Devoir Surveillé - Programmation Orienté Objet Avancé

16 Novembre 2011

1. Il y a 3 types d'attribut : public, private et protected. Quelles sont les différences entre les 3. Lequel est généralement déconseillé d'utiliser ? Pourquoi ?
2. Le premier principe des patrons de conception (design pattern) est de programmer via les interfaces. Pour quelle raison ?
3. Quelle est la différence entre classe abstraite et interface ?
4. Qu'est-ce qu'un patron de conception (design pattern) ?
5. Donnez les définitions pour les concepts de couplage faible et de cohésion forte. Dans vos définitions, préciser l'impact sur les attributs et les opérations d'une classe. Par exemple, une classe qui n'a aucune référence vers d'autre classe favorise-t-elle la cohésion ?
6. Qu'est-ce que la délégation ?
7. Quelle est la différence entre délégation et héritage ?
8. Par rapport à l'héritage, la délégation augmente-t-elle ou diminue-t-elle le couplage ? Pourquoi ?
9. Quel est le principe du pattern Observer ?
10. Qu'est-ce qu'une architecture 3 tiers ?

Correction :

1. Tout membre d'une classe déclarée `public` est visible par toutes les autres classes. Tout membre d'une classe A déclaré `protected` est visible par toutes les classes du même package et par toutes les classes qui héritent de A. Tout membre d'une classe A déclaré `private` est visible uniquement par la classe A. Il vaut mieux ne pas utiliser le mot clé `public`. Le caractère `public` d'un attribut casse le principe d'encapsulation des données. Il permet à n'importe quel objet de modifier les données internes d'un objet compromettant ainsi les données de l'objet et son fonctionnement.
2. L'objectif de l'interface est de séparer les implémentations des différentes classes qui constituent le design pattern. Une classe ne doit pas dépendre de l'implémentation d'une autre classe. Les dépendances doivent être vers des définitions de fonctionnalité et non vers du code d'implémentation.
3. Une interface est un cahier des charges. Elle ne propose aucune implémentation mais définie des fonctionnalités. Une classe abstraite est une classe qui propose des méthodes implémentées et des définitions de méthode. Les classes abstraites ne peuvent pas être instanciées. Une classe abstraite est une classe intermédiaire entre l'interface et la classe concrète. Elles sont utilisées pour factoriser du code entre les classes concrètes en utilisant le mécanisme d'héritage.
4. Un patron de conception est un arrangement caractéristique de modules, reconnu comme bonne pratique en réponse à un problème de conception d'un logiciel. Il décrit une solution standard, utilisable dans la conception de différents logiciels.
5. Cohésion forte = l'objet dispose des données nécessaires à l'exécution des opérations dont il est responsable. Ainsi, si les opérations d'une classes utilisent les attributs de la même classe alors on augmente la cohésion.

   Couplage faible = l'objet n'a pas besoin d'autre objet pour réaliser les opérations dont il est responsable. Ainsi, si les opérations d'une classe n'utilisent pas les opérations d'une autre classe alors on a un couplage faible.

   Une classe qui n'a aucune référence vers d'autre classe favorise la cohésion.
6. La délégation consiste à faire faire par un objet situé dans un attribut de la classe une opération que la classe devait réaliser.
7. Dans le mécanisme d'héritage, la classe fille hérite de toutes les fonctions des classes parentes. Ce n'est pas le cas du mécanisme de délégation. Une classe qui utilise le mécanisme de délégation n'est pas obligé de définir toutes les fonctions de la classe mère.
8. Par rapport à l'héritage, la délégation diminue le couplage. En effet, dans le cas de l'héritage, la classe dépend de l'implémentation de toutes ses classes parentes. Dans le cas de la délégation la classe ne dépend que de la classe à qui elle délègue les différentes tâches à réaliser.
9. Le patron de conception observateur/observable est utilisé en programmation pour envoyer un signal à des modules qui jouent le rôle d'observateur. Le principe consiste à enregistrer des observateurs auprès d'une classe observée. Lorsque la classe observée est modifiée, elle envoie un message aux observateurs qui se sont enregistrés. Les observateurs exécutent alors une action adéquate en fonction des informations qui ont été envoyées par la classe observée.

10. L'architecture 3 tiers, ou pattern MVC est une méthode de conception qui permet de séparer les différents aspects d'une application. Ce paradigme divise l'application en 3 parties : le modèle, la vue et le contrôleur. Le modèle se charge des données de l'application. La vue se charge de l'interaction entre l'homme et la machine. Le contrôleur se charge de synchroniser et de faire communiquer le modèle avec la vue.