Glossaire Java pour BTS SIO

Introduction

Objectif : Comprendre le vocabulaire de base de Java

Public cible : Étudiants en BTS SIO

Pourquoi ce glossaire ?

1. Prise en main de l’environnement Java & Eclipse

- IDE: Environnement de développement intégré (éditeur, compilateur, débogueur)
- Eclipse: IDE open-source populaire pour Java
- JDK: Kit de développement Java (JVM, compilateur, outils)
- JRE: Environnement d'exécution Java (JVM + bibliothèques)

1. Prise en main (suite)

- Workspace: Dossier Eclipse pour vos projets
- Projet Java: Structure organisant le code (src, bin, lib)
- Package: Organisation logique des classes (dossiers)
- Classe publique: Fichier .java avec la méthode main()

1. Prise en main (fin)

- main: Point d'entrée du programme (public static void main(String[] args))
- Compilation: Conversion .java -> .class
- Bytecode: Code intermédiaire exécuté par la JVM
- Exécution: Lancement du bytecode
- Débogage: Exécution pas à pas pour trouver les erreurs
- Refactoring: Amélioration du code sans changer son fonctionnement
- Javadoc: Génération de documentation à partir des commentaires
- PATH/JAVA_HOME: Variables pour trouver Java

2. Les bases de Java

- Type primitif: Donnée de base (int, double, boolean, char)
- byte, short, int, long: Entiers signés de différentes tailles
- float, double: Nombres à virgule flottante
- char: Caractère (lettre, chiffre, symbole)
- boolean: Vrai ou faux (true/false)

2. Les bases (suite)

- Variable: Nom pour stocker une valeur
- Initialisation: Donner une valeur à une variable
- final: Mot-clé pour une constante (valeur ne change pas)
- Opérateurs arithmétiques: +, -, *, /, %
- Opérateurs de comparaison: ==, !=, <, >, <=, >=
- Opérateurs logiques: &&, ||, !
- Incrémentation/décrémentation: ++, --

2. Les bases (fin)

- Instruction if/else: Choix selon une condition
- switch: Choix multiple
- Boucles for, while, do...while: Répéter des actions
- Tableau 1D: Liste d'éléments du même type
- length: Taille d'un tableau
- Méthode: Fonction dans une classe
- Surcharge (overloading): Méthodes avec le même nom, signatures différentes
- void: Pas de valeur de retour
- return: Sortir d'une méthode et renvoyer une valeur
- Portée (scope): Où une variable est visible

3. Introduction à la notion d’objet

- Classe: Plan pour créer des objets (attributs et méthodes)
- Objet: Instance d'une classe
- Attribut (field): Variable d'un objet (son état)
- Méthode: Action que peut faire un objet
- Constructeur: Méthode pour créer un objet
- Instanciation: Création d'un objet avec 'new'

3. Notion d'objet (suite)

- this: Référence à l'objet courant
- Getter: Méthode pour lire un attribut (getX())
- Setter: Méthode pour modifier un attribut (setX())
- Encapsulation: Cacher les attributs et utiliser getters/setters
- Serializable: Interface pour transformer un objet en octets
- enum: Liste de constantes (ex: ItemType)
- final (attribut): Attribut qui ne peut plus être modifié
- toJSON(): Méthode pour convertir un objet en JSON

4. Approfondissement sur les classes

- Constructeur: Méthode pour initialiser un objet (même nom que la classe)
- Surcharge: Plusieurs constructeurs avec des signatures différentes
- this(): Appeler un autre constructeur de la même classe
- public, private, protected: Visibilité des attributs/méthodes
- Encapsulation: Cacher les attributs et contrôler l'accès

4. Approfondissement (suite)

- Getter: Méthode pour lire un attribut
- Setter: Méthode pour modifier un attribut (avec validation)
- static (variable/méthode): Appartient à la classe (pas à l'objet)
- final (attribut): Constante (ne peut pas être modifié)
- serialVersionUID: Numéro de version pour la sérialisation

