VRAI OU FAUX ?
#1

M1 MIAGE IL-DP

2022-2023

1. **Réduire le couplage entre deux classes c'est limiter les dépendances entre ces deux classes et, par conséquent, augmenter la flexibilité du logiciel.**

2. **L'héritage permet de réduire les couplages entre classes.**

3. **L'utilisation de l'opérateur « new » est sans effet sur le couplage entre classes.**

4. **Un pattern, c'est l'association entre un problème récurrent et une solution réutilisable.**

5. **Les design patterns capitalisent l'expérience de conception et permettent de la réutiliser.**

6. **Une description d'un design pattern est composée d'un nom, de la description du problème et du contexte, de la solution et d'éléments complémentaires sur son utilisation et les conséquences.**

7. **Le polymorphisme est un design pattern du GoF.**

8. **Le design pattern « Méthode » est un design pattern du GoF.**

9. **Le pattern MVC est un design pattern du GoF.**

10. **Le design pattern « Patron de Méthode » vise l'organisation du code.**

11. **Quand on met en œuvre le pattern « Patron de méthode », la classe mère de la hiérarchie est toujours une classe abstraite.**

12. **La principale différence entre « Patron de Méthode » et « Stratégie » réside dans l'utilisation de l'héritage pour « Patron de Méthode » et de l'association pour « Stratégie ».**

13. **Le design pattern « Patron de Méthode » peut être combiné avec le design pattern « Stratégie » pour implanter les stratégies concrètes**

14. **Le pattern « Stratégie » sépare les responsabilités entre la classe utilisatrice et les classes qui implantent les stratégies.**

15. **Quand on met en œuvre le pattern « Stratégie », le super-type Stratégie (la classe mère de la hiérarchie des stratégies) est toujours une classe abstraite.**

16. **Quand on met en œuvre le pattern « Stratégie », la classe utilisatrice doit définir une méthode publique setStratégie() qui permet de donner une stratégie à l'objet utilisateur.**

17. **Avec le pattern « Stratégie », en séparant la classe utilisatrice et la stratégie (externalisation du code), on introduit un objet supplémentaire à l'exécution.**

18. **Pour mettre en œuvre le DP « Adaptateur », il faut légèrement modifier la classe Client ou la classe Adaptée**

19. **La solution du pattern « Adaptateur » repose sur l'utilisation du polymorphisme**

20. **Un (objet) adaptateur convertit un appel de méthode entrant en un appel d'une méthode qui est définie dans la classe Adaptée**

21. **Dans la solution du pattern « Adaptateur », le supertype Cible (qui définit le besoin du client) est obligatoirement une interface (au sens « Java »)**

22. **Dans la solution du pattern « Adaptateur », la classe Adaptateur doit hériter de la classe Adaptée**

23. **La solution du pattern « Adaptateur » dite de « niveau objet » permet d'adapter des objets instances de sous-classes de la classe Adaptée**