Exercices Java Fondamentaux - Préparation LeetCode

📊 NIVEAU LOW (50 exercices)

Variables & Opérateurs (15 exercices)

1. **Calculatrice basique** : Créer un programme qui effectue les 4 opérations (+, -, *, /) sur deux nombres
2. **Conversion température** : Convertir Celsius vers Fahrenheit et vice versa
3. **Calcul moyenne** : Calculer la moyenne de 3 notes
4. **Aire rectangle** : Calculer l'aire et périmètre d'un rectangle
5. **Swap variables** : Échanger les valeurs de deux variables (avec et sans variable temporaire)
6. **Nombre pair/impair** : Déterminer si un nombre est pair ou impair
7. **Plus grand de 3 nombres** : Trouver le maximum parmi 3 entiers
8. **Calcul intérêts simples** : Calculer les intérêts simples (Capital × Taux × Temps)
9. **Conversion secondes** : Convertir secondes en heures:minutes:secondes
10. **Opérateurs bit à bit** : Utiliser &, |, ^, ~ sur deux nombres
11. **Modulo magic** : Utiliser % pour diverses opérations (dernier chiffre, etc.)
12. **Variables auto-incrémentées** : Comprendre ++i vs i++
13. **Casting types** : Convertir entre int, float, double, char
14. **Constantes** : Utiliser final pour créer des constantes
15. **Opérateurs de comparaison** : Comparer différents types de données

Structures de Contrôle (20 exercices)

16. **FizzBuzz simple** : Afficher nombres 1-100, "Fizz" pour multiples de 3, "Buzz" pour 5
17. **Calculatrice avec menu** : Menu switch pour choisir l'opération
18. **Table multiplication** : Afficher table de multiplication d'un nombre
19. **Nombre premier** : Vérifier si un nombre est premier
20. **Factorielle** : Calculer factorielle avec boucle
21. **Suite Fibonacci** : Générer les N premiers nombres de Fibonacci
22. **Somme chiffres** : Additionner tous les chiffres d'un nombre
23. **Inversion nombre** : Inverser les chiffres d'un nombre (123 → 321)
24. **Palindrome nombre** : Vérifier si un nombre est palindrome
25. **PGCD** : Calculer PGCD de deux nombres (algorithme d'Euclide)

26. **Triangle étoiles** : Dessiner triangle avec * (plusieurs variantes)
27. **Jeu devinette** : Deviner un nombre entre 1-100
28. **Validation saisie** : Valider entrée utilisateur avec while
29. **Menu répétitif** : Menu qui revient jusqu'à choix "Quitter"
30. **Compteur voyelles** : Compter voyelles dans une chaîne
31. **Pattern diamant** : Dessiner diamant avec caractères
32. **Nombre Armstrong** : Vérifier si nombre = somme cubes de ses chiffres
33. **Boucles imbriquées** : Différents patterns avec boucles imbriquées
34. **Break et continue** : Utiliser break/continue dans boucles
35. **Série harmonique** : Calculer 1 + 1/2 + 1/3 + ... + 1/n

Tableaux (15 exercices)

36. **Initialisation tableau** : Différentes façons d'initialiser tableaux
37. **Somme éléments** : Calculer somme tous éléments d'un tableau
38. **Min/Max tableau** : Trouver minimum et maximum
39. **Recherche linéaire** : Chercher élément dans tableau
40. **Inversion tableau** : Inverser ordre des éléments
41. **Copie tableau** : Copier tableau (shallow vs deep copy)
42. **Tri bulle simple** : Implémenter tri à bulles basique
43. **Tableau 2D basique** : Manipuler matrice 2D simple
44. **Moyenne tableau** : Calculer moyenne avec gestion division par zéro
45. **Élément unique** : Trouver éléments qui apparaissent une fois
46. **Rotation tableau** : Faire rotation à droite/gauche
47. **Fusion tableaux** : Fusionner deux tableaux triés
48. **Fréquence éléments** : Compter fréquence chaque élément
49. **Supprimer doublons** : Enlever doublons d'un tableau
50. **Second plus grand** : Trouver deuxième plus grand élément

---

🔥 NIVEAU MEDIUM (40 exercices)

POO Fondamentale (20 exercices)

51. **Classe Personne** : Créer classe avec attributs privés et méthodes
52. **Constructeurs multiples** : Implémenter surcharge constructeurs
53. **Getters/Setters** : Encapsulation avec validation
54. **Méthodes statiques** : Comprendre static vs instance
55. **Classe Compte bancaire** : Dépôt, retrait, solde avec validation
56. **Héritage Animal** : Classe parent Animal, enfants Chat/Chien
57. **Polymorphisme** : Méthode abstraite implémentée différemment
58. **Interface Drawable** : Interface avec méthodes draw()
59. **Classe abstraite** : Forme abstraite avec Rectangle/Cercle
60. **Override toString()** : Surcharger méthode toString
61. **Equals et hashCode** : Implémenter comparaison objets
62. **Composition** : Classe Voiture contient Moteur
63. **Agrégation** : Classe Équipe contient liste Joueurs
64. **Singleton pattern** : Implémenter pattern Singleton
65. **Factory pattern** : Créer objets via Factory
66. **Enum utilisation** : Utiliser énumérations efficacement
67. **Inner classes** : Classes internes et anonymes
68. **Package organisation** : Organiser classes en packages
69. **Access modifiers** : Comprendre public/private/protected/default
70. **This et super** : Utilisation correcte this/super

