Page de garde

**Spécification fonctionnelle technique**

**Application JAVA  
Zoo féérique**

**Sommaire**

[I – Introduction 1](#_Toc151025547)

[A - Présentation du document 1](#_Toc151025548)

[B - Objectif de l'application 1](#_Toc151025549)

[II - Description générale 1](#_Toc151025550)

[**A - Vue d'ensemble du système** 1](#_Toc151025551)

[B - Architecture MVC (Modèle-Vue-Contrôleur) 1](#_Toc151025552)

[1 – Modèle 1](#_Toc151025553)

[2 – Vue 1](#_Toc151025554)

[3 – Contrôleur 1](#_Toc151025555)

[C - Les Design Pattern 1](#_Toc151025556)

[1 – Singleton 1](#_Toc151025557)

[2 – Factory 1](#_Toc151025558)

[III - Exigences fonctionnelles 2](#_Toc151025559)

[A – Les fonctionnalités de l’application 2](#_Toc151025560)

[1 – Les créatures 2](#_Toc151025561)

[2 – Les enclos 2](#_Toc151025562)

[3 – Le zoo fantastique 2](#_Toc151025563)

[4 – Le maitre zoo 2](#_Toc151025564)

[5 – Gestion automatique du zoo 2](#_Toc151025565)

[6 – Gestion manuelle du zoo 2](#_Toc151025566)

[B - Exigences relatives aux interactions entre les composants du MVC 2](#_Toc151025567)

[IV - Contraintes techniques 2](#_Toc151025568)

[A – Les structures de données 2](#_Toc151025569)

[1 – Les ensembles (Sets) 3](#_Toc151025570)

[2 – Les cartes (Map) 3](#_Toc151025571)

[B – Les algorithmes 3](#_Toc151025572)

[C – Les classes abstraites 3](#_Toc151025573)

[D – Les interfaces 3](#_Toc151025574)

[E – L’algorithme de tri 3](#_Toc151025575)

[D – Convention de nommage 3](#_Toc151025576)

[1 – Le nom des classes et des interfaces 3](#_Toc151025577)

[3 – Les constantes et membres statiques 3](#_Toc151025578)

[4 – Les packages 4](#_Toc151025579)

[V - Diagramme de classes 4](#_Toc151025580)

[VI – Tests 4](#_Toc151025581)

[VII - Gestion des erreurs 4](#_Toc151025582)

[VIII – Documentation 4](#_Toc151025583)

[A - Documentation du code 4](#_Toc151025584)

[B - Guide de l'utilisateur 4](#_Toc151025585)

# I – Introduction

## A - Présentation du document

Cette spécification technique décrit les exigences pour le développement de l'application du Zoo Fantastique. Il fournit des indications claires sur les objectifs et servira de référence sur la portée et la structure du projet. Il constitue une ressource complète pour l'équipe en détaillant les fonctionnalités attendues, les contraintes techniques, ainsi que les lignes directrices en matière de tests et de documentation. Il permet de bien comprendre la nature, la portée et les objectifs du projet, ce qui facilitera une collaboration efficace tout au long du cycle de développement.

## B - Objectif de l'application

L'objectif principal de l'application est d'offrir une expérience dans un monde magique rempli de créatures mythiques (telles que les licornes, les mégalodons, les sirènes…). Les utilisateurs auront la possibilité d'observer, de gérer et d'interagir avec ces créatures dans des enclos spécialement conçus pour répondre à leurs caractéristiques uniques.

Au cœur de l’application se trouve le gardien de zoo, qui joue un rôle central, en supervisant à la fois les enclos et les créatures individuelles. L'application vise à donner au gardien de zoo la possibilité d'effectuer diverses tâches, notamment l'alimentation, le nettoyage et la satisfaction des besoins spécifiques de chaque créature. La gestion pourra se faire de manière manuel ou automatique.

