Oracle Java SE 8 Programmer I

Objectifs

- Ecrire, compiler, exécuter et déboguer des programmes Java
- Utiliser l'IDE Eclipse pour vos projets Java
- Appliquer les concepts de programmation orientée objet au langage Java
- Créer des classes et les implémenter avec des attributs et des méthodes
- Mettre en œuvre l'encapsulation
- Appliquer les mécanismes d'héritage et de polymorphisme, redéfinir et surcharger des méthodes
- Utiliser les classes abstraites et les interfaces
- Structurer les applications en package et gérer correctement les imports de classes et de méthodes
- Utiliser la bibliothèque de classes Java

Prérequis

- Développer des programmes dans un langage de programmation structuré

Programme

Introduction

- Historique de Java
- L'écosystème Java
  - Cas d'utilisation de Java
  - Java dans le paysage informatique
- Principes et caractéristiques de Java
  - L'indépendance par rapport à la plateforme
  - Un langage orienté objet, sûr, robuste et performant
- Le développement Java
  - Cycle de conception d'une application Java
  - Les outils de développement du JDK (compilateur, interpréteur, débogueur)
  - La machine virtuelle Java
  - Structure d'un programme Java

Utilisation de l'IDE Eclipse

- Présentation d'Eclipse
  - Les différentes éditions de l'IDE
  - Les perspectives, éditeurs et vues
- Programmer avec Eclipse
  - Création et configuration des projets
  - Utilisation des assistants de création (wizards)
  - Compiler et exécuter un programme
  - Utiliser le débogueur pour la mise au point des programmes
- Travaux pratiques :
  - Prise en main d'Eclipse, premier projet et exécution de code

Les principes de base du langage

- Les règles syntaxiques
  - Les instructions et les blocs
  - Les identificateurs
  - Utilisation des commentaires Javadoc pour la génération de la documentation
  - Les constantes littérales et différentes expressions littérales
  - Le formatage des expressions numériques
- Les opérateurs et expressions
- Les variables et les constantes
- Les importations de classes et packages
- Les importations statiques de constantes
- Les types de données primitifs et les types wrappers
  - Entiers, réels, caractère et booléen
  - Autoboxing des types primitifs
- Les chaînes de caractères et la classe String
  - Les principales méthodes de manipulation de chaînes de caractères
  - L'opérateur de concaténation et la classe StringBuffer
  - Les « text blocks » (Java 14)
- Création et utilisation de types de données énumérés : enum
- Les tableaux
  - Création et manipulation de tableaux à une ou plusieurs dimensions
  - Utilisation des méthodes utilitaires de la classe Arrays
- Conversion de types de données primitifs
- Les structures de contrôle
  - Conditionnelles (if, switch)
  - Itératives (for, for each, while, do)
  - L'utilisation de switch avec le type String
  - Les mots clés break, continue et return
- Affichage sur la sortie standard avec System.out.println()
- Affichage formaté sur la sortie standard avec la méthode printf()
- Travaux pratiques :
  - Calculer le jour de Noël en fonction d'une année

La programmation orientée objet en Java

- Les principes de la programmation orientée objet
  - Modélisation et conception objet
- Les concepts de programmation objet appliqués à Java
- Les classes, les objets, les attributs et les méthodes
  - L'encapsulation
  - L'héritage
  - L'abstraction
  - Le polymorphisme
- Relation entre les classes et les objets

Création et manipulation de classes et d'objets

- Déclaration d'une classe
- Création d'objets avec l'opérateur new et notion de référence

