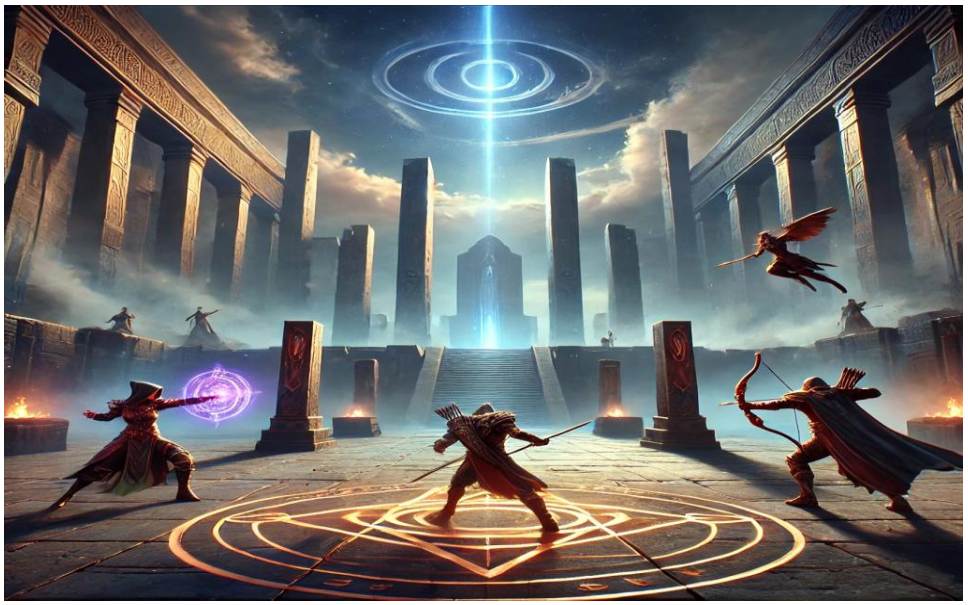
RAPPORT SAE PROGRAMMATION OBJET : création d'une arène de jeu



Réalisé par :

Adiaratou N'DIAYE

Encadré par :

Morgane VIDAL

Table des matières

1.Introduction

Ce projet a pour objectif de développer une application en Java intégrant une interface graphique, permettant de simuler des combats entre différents types de personnages que j'ai créés. Ce rapport détaille les étapes clés de la réalisation, depuis la conception des classes représentant les personnages jusqu'à la mise en place de l'interface graphique.
Le document est structuré en deux parties principales : la première aborde la création des classes, leurs constructeurs et méthodes, ainsi que les choix de conception effectués. La seconde partie se concentre sur la conception et le développement de l'interface graphique, expliquant comment elle permet d’interagir avec le système et de visualiser les combats.

2. Conception et Développement des Classes

2.1 Architecture globale

L'application est structurée selon une architecture modulaire basée sur les principes de la programmation orientée objet (POO). Chaque élément fonctionnel du projet est encapsulé dans une classe spécifique, garantissant ainsi une meilleure organisation du code.

2.2 Description des classes

- Classe personnage (classe parent)

Rôle : sert de bases pour les entités participant aux combats. Elle regroupe les comportements commun aux différents types de personnages.

Attributs principaux :

- PV(pour les points de vie),
- attaque(pour la force d'attaque),
- nom_perso(pour le nom du personnage)

Constructeur : Le constructeur initialise le nom et la force d'attaque du personnage, tandis que les points de vie sont fixés à 100 par défaut.

Méthodes principales :

attaquer(personnage cible) : Simule une attaque en infligeant des dégâts équivalents à la force d'attaque du personnage attaquant.

subirDegats(int degats) : Réduit les points de vie en fonction des dégâts subis, tout en s'assurant qu'ils restent dans l'intervalle [0, 100].

afficherEtat() : Affiche les points de vie, le nom et la force d'attaque du personnage.

Cette classe constitue la base toutes les classes de nos personnages .

Les méthodes comme **attaquer** et **subirDegats** sont redéfinies dans les sous-classes pour refléter des comportements spécifiques.

- Classe Mage(sous-classe)

Rôle : Le mage utilise sa réserve de mana pour lancer des sorts. Il peut infliger des dégâts en utilisant ce mana, et la puissance des sorts est modulée par la quantité de mana disponible.

Attributs supplémentaires :

- **mana : Le nombre de points de mana dont dispose le mage pour lancer ses sorts.**
- **puissanceSort :** La puissance des sorts que le mage peut lancer.

Méthodes :

- **attaquer(personnage cible)** : Le mage lance un sort qui inflige des dégâts à un personnage cible, en fonction de la quantité de mana disponible. Chaque attaque consomme 5 points de mana**.**
- **subirDegats(int degats)** : Lorsque le mage subit des dégâts, la quantité de mana est déduite des dégâts infligés. Si la mana est suffisante, les dégâts peuvent être réduits voire annulés.

- Classe Archer(sous-classe)

Rôle : Représente un personnage spécialisé dans l'usage d'arcs et de flèches. Son agilité lui permet de réduire les dégâts qu'il subit, et il doit gérer son stock de flèches pour attaquer.Attributs supplémentaires :

- **fleches** : le nombre de flèches dont dispose l'archer pour attaquer.
- **agilite :** un score qui permet de réduire les dégâts reçus en fonction de l'agilité du personnage.

