QCM – Java

- API Collections -

Q.C.M. Connaissances préalables

Ce Q.C.M. doit être réalisé **avant** l'élaboration des exercices dédiés aux collections.

Il vous permet de :

- ✔ vous assurer que vous maîtrisez l'environnement des collections.
- ✔ revoir éventuellement certaines parties du support pour lever toute ambiguïté sur le sujet.

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

1. En **Java**, les structures de données séquentielles principales sont :

   ☐ la **pile**.
   ☐ la **file**.
   ☒ le **tableau**.
   ☒ la **liste**.
   ☒ gérées avec un **index** pour désigner leurs éléments.
   ☐ gérées avec un **objet** pour désigner leurs éléments.

2. Avec les structures de données séquentielles, un nouvel élément peut être inséré …

   ☐ seulement au **début**.
   ☐ seulement à la **fin**.
   ☒ entre le **début** et la **fin**.
   ☒ n'importe où.

3. En **Java**, un tableau de 10 entiers doit être déclaré par :

   ☒ **int** [] tableau = **new int** [10] ;
   ☒ **int** tableau[] = **new int** [10] ;.
   ☐ **int** [10] tableau = **new int** [] ;
   ☐ **int** tableau [10] = **new int** [10] ;
   ☐ **int** [] tableau = **int** [10] ;

4. En **Java**, un tableau peut contenir des éléments de type :

☒ **primitif**.
☒ **objet**.
☒ **String**.

5. En **Java**, un tableau :

☒ a une **taille invariable** valorisée à la déclaration.
☐ a une **taille variable** selon le nombre initial d'éléments qu'on y insère.
☐ possède une variable ***size*** qui représente le nombre d'éléments contenus.
☐ peut contenir des **éléments de type différent**.
☒ est un objet et peut lever des exceptions.
☐ instancie **5 String** dans l'extrait de code suivant : *String [] tableau = **new** String [5] ;*

6. Une liste chaînée :

☐ gère des **maillons** contigus en mémoire.
☐ impose un **nombre maximum** de maillons, déclaré lors de sa création.
☒ est très performante pour les insertions ou suppressions de maillons.
☐ est très performante pour les accès aux éléments.
☒ se manipule par l'utilisation de **références**.
☒ est composée de maillons possédant **au moins une référence**.
☒ se manipule en gérant **dynamiquement** ses maillons.

7. La structure de données de type « **pile** » :

☒ est une structure **LIFO**.
☐ est une structure **FIFO**.
☐ permet d'insérer un nouvel élément grâce à un **index**.
☐ possède une organisation interne dont l'utilisateur doit tenir compte.
☒ propose une interface publique **masquant son implémentation**.
☒ est disponible sous forme de classe dans **l'API *Collection***.

8. La structure de données de type « **file** » :

☐ est une structure **LIFO**.
☒ est une structure **FIFO**.
☐ permet d'insérer un nouvel élément grâce à un **index**.
☐ possède une organisation interne dont l'utilisateur doit tenir compte.
☒ propose une interface publique **masquant son implémentation**.
☒ est disponible sous forme de classe dans **l'API *Collection***.

9. En **Java,** toute collection :

☒ est **itérable**.
☒ est une **instance** destinée à contenir d'autres instances.
☐ est forcément issue de l'interface ***List***.
☒ est forcément issue d'une classe implémentant ***Collection*** ou ***Map***.
☒ propose un outil **d'itération typé**.
☒ propose un ensemble unifié d'outils pour gérer ses éléments.

10. En **Java**, une classe de collection implémentant ***List*** :

☐ propose la manipulation de ses éléments grâce à un **index**.
☒ peut contenir **plusieurs fois** le même objet.
☐ utilise la méthode ***equals()*** lors de l'ajout d'un nouvel élément.
☒ peut être la modélisation d'une **pile**.
☒ peut être la modélisation d'une **liste chaînée**.
☐ peut **trier** ses éléments.
☐ peut contenir des **variables primitives** (*int*, *char* , ….).