5. Interaction entre objets

- Association: Lien entre des classes (utilisation)
- Composition: Un objet contient d'autres objets (fort)
- Agrégation: Un objet utilise d'autres objets (faible)
- Paramètre: Variable reçue par une méthode
- Pass-by-value (référence): Java passe une copie de la référence
- Retour d'objet: Méthode renvoyant un objet
- null: Pas d'objet
- boucle for-each: Parcourir facilement des collections
- UML: Diagramme pour représenter les classes
- Multiplicité: Nombre d'objets dans une relation UML

6. Héritage en Java

- Héritage: Une classe hérite d'une autre (réutilisation)
- extends: Mot-clé pour l'héritage
- super-classe: Classe dont on hérite
- sous-classe: Classe qui hérite
- surcharge (overloading): Méthodes mêmes noms, signatures différentes
- redéfinition (override): Changer une méthode héritée
- @Override: Vérifier qu'on redéfinit bien une méthode
- super(...): Appeler le constructeur de la super-classe
- super.methode(): Appeler une méthode de la super-classe
- Polymorphisme: Utiliser des objets de différentes classes de la même manière
- Upcasting: Convertir vers la super-classe
- Downcasting: Convertir vers la sous-classe

7. Classes abstraites et interfaces

- Abstraction: Simplifier en cachant les détails
- abstract: Mot-clé pour classe/méthode abstraite
- Classe abstraite: Ne peut pas être instanciée
- Méthode abstraite: Pas de code (à implémenter)
- Interface: Contrat (méthodes à implémenter)
- implements: Mot-clé pour implémenter une interface
- default (interface): Méthode avec code dans une interface
- static (interface): Méthode utilitaire dans une interface
- Comparable<T>: Interface pour comparer des objets
- Polymorphisme: Utiliser des interfaces pour code générique
- @Override: Redéfinir une méthode héritée ou d'interface

8. Collections et objets

- Collection: Structure pour grouper des éléments (List, Set, Map)
- List<E>: Liste ordonnée avec doublons (ArrayList, LinkedList)
- ArrayList: Liste basée sur un tableau
- Set<E>: Collection sans doublons (HashSet, TreeSet)
- Map<K,V>: Association clé-valeur (HashMap, TreeMap)
- add(), remove(): Ajouter et supprimer des éléments
- get(i), size(): Accéder par index et taille
- clear(): Vider la collection
- contains(): Vérifier si un élément est présent
- toArray(): Convertir en tableau
- for-each: Parcourir facilement une collection
- for indexé: Parcourir par indice pour List

9. Les exceptions

- Exception: Erreur qui interrompt le programme
- RuntimeException: Exception non vérifiée
- Exception (checked): Exception vérifiée (à gérer)
- try: Bloc de code qui peut lancer une exception
- catch: Bloc pour gérer une exception
- finally: Bloc exécuté toujours (nettoyage)
- throw: Lancer une exception
- throws: Indiquer qu'une méthode peut lancer une exception
- Exception personnalisée: Exception créée par l'utilisateur
- Stack trace: Liste des appels de méthodes lors d'une exception

10. Structuration d’un projet objet complet

- Workspace: Dossier pour les projets Eclipse
- Package: Dossier logique pour les classes
- MVC: Séparer le code en Model-View-Controller
- Model: Données (classes Java)
- View: Interface utilisateur (Swing, console)
- Controller: Lien entre Model et View
- Refactoring: Améliorer le code
- Test unitaire: Tester le code automatiquement (JUnit)
- EGit: Plugin Eclipse pour Git
- JAR exécutable: Archive Java exécutable

11. Introduction à la Javadoc

- Javadoc: Outil pour créer de la documentation HTML
- Commentaire JavaDoc: /** ... */
- @param: Décrire un paramètre
- @return: Décrire la valeur renvoyée
- @throws: Lister les exceptions
- @see: Référencer une autre classe/méthode
- @author: Indiquer l'auteur
- @version: Indiquer la version

Conclusion

- Ce glossaire vous aidera à comprendre le vocabulaire Java.
- N'hésitez pas à le consulter régulièrement.
- Bon courage dans votre apprentissage de Java !