- Déclaration des constructeurs et règles de mise en œuvre
- Finalisation d'objet et le garbage collector
- Déclaration et manipulation de membres (variables et méthodes) de classes (static) et d'instances
- Les méthodes et le passage de paramètres par valeur ou référence
- Bonnes pratiques pour la mise en œuvre des accesseurs
- Mise en œuvre de l'héritage simple en Java
- Le cas particulier des classes finales et méthodes finales
    - Impact sur l'héritage
    - Cas d'utilisation des classes finales
- Le cas particulier les classes abstraites
    - Impact sur l'héritage
    - Cas d'utilisation des classes abstraites
- Effectuer des conversions d'objets
- Les modificateurs d'accès et l'accès aux membres des classes
- Mise en œuvre de la surcharge de méthodes
- Simplification de la surcharge de méthodes par la réalisation de méthodes à arguments variables
- Mise en œuvre de la redéfinition de méthodes
- La classe Object et ses méthodes utilitaires
    - Cloner les objets avec la méthode clone() et l'interface Cloneable
    - Comparaison d'objet avec la méthode equals()
    - Obtenir une représentation d'un objet sous forme de chaîne de caractères avec la méthode toString()
    - La réflexion objet avec la méthode getClass()
- Utilisation des mots clés this, this() et super, super() et patterns de mises en œuvres
- Tester le type d'un objet avec l'opérateur instanceof et pattern de mise en œuvre
- Travaux pratiques :
    - Réalisation progressive d'une application de gestion de comptes bancaires avec conception des classes : Compte, CompteEpargne, Client, Banque

Concepts avancés de programmation Java

- Les classes abstraites et les interfaces
    - Points communs et divergences
    - Utilisation des interfaces Cloneable pour cloner des objets et Comparable pour trier des tableaux/collections d'objets
    - Création et utilisation de classes abstraites et d'interfaces et mise en œuvre du polymorphisme
    - Cas particulier des interfaces fonctionnelles
    - Les interfaces de java.util.function
- Les « Records » (Java 14)
    - Caractéristiques et cas d'usage
- Les classes scellées (Java 17)
    - Comment limiter une hiérarchie d'héritage
    - Le mot clé « sealed »
- Les expressions Lambdas
    - Principes et syntaxe
    - Application aux interfaces fonctionnelles de l'API standard
- Création et utilisation de ses propres packages

  - Convention et règle de nommage
- Principe de fonctionnement et traitement des erreurs avec les exceptions
  - Les classes Throwable, Error et Exception
  - Les méthodes communes issues de la classe Throwable : getMessage(), getCause(), …
  - Intercepter et gérer les exceptions avec les instructions try, catch, finally
  - Déclarer des méthodes comme étant susceptibles de lever les exceptions avec l'instruction throws
  - Déclencher des exceptions avec l'instruction throw
  - Le chaînage d'exceptions : bonnes pratiques et cas d'utilisations
  - Créer ses propres classes d'exception
  - Gestion automatique de la fermeture des ressources avec try-with-ressource
  - Multicatch et simplification de la gestion des exceptions
- La méta-programmation par annotations
  - Déclaration, utilisation et syntaxe des annotations
  - Annotations standards : @Deprecated, @Overrides, @SuppressWarning, …
- Les classes internes et anonymes
  - Cas d'utilisations et bonnes pratiques de mise en œuvre
- Les modules
  - Principes et avantages des modules
  - Organisation des projets en module
  - Syntaxe de déclaration et exportation de packages
  - La notion de « module path » et les changements en termes de déclaration de dépendances
  - L'impact sur le JDK : réorganisation
- Travaux pratiques :
  - Mise en œuvre de la gestion des erreurs dans la classe Compte et la classe Banque
  - Trier les comptes en banque selon différents critères

La bibliothèque de classes Java

- La gestion des dates et du temps
  - L'API historique et les classes java.util.Date et java.util.Calendar
  - L'API Date (java.time)
  - Gestion des différences entre les dates
  - Méthodes de conversions de formats entre la nouvelle API et l'ancienne et vice-versa
- Les collections
  - Inconvénients des tableaux et avantages apportés par l'utilisation des collections
  - Les différents types de collections : les Set, les List, les Map
  - Création et manipulation de ArrayList et de HashMap
  - Utilisation des méthodes utilitaires de la classe Collections
- Les collections génériques
  - Le meilleur des tableaux et des collections
  - Utilisation des génériques pour typer les collections
  - Simplification de l'utilisation des Generics avec le « Diamond operator »
  - Utilisation des méthodes d'initialisation des collections
- Manipulation des collections avec les Streams

- Travaux pratiques :
  - Compléter la classe Banque avec des méthodes de recherche de compte en utilisant les streams et les expressions lambdas