Tableaux Avancés (10 exercices)

71. **Matrice spirale** : Parcourir matrice en spirale
72. **Transposition matrice** : Transposer matrice carrée
73. **Multiplication matrices** : Multiplier deux matrices
74. **Recherche binaire** : Implémenter recherche binaire
75. **Tri rapide (QuickSort)** : Implémenter algorithme QuickSort
76. **Tableau dynamique** : Créer classe ArrayList simplifiée
77. **Matrice rotation** : Faire rotation 90° d'une matrice

78. **Sous-tableau max** : Trouver sous-tableau avec somme maximale
79. **Tri fusion (MergeSort)** : Implémenter MergeSort
80. **Matrice identité** : Vérifier si matrice est identité

Structures Contrôle Avancées (10 exercices)

81. **Récursion factorielle** : Factorielle récursive vs itérative
82. **Tours Hanoi** : Résoudre Tours de Hanoï récursivement
83. **Backtracking simple** : N-Queens pour petit N
84. **Parcours arbre** : Parcours récursif structure arborescente
85. **Validation parenthèses** : Vérifier parenthèses équilibrées
86. **Génération permutations** : Générer toutes permutations
87. **Labyrinthe simple** : Trouver chemin dans labyrinthe 2D
88. **Calcul combinaisons** : Calculer C(n,k) récursivement
89. **Exponentiation rapide** : Calculer a^n efficacement
90. **Séquence Collatz** : Implémenter conjecture Collatz

🚀 NIVEAU HIGH (30 exercices)

POO Avancée (15 exercices)

91. **Génériques basiques** : Créer classe générique simple
92. **Collections customs** : Implémenter Stack/Queue
93. **Exceptions customs** : Créer et gérer exceptions personnalisées
94. **Builder pattern** : Implémenter pattern Builder
95. **Observer pattern** : Système événements/observateurs
96. **Decorator pattern** : Décorer objets dynamiquement
97. **Strategy pattern** : Différents algorithmes interchangeables
98. **Template method** : Squelette algorithme avec parties variables
99. **Proxy pattern** : Contrôler accès à objets
100. **Clonage profond** : Implémenter clone() correctement
101. **Sérialisation** : Sauver/charger objets
102. **Reflection basique** : Utiliser reflection pour introspection

103. **Annotations** : Créer et utiliser annotations custom
104. **Multithreading** : Threads basiques avec synchronisation
105. **Lambda expressions** : Utiliser lambda avec interfaces fonctionnelles

Algorithmes Complexes (15 exercices)

106. **Arbre binaire** : Implémenter BST avec insertion/recherche
107. **Graphe DFS/BFS** : Parcours profondeur/largeur
108. **Plus court chemin** : Dijkstra simplifié
109. **Hachage** : Table de hachage avec gestion collisions
110. **Trie structure** : Arbre préfixes pour mots
111. **Heap binaire** : Min/Max heap avec opérations
112. **Union-Find** : Structure données Union-Find
113. **Tri topologique** : Tri topologique sur DAG
114. **LCS** : Plus longue sous-séquence commune
115. **Edit distance** : Distance Levenshtein
116. **KMP pattern** : Recherche motif algorithme KMP
117. **Cache LRU** : Implémenter cache LRU
118. **Bloom filter** : Filtre de Bloom basique
119. **Consistent hashing** : Hachage cohérent simplifié
120. **Rate limiter** : Limiteur débit simple

🎯 PROGRESSION RECOMMANDÉE

Semaine 1-2 : Niveau LOW

- **Objectif** : Maîtriser syntaxe et concepts de base
- **Temps** : ~3-4 exercices par jour
- **Focus** : Ne pas passer au niveau suivant tant que tous ne sont pas maîtrisés

Semaine 3-4 : Niveau MEDIUM

- **Objectif** : Comprendre POO et structures données intermédiaires
- **Temps** : ~2-3 exercices par jour
- **Focus** : Bien comprendre l'encapsulation et héritage

Semaine 5-6 : Niveau HIGH

- **Objectif** : Patterns avancés et algorithmes complexes
- **Temps** : ~2 exercices par jour
- **Focus** : Optimisation et bonnes pratiques

Semaine 7+ : LeetCode

- **Prêt pour** : Easy/Medium LeetCode
- **Base solide** : Tous fondamentaux maîtrisés

💡 CONSEILS PRATIQUES

1. **Code propre** : Toujours commenter et nommer variables clairement
2. **Tests** : Tester chaque exercice avec plusieurs cas
3. **Optimisation** : Après solution qui marche, chercher à optimiser
4. **Documentation** : Écrire javadoc pour méthodes complexes
5. **Git** : Versionner solutions pour suivre progression
6. **Review** : Revenir sur anciens exercices régulièrement

Cette progression vous donnera une base solide pour attaquer LeetCode avec confiance !