Méthodes :

- attaquer(personnage cible) : Si l'archer a encore des flèches, il peut attaquer un personnage cible en infligeant des dégâts égaux à sa force d'attaque. À chaque attaque, le nombre de flèches est réduit de 1.
- subirDegats(int degats) : Réduit les dégâts subis en fonction de l'agilité de l'archer. Si l'agilité est suffisante, les dégâts peuvent être complètement évités.
- peutCombattre() : Cette méthode détermine si l'archer peut encore participer à un combat, en fonction de son nombre de points de vie (PV) et de flèches disponibles.

- Classe Guerrier(sous-classe)

Rôle : Le guerrier est un combattant robuste avec une défense élevée grâce à son armure. Il utilise principalement des attaques physiques (par exemple avec une épée).

Attributs supplémentaires :

- **armure :** La résistance physique qui permet de réduire les dégâts subis pendant un combat.

Méthodes :

- **attaquer(personnage cible)** : Le guerrier inflige des dégâts égaux à sa force d'attaque à un autre personnage, tant qu'il est en état de combattre et que la cible est également en vie.
- **subirDegats(int degats)** : L'armure du guerrier permet de réduire les dégâts subis. Si les dégâts sont inférieurs à la valeur de l'armure, l'attaque est complètement bloquée.

- Classe Arène

La classe Arène est responsable de la gestion d'un combat entre deux personnages. Elle contient les méthodes permettant de simuler l'affrontement entre les deux combattants, en organisant les tours d'attaque jusqu'à ce qu'un personnage soit vaincu (c'est-à-dire lorsque ses points de vie (PV) tombent à zéro ou moins).

Attributs :

personnage1 : Un objet de type personnage représentant le premier combattant dans l'arène.

personnage2 : Un objet de type personnage représentant le deuxième combattant dans l'arène.

Constructeur :

arene(personnage personnage1, personnage personnage2) : Le constructeur prend deux objets de type personnage en argument. Ces deux objets représentent les combattants qui s'affronteront dans l'arène.

Méthodes :

combat() : Cette méthode organise le déroulement du combat entre les deux personnages. Elle fonctionne en alternant les attaques entre les deux combattants jusqu'à ce que l'un d'eux soit vaincu (c'est-à-dire, que ses points de vie atteignent zéro ou moins). Le combat s'arrête dès qu'un des personnages ne peut plus combattre.

Affichage des états initiaux : La méthode commence par afficher les états des deux personnages (points de vie, nom, force d'attaque, etc.).

Boucle de combat : Une boucle ***while*** est utilisée pour alterner les attaques entre les deux personnages, tant que les deux peuvent encore combattre.

Attaque de personnage1 sur personnage2 : Le premier personnage attaque le second.

Vérification de la capacité à combattre de personnage2 : Après chaque attaque, on vérifie si le deuxième personnage peut encore combattre. Si ses points de vie tombent à zéro ou moins, le combat se termine et la méthode retourne true pour signaler la fin du combat.

Attaque de personnage2 sur personnage1 : Si personnage2 est encore en vie, il riposte en attaquant personnage1.

Vérification de la capacité à combattre de personnage1 : Enfin, après l'attaque de personnage2, on vérifie si le premier personnage peut encore combattre. Si ses points de vie tombent à zéro ou moins, le combat prend fin.

Retour de la méthode : Si un des personnages ne peut plus combattre (ses points de vie sont à zéro ou moins), la méthode retourne true, signifiant la fin du combat. Si les deux personnages sont encore en vie après un tour d'attaque, la méthode retourne false, indiquant que le combat continue.

3. Interface graphique

Pour la création de notre interface il a fallu crée une classe adaptée que l'on a nommée fenetreprincipale et dans laquelle on a établit tout ce qu'il faut. Après maintes et maintes mises a jour l'interface graphique présentera :

1. Création de l'interface utilisateur :

Nous avons créer une fenêtre principale JFrame qui permettra de sélectionner les deux opposants du combat.

Par la suite on a créé 3 boutons permettant de choisir entre un mage un guerrier ou un archer.

Un label a été crée permettant de voir l'état actuel de la sélection des personnages.

Un bouton « Commencer le combat » est initialement inactif et ne le devient qu'au moment où l'on à choisit nos deux opposants.

2. Gestion de la sélection des personnages :

Une méthode « choisirPerso » permet d'enregistrer les choix des personnages(personnage1et personnage2) et ainsi mettre à jour l'affichage en conséquence.

3. Lancement du combat dans une nouvelle fenêtre :

Après sélection des opposants le combat se déroule dans une nouvelle fenêtre (combatFrame)

Cette fenêtre affiche les points de vie (PV) des combattants via des labels.

Le bouton « Lancer le tour » permet de lancer une attaque pour chaque personnage.

4. Logique de combat :

Une méthode (lancerTour) applique les attaques entre les deux personnages et met à jour les points de vie dans l'interface.

La méthode (verifierFinCombat) vérifier si l'un des personnages est vaincu(PV<=0) et affiche un message de fin de combat avec le gagnant et le perdant.

5. Retour à l'écran principal :

Un bouton « Retour à l'écran principal » ferme la fenêtre de combat et permet de revenir à n'importe quel moment à l'écran principal.

Le programme utilise des ActionListener afin de gérer les clics sur les boutons.

Conclusion

Ce projet de simulation de combats dans une arène a permis de mettre en œuvre les principes de la programmation orientée objet (POO) dans un contexte interactif. Les classes représentant différents types de personnages, tels que le mage, l'archer et le guerrier, ont été conçues avec des comportements distincts et ont été intégrées dans une interface graphique permettant à l'utilisateur de sélectionner et d'affronter ces personnages. Ce travail a aussi renforcé mes compétences en gestion d’interfaces graphiques, ainsi qu'en conception et modélisation de classes en Java