

## Formation Préparation de la certification Java SE 8 Programmer II (1Z0-809)

<b>Durée :</b>	2 jours
<b>Public :</b>	Développeurs et analystes programmeurs Java
<b>Pré-requis :</b>	Connaissances en programmation orientée objet Java
<b>Objectifs :</b>	Découvrir le cursus de certification Oracle Java ' Préparer la certification Java SE 8 Programmer I : comprendre le type et le format des questions, s'entraîner à répondre à des questions, révision des thèmes abordés dans l'examen.
<b>Sanction :</b>	Attestation de fin de stage mentionnant le résultat des acquis
<b>Taux de retour à l'emploi:</b>	Aucune donnée disponible
<b>Référence:</b>	JAV1107-F
<b>Note de satisfaction des participants:</b>	Pas de données disponibles

### Présentation de la certification

Cursus de certification Oracle

Java SE 8 Programmer : type d'examen, nombre de questions/durée, % minimal de réussite

Plateforme de certification

Considérations lors de l'examen : missing packages, import, fichiers, chemins, fragments de code, commentaires

**Atelier :** présentation de l'examen et des sujets couverts par le questionnaire

### Révision du contenu de la certification

- **Implémentation de classes :** encapsulation, héritage, polymorphisme, redéfinition des méthodes hashCode>equals/toString, codage d'un singleton, blocs static.

- **Notions avancées de l'objet :** abstraction, mot-clé final, classes internes, static, anonymes, types enum, implémentation/héritage d'interfaces, création et utilisation d'expressions lambda.

- **Généricité et collections :** création et utilisation de classes génériques, collections génériques (ArrayList, TreeSet, TreeMap, ArrayDeque), Utilisation de comparateurs Comparable/Comparator, Streams et filtres de collections, itération de streams et lists, interface Stream, filtres de collections avec des expressions lambda, références de méthodes avec Streams.

- **Interfaces fonctionnelles** : package java.util.function, implémentation d'interfaces fonctionnelles (primitive, binary, unary).
- **Java Stream API** : extraction de données (peek, map), recherches (findFirst, findAny, anyMatch, allMatch, noneMatch), classes optionnelles, tri de collections avec Stream API, méthodes de collecte de résultats, usage de flatMap().
- **Exceptions et assertions** : utilisation du try/catch et du throw, multi-catch/finally, AutoClose (try-with-resources), création d'exceptions et d'auto-closeable resources, utilisation d'assertions.
- **API Time** : Gestion des dates/heures : LocalDate, LocalTime, Instant, Period et Duration, usage des timezones et formatage de dates, TemporalUnit.
- **Java IO et NIO2** : Lecture et écriture depuis la console, utilisation du package java.io, nouveautés du package nio : Path, Files, Stream API avec NIO.2
- **Concurrence en Java** : Implémentation de threads (Runnable, Callable, ExecutorService), identification de problèmes (deadlock, starvation, livelock, race conditions), contrôle et synchronisation (synchronized, package java.util.concurrent.atomic), collections concurrentes (java.util.concurrent), parallel Fork/join, parallel Streams (reduction, decomposition, merging processes, pipelines).
- **Accès aux bases de données avec JDBC** : Interfaces de JDBC (Driver, Connection, Statement, ResultSet) et implémentations, composants nécessaires de connexions, écriture de requêtes et traitement de résultats.
- **Localisation** : object Local, manipulation de fichiers properties, création de ressources bundles et chargement.

## **Passage et correction d'un test blanc**

### **Ateliers corrigés**

### **Questions/réponses, gestion du temps lors du passage de l'examen**