Travaux pratique du chapitre 2 : Patterns de construction d'objets (créateur)

Singleton

Contexte

Vous êtes engagé(e) comme développeur(euse) pour construire un système de gestion pour un zoo virtuel. La direction de ce zoo souhaite avoir une application qui suit l'état général du zoo et qui permet de prendre des décisions en temps réel. Pour cela, le système devra avoir un point d'entrée unique qui centralise toutes les informations globales à propos du zoo, tel qu'un contrôleur principal. Ce contrôleur permettra de garantir que les informations sont cohérentes et à jour.

Mission

Votre mission est de concevoir et d'implémenter un pattern Singleton pour créer un contrôleur principal qui gère l'accès aux informations centralisées du zoo. Vous devez garantir qu'il n'existe qu'une seule instance de ce contrôleur à tout moment pour éviter des conflits de données et assurer la bonne gestion des ressources du zoo.

Exigence

1. Création du Contrôleur du Zoo :

- Implémenter une classe `ZooController` en utilisant le pattern Singleton.

- Assurez-vous que cette classe ne peut être instanciée qu'une seule fois.

2. Fonctionnalités à Implémenter :

- Affichage de l'État du Zoo : Le contrôleur doit pouvoir afficher les statistiques globales du zoo, telles que le nombre total d'animaux, les visiteurs actuels, etc.

- Mise à Jour des Informations : Ajouter des fonctions pour mettre à jour ces informations de manière centralisée.

3. Démonstration d'Utilisation :

- Implanter un petit script qui démontre l'utilisation de l'instance unique du `ZooController`.

- Montrez qu'il est impossible de créer plusieurs instances de `ZooController`.

4. Langage :

- Le projet doit être codé en Java. Vous êtes encouragé(e) à ajouter des commentaires expliquant chaque étape de votre implémentation.

Factory

Contexte

Vous avez été embauché par une organisation secrète spécialisée dans la gestion et l'étude des créatures mythiques. Vos nouvelles responsabilités incluent la conception et le développement d'une application qui gère différentes créatures mythiques en utilisant le pattern Factory.

Pour cette tâche, vous devez créer un système qui peut générer différentes créatures mythiques sur la base de demandes spécifiques. Chaque créature a des caractéristiques propres, et de nouvelles créatures peuvent être ajoutées au système sans modification majeure du code existant.

Objectif

Implémenter une application en utilisant le pattern Factory pour créer et gérer des créatures mythiques.

Exigence

1. Classe de Base

- Créez une classe `Creature` qui servira de classe de base pour toutes les créatures mythiques. Cette classe doit avoir une méthode abstraite ou virtuelle `get_description()` qui sera implémentée par les classes dérivées.

2. Classes de Créatures

- Implémentez au moins trois classes de créatures différentes : `Dragon`, `Phoenix`, `Griffon`.

- Chaque classe doit hériter de la classe `Creature` et implémenter la méthode `get_description()`, qui retourne une description unique de la créature.

3. Classe Factory

- Créez une classe `CreatureFactory` qui a une méthode `create_creature(type_creature : str)` pour générer une instance d'une créature en fonction du type spécifié.

- La méthode `create_creature` doit retourner un objet de la classe appropriée (`Dragon`, `Phoenix`, `Griffon`) en fonction du type demandé.

4. Interface Utilisateur

- Créez une simple interface utilisateur (console ou graphique) où l'utilisateur peut entrer le type de créature qu'il souhaite créer.

- Affichez la description de la créature générée.

5. Extensions

- Ajoutez la capacité d'intégrer de nouvelles créatures en tant que nouvelles classes sans modifier la structure de votre `CreatureFactory`.

- Documentez comment vous ajouteriez une nouvelle créature, par exemple un `Cerberus`.

6. Langage

- Le projet doit être codé en Python. Vous êtes encouragé(e) à ajouter des commentaires expliquant chaque étape de votre implémentation.

Prototype

Contexte

Dans cet exercice, vous allez concevoir une application JavaScript qui utilise le pattern Prototype pour cloner des super-héros. Vous faites partie d'une équipe de développeurs travaillant pour une agence de super-héros fictive. Votre mission est de créer un système qui permet de cloner rapidement des super-héros déjà existants afin de réagir efficacement à des menaces urgentes.

Objectif

L'agence de super-héros dispose d'un ensemble de super-héros dotés de pouvoirs variés. Chaque super-héros a les caractéristiques suivantes :

1. **Nom** : Le nom du super-héros.
2. **Pouvoirs** : Une liste de pouvoirs (par exemple, vol, super-force, invisibilité).
3. **Expérience** : Années d'expérience dans le combat contre le crime.
4. **Équipe** : Nom de l'équipe à laquelle le super-héros appartient.

L'objectif est de créer une application qui permet de :

1. Créer des instances de super-héros.
2. Cloner ces super-héros pour créer de nouveaux super-héros disposant des mêmes caractéristiques (nom, pouvoirs, expérience, équipe).
3. Modifier certaines propriétés si nécessaire après le clonage (par exemple, le nom ou l'équipe du super-héros).

Exigence

1. Créer une classe `SuperHero`

- Définit les propriétés `nom`, `pouvoirs`, `experience`, et `equipe`.

2. Ajouter une méthode `clone()` à la classe `SuperHero`

- Cette méthode doit retourner une nouvelle instance de `SuperHero` avec des propriétés identiques à l'objet original.

3. Démonstration de clonage

- Créez au moins deux super-héros originaux.

- Clonez chacun d'eux et changez leur nom pour s'assurer qu'ils sont des individus uniques malgré le clonage.

- Affichez les caractéristiques des super-héros clonés pour valider le processus.

4. Langage

- Le projet doit être codé en Javascript. Vous êtes encouragé(e) à ajouter des commentaires expliquant chaque étape de votre implémentation.