EXAMEN SE 350

Design Pattern 2023-2024

Cet examen comporte 3 parties :

1. Quelques questions de réflexion (5 pts)

2. Exemples à décrypter (2,5 pts)

3. Pratique (12,5 pts)

Partie 1

Question 1)

Qu'est ce que l'héritage et la composition ? Qu'est ce qui distingue ces deux approches ?

Question 2)

Indiquez des différences entre les designs patterns Observer et Strategy.

Question 3)

Dans un simulateur de jeu de course, des obstacles apparaissent aléatoirement. De même, le véhicule peut être endommagé où tomber en panne. Selon la situation, la performance du pilote est affectée et les options du jeu évoluent. Justifiez votre choix de design pattern.

Question 4)

Décrivez un cas d'application pour lequel le design pattern decorator serait le plus approprié.

Question 5)

Dans un développement logiciel, on essaie le plus possible de séparer la création des objets et l'utilisation de ces objets.

Quel est l'intérêt de cette séparation ?

Partie 2

Dans un programme qui permet de faire des plans d'une pièce (cuisine, salon, ...) avec ses meubles, nous avons, suivant les cas, plusieurs modes d'affichage d'un même plan :

- Le plan lui-même avec l'emplacement des différents meubles : le plan de base ;
- Le plan de base avec les mesures des meubles et de la pièce ;
- Le plan de base avec un devis ;
- Le plan de base avec un devis et les mesures.

Quel Design Pattern identifiez-vous ?

Partie 3

Exercice sur le Design Pattern Inversion of Control (IoC) avec Spring Framework :

Objectif :

Implémenter le Design Pattern IoC en utilisant Spring Framework pour gérer les dépendances dans une application de gestion de commandes.

Scénario :

Supposons que vous développiez une application de gestion de commandes. Vous avez une classe OrderService qui gère les opérations liées aux commandes. La classe OrderService dépend d'une interface OrderRepository pour accéder aux données des commandes.

Instructions :

Définir l'interface OrderRepository :

Créez une interface OrderRepository avec des méthodes pour accéder aux données des commandes, telles que getOrderById, getAllOrders, etc.

Implémenter une classe OrderRepository :

Implémentez une classe JdbcOrderRepository qui implémente l'interface OrderRepository. Cette classe pourrait utiliser JDBC pour accéder à une base de données.

Configurer Spring :

Utilisez Spring pour configurer l'application. Utilisez l'inversion de contrôle pour déclarer les beans nécessaires, notamment une instance de JdbcOrderRepository et une instance de OrderService avec la dépendance injectée.

Utiliser l'IoC de Spring :

Utilisez l'inversion de contrôle de Spring pour injecter la dépendance OrderRepository dans la classe OrderService. Assurez-vous d'annoter correctement les classes et méthodes avec les annotations Spring nécessaires.

Ajouter des fonctionnalités à OrderService :

Dans la classe OrderService, ajoutez des méthodes pour effectuer des opérations sur les commandes, comme placeOrder, getOrderDetails, etc.

Testez :

Créez une classe principale avec une méthode main qui charge l'application Spring. Utilisez ApplicationContext pour récupérer une instance de OrderService et appelez ses méthodes pour effectuer des opérations sur les commandes.

Conseils :

Utilisez les annotations Spring telles que @Component, @Autowired, @Configuration, etc., pour configurer les dépendances et les beans.

Assurez-vous que votre application est correctement annotée pour permettre à Spring de scanner et de configurer les dépendances.

Utilisez une base de données simple (par exemple, H2) pour simplifier la configuration de la source de données.