# II - Description générale

## A - Vue d'ensemble du système

L'application fonctionne selon une architecture Modèle-Vue-Contrôleur (MVC), offrant un système complet de gestion et de présentation du monde des créatures mythiques. Au cœur du système se trouvent les créatures, les enclos et le zoo lui-même. Chaque action sera réalisée par le contrôleur, et enverra à la vue une information afin de prévenir l’utilisateur de tout ce que l’application est en train de faire. Il aura la possibilité de choisir lui-même les actions qu’il veut réaliser.

## B - Architecture MVC (Modèle-Vue-Contrôleur)

### 1 – Modèle

Dans le modèle, l'application encapsule toutes les données et la logique liées aux créatures, aux enclos et au zoo. Le modèle sert de colonne vertébrale à l'application, en garantissant l'intégrité des données et en gérant les fonctionnalités de base qui dictent le comportement des créatures et de leurs habitats.

### 2 – Vue

Les composantes de vue se concentrent sur l'interface et l'expérience utilisateur, responsable de la présentation du monde du Zoo Fantastique aux utilisateurs. En communication constante avec le contrôleur, la vue va permettre de prévenir l’utilisateur sur l’ensemble des actions réalisés par l’application, mais aussi de récupérer directement les entrées qu’il pourra faire.

### 3 – Contrôleur

Le contrôleur sert de médiateur entre le modèle et la vue. Il orchestre les interactions des utilisateurs et les réponses du système. Il interprète les entrées des utilisateurs, met à jour le modèle en conséquence et s'assure que la vue reflète les changements les plus récents. Le contrôleur joue un rôle essentiel dans le maintien de la séparation des préoccupations au sein de l'application, ce qui facilite une conception flexible et modulaire qui améliore la réactivité et la fonctionnalité globales de l’application.

## C - Les Design Pattern

### 1 – Singleton

Le modèle de conception Singleton est utilisé pour les instances du zoo et du maitre du zoo. En utilisant Singleton, l'application s'assure qu'il n'y a qu'une seule instance du zoo et une seule instance du maitre zoo tout au long de son cycle de vie. Ce choix de conception maintient un point de contrôle et de coordination unique, évitant ainsi de multiples instances conflictuelles et améliorant la cohérence et l'intégrité du zoo fantastique.

### 2 – Factory

Le design pattern Factory est implémenté pour la création de créatures au sein de l'application. Il impose de créer les créatures par l'intermédiaire de la fabrique désignée, fournissant ainsi une approche structurée et contrôlée de l'instanciation des créatures. Ce modèle permet de passer des types de créatures en tant que paramètres, ce qui assure une souplesse et une extensibilité au processus de création. En centralisant la création de créatures par le biais du modèle Factory, le zoo fantastique garantit une approche standardisée et personnalisable de l'introduction de nouvelles créatures.

# III - Exigences fonctionnelles

## A – Les fonctionnalités de l’application

### 1 – Les créatures

Chaque créature (dragon, lycanthrope, licorne, nymphe…) doit être dotée de caractéristiques telles que le nom, le sexe, le poids, la taille, l'âge, un indicateur de faim, un indicateur de sommeil, et un indicateur de santé. Les créatures peuvent accomplir des actions basiques telles que manger, émettre un son, être soignées, s'endormir, se réveiller, et vieillir.

De plus, chaque créature aura une famille (vivipare ou ovipare) avec des actions spécifiques à leur genre comme pondre un œuf ou mettre bas. Les créatures pourront aussi hériter de méthodes spécifiques à leur genre (aquatique, terrestre, immortel…) comme courir, nager, voler ou renaitre.

### 2 – Les enclos

Le Zoo fantastique comprend trois types d'enclos distincts : Enclos classique, Volière (avec un toit), et Aquarium (avec des considérations de profondeur et de salinité de l'eau). Il faut assurer la compatibilité des types de créatures avec leurs enclos respectifs afin de permettre une cohabitation harmonieuse. Les enclos contiennent des fonctions de gestion essentielles telles que le nettoyage de l’enclos et l'alimentation des créatures.

