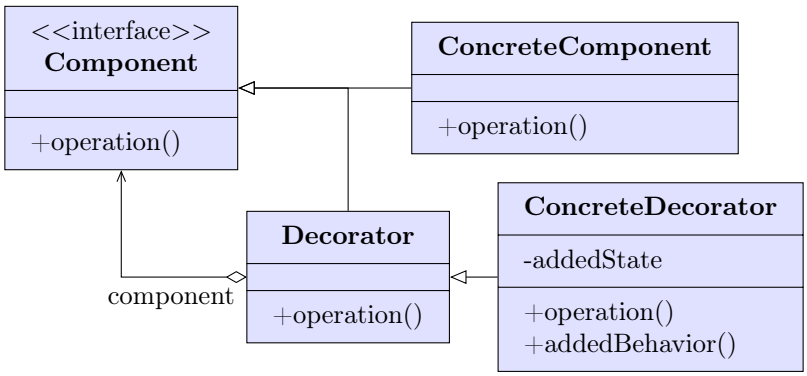

Figure 3: Le diagramme UML du *Decorator*.

11.2 Stratégies

Les limitations identifiées précédemment peuvent être levées en considérant le design pattern *Strategy* (Figure 2) pour les actions de tir, de passe et de défense.

⇒ Décrivez de façon diagrammatique la mise en œuvre du pattern *Strategy*, en expliquant les classes qui jouent le rôle de :

- Contexte abstrait et contextes concrets.
- Stratégie (ou comportement) abstraite et stratégies concrètes.

⇒ Détaillez le code des différentes classes Stratégie et expliquer les modifications nécessaires sur les classes de joueurs.

11.3 Décorateurs

Finalement, on souhaiterait que la probabilité de réussite d'une action soit majorée de 10% si le joueur qui l'effectue joue à domicile. De plus, on voudrait que la capacité de réussite d'un joueur ne soit pas figée mais évolue au cours du match selon sa proportion de coups réussis (confiance du joueur).

Pour cela, nous proposons de mettre en œuvre le design pattern *Decorator* (Figure 3) afin de décorer les stratégies de tir par deux décorations (on pourrait imaginer des décorateurs similaires pour les actions de passe et de défense).

Il s'agit d'ajouter les trois classes suivantes:

- la classe `StrategieTirDecorateur` représentant le décorateur abstrait de stratégie de tir. Elle définit un attribut avec accesseur protégé `strategieADecorer` de type `StrategieTir` et une méthode abstraite protégée `modifieTir(Joueur j, double tir)` retournant la probabilité de réussite du tir. La perturbation de la probabilité de réussite est effectuée par la redéfinition, dans la classe `StrategieTirDecorateur`, de la méthode de tir `tirImpl()` retournant la probabilité de réussite du tir :

```
public int tirImpl(Joueur j){
  return (modifieTir(j, strategieADecorer.tirImpl(j)));
}
```

- la classe `StrategieTirDomicile` représentant un décorateur concret permettant de prendre en compte le fait qu'un joueur joue à domicile ou non. Ce décorateur donne 10% de chance supplémentaire de réussir le coup si le joueur appartient à l'équipe évoluant à domicile.
- la classe `StrategieTirConfiance` représentant un décorateur concret permettant de prendre en compte l'état de confiance du joueur. Ce décorateur ajoute à la probabilité de réussir le coup

un nombre n = 20 *(nombre de tirs réussis par le joueur / nombre de tirs du joueur) - 10.

⇒ Proposez les modifications nécessaires sur les diagrammes UML pour introduire les trois nouvelles classes.

⇒ Définissez les codes des trois nouvelles classes.

⇒ Écrivez un code client créant les deux équipes, les faisant jouer aléatoirement pendant un nombre d'itérations donné (à chaque itération, une action est choisie de manière équiprobable : passe du joueur possédant le ballon, tir du joueur possédant le ballon ou tacle d'un joueur tiré au hasard sur le joueur possédant le ballon) et affichant finalement le gagnant de la rencontre.