Exercice 1 : QCM :

1. Quelle classe implémente une liste chaînée en Java ?

A. ArrayList
B. LinkedList
C. HashSet
D. TreeMap

2. Quelle structure permet un accès FIFO (First In First Out) ?

A. Stack
B. Queue
C. Deque
D. Map

3. Quelle méthode ajoute un élément à la fin d'une Queue ?

A. push()
B. put()
C. offer()
D. insert()

4. Quelle classe peut être utilisée comme Queue et comme Deque ?

A. HashMap
B. LinkedList
C. TreeSet
D. ArrayList

5. Dans une Deque, quelle méthode ajoute un élément au début ?

A. addFirst()
B. addLast()
C. offerLast()
D. pushBack()

6. Quelle interface représente une collection clé-valeur ?

A. Queue
B. Set
C. Map
D. List

7. Quelle implémentation de Map ne garantit aucun ordre des clés ?

A. TreeMap

B. LinkedHashMap

C. HashMap

D. SortedMap

8. Quelle implémentation de Map trie les clés automatiquement ?

A. HashMap
B. LinkedList
C. TreeMap
D. ArrayDeque

9. Quelle méthode permet de récupérer une valeur dans une Map ?

A. find()
B. get()
C. value()
D. search()

10. Que retourne map.put("a", 5) si la clé "a" existait déjà avec la valeur 2 ?

A. 5
B. true
C. 2
D. null

11. Quelle structure est la plus adaptée pour insérer souvent au début d'une liste ?

A. ArrayList
B. LinkedList
C. HashMap
D. TreeMap

12. Quelle méthode retire l'élément en tête d'une Queue sans erreur si vide ?

A. remove()
B. pop()
C. poll()
D. delete()

13. Une Map peut-elle contenir des clés dupliquées ?

A. Oui
B. Non
C. Seulement TreeMap
D. Seulement HashMap

14. Quelle structure utilise le principe LIFO si on emploie push() et pop() ?

A. Queue
B. Deque
C. Map
D. Set

Exercice 2 : Soit le code JAVA suivant, déterminer les erreurs qui existent dans ce code et comment peut-on les corriger ?

```
class Node {
   int data;
   Node next;

   Node(int data) {
      data = data;
      next = null;
   }}

class LinkedList {
   Node head;

   void insert(int data) {
      Node newNode = new Node(data);

      if (head == null) {
         head.next = newNode;
         return;}

      Node temp = head;
      while (temp != null) {
         temp = temp.next; }

      temp.next = newNode;    }

   void display() {
      Node temp = head;

      while (temp.next != null) {
         System.out.print(temp.data + " -> ");
         temp = temp.next; }

      System.out.println("null");
   }

   void delete(int value) {
      if (head == null) return;

      if (head.data = value) {
```

```
        head = head.next;
        return; }

      Node temp = head;

      while (temp.next != null && temp.next.data != value) {
        temp = temp.next; }

      if (temp.next == null) {
        temp.next = temp.next.next;    } }}

public class Main {
   public static void main(String[] args) {
      LinkedList list = new LinkedList();

      list.insert(10);
      list.insert(20);
      list.insert(30);

      list.display();

      list.delete(20);

      list.display();
   }
}
```

Exercice 3 : Soit le code JAVA suivant, déterminer les erreurs qui existent dans ce code et comment peut-on les corriger ?

```
import java.util.HashMap;

public class Main {
public static void main(String[] args) {

HashMap<String, Integer> notes = new HashMap<>();

notes.put("Alice", 15);
notes.put("Bob", 12);
notes.put("Charlie", "18");

System.out.println(notes.get(0));

if (notes.contains("Alice")) {
System.out.println("Alice existe");}
```

```
int noteDavid = notes.get("David");

for (String nom : notes) {
System.out.println(nom + " : " + notes.get(nom));}

notes.remove(15);

System.out.println("Taille : " + notes.length());}}
```

Exercice 4 : Pour chaque situation suivante, indiquez la structure de données Java la plus adaptée parmi : Map, Queue, Deque, LinkedList et ArrayList.

- Une université veut stocker les étudiants avec leur numéro matricule afin de retrouver rapidement un étudiant à partir de son identifiant.
- Dans une imprimerie, les documents envoyés sont imprimés dans l'ordre d'arrivée.
- Une application de musique permet d'ajouter ou supprimer souvent des chansons au milieu d'une playlist.
- Un éditeur de texte veut gérer les fonctions Annuler / Rétablir (Undo / Redo).
- Une boutique en ligne affiche une liste de produits consultés fréquemment par leur position (produit 0, produit 1, produit 2...).
- Dans un hôpital, les patients passent dans la salle de consultation selon leur ordre d'arrivée.
- Une application doit insérer souvent des éléments au début et à la fin d'une collection.
- Une entreprise veut stocker une liste simple d'employés et accéder rapidement au 50e employé.
- Un programme veut compter combien de fois chaque mot apparaît dans un texte.
- On veut parcourir une liste dans l'ordre uniquement, avec beaucoup de suppressions.