Oracle Java SE 8 Programmer II

Objectifs

- Utiliser les API de programmation avancée de la plate-forme Java
- Écrire des programmes accédants aux bases de données
- Mettre en œuvre la programmation parallèle par l'utilisation des Threads
- Ecrire des programmes manipulant les entrées/sorties et l'accès aux ressources réseau
- Implémenter efficacement un système de journalisation dans une application Java
- Concevoir des interfaces graphiques avancées
- Externaliser les chaînes de caractères des codes sources et mettre en œuvre l'internationalisation des applications
- Superviser une application Java avec JMX

Prérequis

- Maitriser le langage Java

Programme

Accès aux bases de données

- Présentation de l'API JDBC (Java DataBase Connectivity)
- Notion de pilote/fournisseur JDBC
  - Présentation des différents types de pilotes JDBC
  - Utiliser et intégrer un pilote JDBC dans son projet Java sous Eclipse
- Utilisation des classes du package JDBC standard : java.sql
  - Utilisation du driver et connexion au système
  - Formuler des requêtes SQL au travers des différents types d'interfaces (Requêtes simples, pré compilées, procédures stockées)
  - Exploiter les résultats
- Méthodologies pour l'écriture des programmes d'accès aux bases de données
  - Gestion efficace des exceptions
  - Libération des ressources
- Obtenir des informations sur le système de base de données
  - Les interfaces de gestion des MetaData
- Les transactions
  - Utiliser les transactions JDBC
  - Présentation du concept de transactions distribuées
- Utiliser les fonctionnalités de l'IDE Java pour faciliter l'accès et la visualisation des données d'un SGBDR
- Travaux pratiques :
  - Conception d'une base de données
  - Développement d'une couche d'accès aux données

Programmation multitâches

- La classe java.lang.Thread et l'interface java.lang.Runnable
- Structure d'un programme multithread

    - Organisation des méthodes
    - Résolution des problématiques d'accès concurrentielles
- Utilisation des groupes de thread
- Synchronisation et exclusion mutuelle
    - Utilisation de méthodes et de blocs synchronisés
- Utilisation de l'API de concurrence
    - Les exécuteurs
    - Les queues
    - Les Map atomiques
    - Les synchroniseurs
    - Les verrous
- Travaux pratiques :
    - Création d'un thread dédié pour la récupération de données en masse depuis la base de données

Les entrées/sorties

- Présentation des classes du package java.io et java.nio
- Lecture et écriture de flux de données
    - Lecture/écriture de données binaires
    - Lecture/écriture de données textes
- Utilisation des entrées/sorties pour la sérialisation d'objets Java
    - Principes de la sérialisation de données
    - Création d'objets Java sérialisables
    - ObjetInputStream et ObjectOutputStream
- L'API NIO2 pour la gestion des fichiers et des systèmes de fichiers
    - L'interface java.nio.file.Path pour simplifier l'accès aux fichiers
    - La classe utilitaire à tout faire : Files
    - DirectoryStream et FileVisitor pour parcourir les dossiers et fichiers d'une arborescence
- Travaux pratiques :
    - Création d'une configuration pour l'application
    - Mise en place de la sérialisation/désérialisation de la configuration dans un fichier

Programmation réseau

- Le package java.net
- Utilisation des sockets (Clients et serveurs)
- La classe URL
- Utilisation des threads et des sockets serveurs
- Travaux pratiques :
    - Création d'un serveur réseau pour la partie métier de l'application
    - Préparation d'une couche de communication cliente pour la partie IHM de l'application

Les librairies de journalisation Java

- L'intérêt de la journalisation dans les applications logicielles
- Les différentes approches Java

    - Les classes du package java.util.logging
    - L'API Commons Logging
    - L'API Log4J
- Implémentation d'un système de journalisation
    - Définition des stratégies de journalisation
    - Identification des destinations de message
    - Filtrage
- Travaux pratiques :
    - Mise en place de Log4J dans l'application
    - Génération de diverses traces

Conception d'interfaces graphiques avec Swing

- Présentation des API Swing et AWT
    - Différences, avantages et inconvénients
- Modèle de conception des interfaces
    - Les conteneurs et panneau
    - Les gestionnaires de positionnement (Layout Manager) et le positionnement libre
- Gestion des événements des applications
    - Les classes et interfaces de gestion événementielle
    - Écriture de gestionnaires (Classes imbriquées)
- Utilisation d'un concepteur graphique pour la réalisation des interfaces graphiques
- Travaux pratiques :
    - Conception d'une interface graphique complète pour l'application
    - Mise en place de la liaison avec la partie serveur
    - Utilisation du multitâches pour l'affichage graphique de la progression des traitements

Internationalisation des applications

- Externalisation des chaînes de caractères
- Utilisation des classes ResourceBundle et locale
- Conception des fichiers properties pour le stockage des chaînes
- Travaux pratiques :
    - Externalisation de toutes les chaines de caractères de l'application avec les fonctionnalités de l'IDE
    - Internationalisation Anglais/Français de l'application
    - Adaptation de l'interface graphique pour la prise en charge de l'internationalisation

Gestion et supervision des applications Java avec JMX

- Présentation de l'architecture de JMX
    - Les possibilités offertes par JMX
    - Les MBeans, le MBeanServer, les connecteurs…
- Ajouter le support de JMX à une application Java
    - Introduction au développement JMX
- Superviser une application Java
    - Localement et à distance

- Activer le support de JMX dans la JVM
- Utiliser un outil de supervision

- Travaux pratiques :
  - Mise en place de JMX dans l'application
  - Exposition de métriques sensibles via JMX
  - Exploitation des informations avec un client JMX