11. En **Java**, une classe de collection implémentant ***Set*** :

☐ propose la manipulation de ses éléments grâce à un **index**.
☐ peut contenir **plusieurs fois** le même objet.
☒ utilise la méthode ***equals()*** lors de l'ajout d'un nouvel élément.
☐ peut être la modélisation d'une **pile**.
☐ peut être la modélisation d'une **liste chaînée**.

12. En **Java**, une classe de collection implémentant ***SortedSet*** :

☐ autorise l'insertion de **plusieurs éléments identiques**.
☒ **trie** ses éléments selon une relation d'ordre.
☒ utilise une **table de hachage** pour ses éléments.
☒ peut s'appuyer sur l'ordre naturel de ses éléments pour son organisation.
☒ peut nécessiter une instance de ***Comparator***.
☒ peut solliciter implicitement l'interface ***Comparable***.

13. En **Java**, une collection implémentant ***Map*** :

☒ autorise l'insertion de **plusieurs valeurs identiques**.
☒ utilise une **clé** pour accéder à ses éléments.
☐ utilise une **table de hachage** pour ses éléments.
☐ peut contenir plusieurs associations avec la **même clé**.
☒ peut contenir plusieurs associations avec la **même valeur**.
☐ gère ses clés dans un ***List***.
☐ gère ses valeurs dans un ***Set.***

14. Dans l'**API Collection** de **Java**, ***SortedMap*** :

☐ est une **classe**.
☒ est une **interface**.
☒ hérite de ***Map***.
☒ établit une relation d'ordre sur les **clés**.
☐ établit une relation d'ordre sur les **valeurs**.
☒ gère ses clés dans un ***Set***.
☐ gère ses clés dans un ***List.***
☒ peut requérir une instance de ***Comparable*** lors de son instanciation
☒ peut requérir une instance de ***Comparator*** lors de son instanciation

15. Dans l'**API Collection** de **Java**, ***Map.Entry*** :

☐ gère les clés d'une ***Map***.
☐ gère les valeurs d'une ***Map***.
☒ gère les associations d'une ***Map***.
☒ utilise la notion de **généricité**.
☒ est le type des éléments de l'ensemble renvoyé par ***entrySet***() de toute ***Map***.
☒ est une classe interne à l'interface ***Collection***.

16. Dans l'**API Collection** de **Java**, que représente la classe ***Collections ?***

☐ l'**interface mère** au sommet de la hiérarchie de l'**API** .
☒ un conteneur de **méthodes de classes** pour les opérations courantes liées aux collections.
☐ la **classe abstraite** dont héritent toutes les collections concrètes.
☒ une classe proposant des **services publics récurrents et communs** à toutes les collections.

17. Dans l'**API Collection de Java**, que représente la classe ***Arrays ?***

☐ l'**interface mère** au sommet de la hiérarchie de l'**API** pour gérer des tableaux.
☒ un conteneur de **méthodes de classes** pour les opérations courantes liées aux tableaux.
☐ la **classe abstraite** dont héritent toutes les collections modélisant les **tableaux**.
☒ une classe proposant des **services publics récurrents et communs** aux tableaux.

18. Rédigez ou explicitez verbalement à vous ou à votre collègue la notion de **généricité**.

QCM-EXERCICES-JAVA-API-COLLECTIONS

Copyright

Chef de projet (responsable du produit de formation)

PERRACHON Chantal, DIIP Neuilly-sur-Marne

Ont participé à la conception

COULARD Michel, CFPA Evry Ris-Orangis

Réalisation technique

COULARD Michel, CFPA Evry Ris-Orangis

Crédit photographique/illustration

Sans objet

AFPA Février 2014

Association nationale pour la Formation Professionnelle des Adultes
13 place du Général de Gaulle – 93108 Montreuil Cedex

www.afpa.fr