### 3 – Le zoo fantastique

L'entité globale, le Fantastic Zoo, agit comme un Singleton, encapsulant l'ensemble de l'écosystème du zoo. Ce composant de gestion central facilite une navigation efficace, permettant de retrouver les enclos par leur nom et fournissant une vue d'ensemble du nombre total de créatures dans le zoo. Le Zoo Fantastique sert de point central pour la gestion globale du zoo.

### 4 – Le maitre zoo

Le maître zoo est lui aussi un Singleton qui représentera l’utilisateur. C’est le gestionnaire qui aura la responsabilité d’examiner des enclos, de les nettoyer, de nourrir les créatures, et de transférer de créatures entre les enclos. L'aspect temporel de la simulation est pris en compte, avec un compteur décrémenté selon le nombre d'actions réalisées.

L'aspect temporel de la simulation doit être pris en compte, introduisant des éléments aléatoires tels que des changements d'état pour certaines créatures (maladie, sommeil, etc.) et des altérations dans les enclos (propreté, salinité, etc.). Nous avons choisi pour cela d’implémenter dans un premier temps le temps sous forme de compteur qui sera décrémenté selon le nombre d’action réalisé. Une fois le compteur à 0, un changement d’année sera effectué avec toutes les actions que cela engendre (les créatures vieillissent, les enclos se dégradent, des enfants naissent…).

### 5 – Gestion automatique du zoo

L'application intègre des méthodes de contrôle automatisées qui gèrent dynamiquement le zoo, assurant le bien-être des créatures et la cohérence globale de l'écosystème. Cette fonctionnalité permet au zoo d’être géré automatiquement grâce à des méthodes de contrôle des différentes caractéristiques.

### 6 – Gestion manuelle du zoo

La gestion manuelle du zoo par l’utilisateur permet une prise de décision directe. Les utilisateurs peuvent choisir d'intervenir dans l'entretien et l'organisation du zoo, ce qui favorise une interaction personnalisée et engageante avec les créatures mythiques et leurs habitats. Leur but est d’avancer dans leur aventure et de créer un zoo de plus en plus grand et sain.

## B - Exigences relatives aux interactions entre les composants du MVC

Des interactions efficaces entre les composants Modèle, Vue et Contrôleur au sein de l'architecture MVC sont essentielles au bon fonctionnement de l'application. Le modèle, responsable de l'encapsulation des données et de la logique liées aux créatures, aux enclos et au zoo, garantit que toutes les créatures et tous les enclos sont représentés avec précision. La vue, qui se concentre sur l'interface et l'expérience utilisateur, communique avec le modèle pour récupérer les données à présenter. Elle met à jour dynamiquement l'interface utilisateur en fonction des changements apportés au modèle. Le contrôleur, qui joue le rôle de médiateur entre le modèle et la vue, interprète les données de l'utilisateur et les traduit en actions sur le modèle. Le modèle, la vue et le contrôleur doivent collaborer de manière harmonieusement, afin de créer un cadre cohérent et efficace pour la gestion des créatures mythiques et des enclos de l’application.

# IV - Contraintes techniques

## A – Les structures de données

### 1 – Les ensembles (Sets)

L'application utilise les ensembles comme structure de données principale (en particulier pour la gestion des enclos dans le zoo). Les ensembles constituent un moyen efficace de gérer des collections uniques d'enclos, en veillant à ce que chaque enclos soit distinct et en évitant les doublons. Ce choix favorise l'organisation et la catégorisation des enclos, ce qui permet de rationaliser les opérations dans l'environnement du zoo. L’ajout et la suppression d’éléments est rapide et simplifié par rapport à un tableau.

### 2 – Les cartes (Map)

Les cartes sont utilisées pour organiser les créatures dans les enclos, chaque créature se voyant attribuer un identifiant unique (ID) comme clé. L'utilisation de cartes facilite l'accès rapide et direct à des créatures spécifiques sur la base de leur identifiant, optimisant ainsi la récupération et la manipulation d'êtres mythiques individuels. L’association d’une clé à chaque créature (qui est mise à jour à chaque ajout et suppression) permet aussi à l’utilisateur d’effectuer une sélection de créature de manière simplifié et rapide.

