Java – Optimisations de code

Programme *(Mis à jour le 24/04/2025)*

Quelques rappels fondamentaux

- L'environnement Java SE (Java Standard Edition) :
  Concepts fondamentaux : ClassLoader, JIT, GC, Gestionnaire d'exceptions, …
  JVMTI (Java Virtual Machine Tools Interface)
  Quelques outils du J2SE : javap, …
- Notion de byte code :
  Utilisation du désassembleur javap
  Notions de pile et de tas (Stack et Heap)
  Mécanisme de déclarations : variables locales et attributs
  Appels de méthodes en assembleur
- Quelques considérations sur les performances :
  Temps d'exécution VS taille de l'exécutable
  Génie logiciel VS Performances

Développement d'agents JVMTI

- Concepts fondamentaux de JVMTI :
  Les différentes versions l'interface JVMTI
  Les évènements JVMTI
- Codage d'agents JVMTI :
  Traquer l'activité du ClassLoader
  Traquer l'activité en termes d'allocations dynamiques
  Traquer l'activité en termes d'appels de méthodes

« Guide lines » de programmation

- Correctement utiliser les Logger (Log4J, …)
- Fonctionnement de l'instruction switch
- Limitation d'instances temporaires
- Correctement utiliser les chaînes de caractères
- Utilisation des collections Java :
  Choisir les bons algorithmes faces aux besoins
  Collections synchronisées ou non (Vector vs ArrayList, …)
  Les différentes techniques de parcours d'une collection
  Problématiques de l'autoboxing et du unboxing
- Utilisation de tableaux Java typés :
  Accès indexés aux valeurs
  Traitements des valeurs de types primitifs
- Encapsulation VS performance

Le garbage collector (GC)

- Aspects fondamentaux :
  Tâches du garbage collector : libération et défragmentation
  Les différents algorithmes utilisés
- Fonctionnement du Garbage Collector de la JVM Hotspot (Oracle) :
  Monitorer l'activité du GC
  Collectes mineures et collectes majeures
- Comparaison Hotspot/OpenJ9
- Paramétrage du Garbage Collector de la JVM HotSpot :
  Gestion de la taille des Heap (-Xmx, -Xms, -XX:NewRatio, –XX:SurvivorRatio, …)
  Libération incrémentale des ressources
- Les évolutions du GC au fil des versions de Java

Référence
THIL3412

Durée
3 jours / 21 heures

Prix HT / stagiaire
2100€

Objectifs pédagogiques

- Analyser la consommation des ressources engendrées par une modélisation et par l'implémentation de cette dernière
- Analyser les mécanismes de bas niveaux de la JVM permettra de mieux ressentir les différents concepts présentés
- Définir les outils graphiques de monitoring et de profiling d'applications Java

Niveau requis

- Garantir maîtriser les bases du langage Java

Public concerné

- Développeurs JAVA, chef de projet et consultant Java

Formateur

Les formateurs intervenants pour Themanis sont qualifiés par notre Responsable Technique Olivier Astre pour les formations informatiques et bureautiques et par Didier Payen pour les formations management.

Conditions d'accès à la formation

Délai : 3 mois à 1 semaine avant le démarrage de la formation dans la limite des effectifs indiqués

Moyens pédagogiques et techniques

Salles de formation (les personnes en situation de handicap peuvent avoir des besoins spécifiques pour suivre la formation. N'hésitez pas à nous contacter pour en discuter) équipée d'un ordinateur de dernière génération par stagiaire, réseau haut débit et vidéo-projection UHD
Documents supports de formation projetés
Apports théoriques, étude de cas concrets et exercices
Mise à disposition en ligne de documents supports à la suite de la formation

Dispositif de suivi de l'éxécution de l'évaluation des résultats de la formation

Feuilles d'émargement (signature électronique privilégiée)
Evaluations formatives et des acquis sous forme de questions orales et/ou écrites (QCM) et/ou mises en situation
Questionnaires de satisfaction (enquête électronique privilégiée)

Recyclons les ressources de la JVM

- Utilisation d'un pool d'instances :
  Implémentation d'un pool d'instances
  Utilisation d'un pool d'instances
  Etude du comportement du GC via un agent JVMTI
- Application multithreadées :
  Monitoring via JVMTI sur l'utilisation des threads
  Le package *java.util.concurrent*
  Utilisation de pool de threads
  Gestion de la synchronisation de vos threads
- SoftReferences et WeakReferences

Outils graphiques de monitoring et de profiling

- La JConsole :
  Présentation du modèle JMX (Java Monitoring eXtensions)
  Lancement et attachement de la JConsole
  Les différentes catégories d'informations collectées
- Visual GC et GC Viewer :
  Les différences entre les deux outils
  Analyses poussées de l'activité du GC
- Java VisualVM :
  Présentation de l'outil
  Profilling de type CPU
  Profilling de type Memory
- Autres outils de profilage :
  Utilisation du plugin Eclipse Memory Analyser Tool
  JProfiler