QCM-Java-Série-1

Q.C.M. Connaissances préalables

NB : Pour certaines questions, plusieurs propositions peuvent convenir. Cochez alors les réponses que vous jugerez pertinentes.

1. En **Java**, par quelle méthode démarre un programme ?

   ☐ par la méthode static int main (String arg[])
   ☐ par la méthode public static int main (String arg[])
   ☐ par la méthode static void main (String arg[])
   ☐ par la méthode public static int main (String arg[])
   ☐ par la méthode public static void main (String arg[])

2. En **Java**, un type **primitif** permet :

   ☐ de désigner des instances.
   ☐ de référencer des objets.
   ☐ de manipuler des variables scalaires.
   ☐ de pointer vers des entiers.

3. En **Java**, un type **objet** déclaré appartenir à une classe particulière permet :

   ☐ de désigner des instances.
   ☐ d'être de type **interface**.
   ☐ de désigner une instance issue d'une classe dérivée.
   ☐ de référencer des objets.
   ☐ de désigner une instance d'une classe de base.

4. En **P.O.O.,** un objet représente :

   ☐ un type de données qui permet de générer des instances.
   ☐ une entité créée à partir d'une classe.
   ☐ une variable particulière générée à partir d'un modèle .
   ☐ une instance.

5. En **Java**, une instance est localisée :

   ☐ dans la pile.
   ☐ dans le tas.
   ☐ dans la face.

6. En **Java**, une variable locale est localisée :

- ☐ dans la pile.
- ☐ dans le tas.
- ☐ dans la face.

7. En **Java**, quelle est la durée de vie d'une variable **locale** ?

- ☐ le temps d'exécution du bloc.
- ☐ cela dépend si c'est une variable de type **primitif** ou de type **objet.**
- ☐ le temps d'exécution de la méthode.
- ☐ le temps que dure l'application.

8. Une **classe** représente :

- ☐ l'abstraction d'un concept.
- ☐ un type de données évolué (abstrait).
- ☐ une instance particulière issue d'une classe.

9. **Java** permet l'héritage multiple ?

- ☐ oui.
- ☐ non.
- ☐ peut-être.

10. Une **interface** en **Java** représente :

- ☐ une classe ordinaire.
- ☐ une classe abstraite dotée de variables d'instances.
- ☐ un outil de spécification contractuel.
- ☐ une classe abstraite sans variables d'instances et dont les méthodes ne sont que déclarées.

11. Le **polymorphisme** permet :

- ☐ de contourner les contraintes des interfaces.
- ☐ de rendre non instanciable une classe concrète.
- ☐ de s'affranchir du type de l'instance référencée.
- ☐ d'unifier les appels de méthodes pour des objets issus de classes différentes .

12. L'opérateur **Java *new*** :

- ☐ ne s'utilise que pour les interfaces.
- ☐ ne s'utilise que pour les classes.
- ☐ permet la création d'un objet cohérent, grâce au concept de **constructeur**.
- ☐ s'utilise pour créer une classe à partir d'une autre.

13. Une **méthode de classe** :

☐ est atteignable partout où la classe est accessible.
☐ est atteignable sans qu'une instance ne soit créée .
☐ est un traitement spécifique à une instance.
☐ a une vocation d'utilitaires.
☐ est généralement ***public***.

14. Le modificateur d'accès ***private*** permet :
☐ de donner un accès aux méthodes de la classe.
☐ de donner un accès aux méthodes des classes dérivées.
☐ de donner un accès aux méthodes des classes du même package.

15. Le modificateur d'accès ***protected*** permet :
☐ de donner un accès sans restriction à la propriété.
☐ d'interdire l'accès à la propriété en dehors de la classe.
☐ de donner un accès dans les classes du même package.
☐ de rendre accessible la propriété concernée dans les classes filles.

16. Afin de vérifier votre compréhension, veuillez mener une réflexion et :

donnez (à vous-même ou à votre collègue) une définition la plus précise possible de la notion de **classe**.

17. Similairement :

donnez (à vous-même ou à votre collègue) une définition exacte de la notion **d'interface**.

18. Expliquez (à vous-même ou à votre collègue) :

ce que représente la notion **d'implémentation d'interface**.