## B – Les algorithmes

**1 – Recherche linéaire**

L'application utilise l'algorithme de recherche linéaire pour des tâches telles que la recherche d'un enclos sur la base de son nom. Le principe de la recherche linéaire consiste à parcourir une liste de manière séquentielle jusqu'à ce que l'élément recherché soit trouvé. Dans le contexte de la localisation des enclos, cet algorithme parcourt la liste des enclos en comparant les noms jusqu'à ce qu'une correspondance soit identifiée. La mise en œuvre implique une itération directe à travers la liste, ce qui la rend adaptée aux scénarios dans lesquels la taille de la liste est relativement petite ou n'est pas triée.

## C – Les classes abstraites

Les classes abstraites jouent un rôle crucial dans la conception de l'application, notamment en définissant des caractéristiques communes pour les créatures. Les classes abstraites, telles que celles des créatures, des ovipares et des vivipares, encapsulent des attributs et des comportements partagés tout en permettant des implémentations spécifiques dans leurs sous-classes concrètes. Cette abstraction fournit une hiérarchie structurée, favorisant la réutilisation du code et assurant la cohérence dans l'implémentation des différents types de créatures.

## D – Les interfaces

Les interfaces sont utilisées pour classer les créatures en fonction de certains traits, comme le fait d'être aquatique, aérien ou immortel. Ce choix de conception facilite à la fois la vérification des caractéristiques d'une créature et la standardisation des méthodes propres à chaque type de créature. En mettant en œuvre des interfaces, le zoo fantastique globalise les méthodes associées à des types de créatures spécifiques, ce qui favorise une approche modulaire et extensible.

## E – L’algorithme de tri

Afin de trier les créatures dans un enclos, nous avons choisi d’utiliser le tri par arbre binaire de recherche (ABR) afin que les créatures soient triées selon leur âge dans leur enclos. Chaque créature est représentée par un nœud de l’arbre. Les créatures plus jeunes sont situées du côté gauche, et les plus âgées sont placées du côté droit. Pour récupérer la liste des créatures de l’arbre, un parcours préfixe est effectué.

Le principal avantage de cette méthode est son efficacité, mais aussi que c’est une structure auto-équilibrée. C’est une méthode similaire aux arbres AVL. Cet algorithme de tri va permettre un maintien automatique de l’équilibre lors de l’insertion et de la suppression. Nous avons donc choisi cette méthode qui est déjà implémenté au sein de la structure TreeMap car elle correspond à nos besoins.

## D – Convention de nommage

### 1 – Le nom des classes et des interfaces

Les noms de classes et d'interfaces respectent la convention PascalCase, commençant par une majuscule, et sont choisis pour être significatifs et indiquer leur but dans la base de code.

### 2 – Le nom des méthodes et des variables

Les noms de méthodes et de variables suivent la convention camelCase, commençant par une lettre minuscule. Cette convention garantit une approche cohérente et claire de l'attribution des noms, contribuant ainsi à la lisibilité et à la maintenabilité du code.

### 3 – Les constantes et membres statiques

Les constantes et les membres statiques sont nommés en majuscules, les mots étant séparés par des traits de soulignement. Cette convention les distingue des variables ordinaires et met l'accent sur leur immuabilité ou leur nature statique dans la base de code.

### 4 – Les packages

Les noms de packages sont écrits en minuscules. Cette convention s'aligne sur les pratiques standard de Java, contribuant à une structure unifiée et organisée au sein de la base de code.

# V - Diagramme de classes

UML

# VI – Tests

Plan de tests pour chaque fonctionnalité

# VII - Gestion des erreurs

Comment les erreurs seront gérées et signalées

# VIII – Documentation

## A - Documentation du code

E

## B - Guide de l'utilisateur

E