Chapitre 1 : Objets et classes
Résumé des concepts

- **objet** Les objets Java modélisent les objets d'un problème donné.
- **classe** Les objets sont créés à partir de classes. La classe décrit le type d'objet ; les objets représentent des **instances** individuelles de la classe.
- **méthode** Il est possible de communiquer avec des objets en invoquant leurs méthodes. Cela amène généralement les objets à réaliser une action.
- **paramètre** Les méthodes peuvent disposer de paramètres qui permettent de leur fournir des informations supplémentaires pour une tâche déterminée.
- **signature** L'en-tête d'une méthode définit sa signature. La signature indique les informations nécessaires pour invoquer cette méthode.
- **type** Les paramètres possèdent un type. Le type définit les catégories de valeurs que peut prendre un paramètre.
- **instances multiples** Il est possible de créer plusieurs objets similaires à partir d'une classe unique.
- **état** Les objets possèdent un état. L'état est représenté par les valeurs stockées dans les champs.
- **appel de méthode** Les objets peuvent communiquer en appelant les méthodes d'autres objets.
- **code source** Le code source d'une classe détermine la structure et le comportement (les champs et les méthodes) de chacun des objets de cette classe.
- **résultat** Les méthodes peuvent renvoyer des informations sur un objet par le biais d'une valeur de retour.

Chapitre 2 : Comprendre les définitions de classes
Résumé des concepts

- **création d'objet** Certains objets ne peuvent pas être construits sans que des informations supplémentaires soient fournies
- **champ** Les champs stockent des données qui seront utilisées par un objet. Ils sont aussi appelés **attributs** ou **variables d'instance**.
- **commentaire** Les commentaires sont placés dans le code source d'une classe pour fournir des explications aux lecteurs humains. Ils n'ont aucun effet sur le fonctionnement de la classe.
- **constructeur** Les constructeurs permettent de configurer correctement chaque objet lors de sa création.
- **portée** La portée d'une variable définit la partie du code source d'où l'on peut accéder à la variable.
- **durée de vie** La durée de vie d'une variable correspond à la période pendant laquelle elle continue à exister avant sa destruction.
- **affectation** Les instructions d'affectation stockent la valeur présente du côté droit de l'instruction dans la variable dont le nom est du côté gauche.
- **méthode** Les méthodes sont constituées de deux parties : un en-tête et un corps.
- **accesseur** Les accesseurs sont des méthodes qui renvoient des informations sur l'état d'un objet.
- **mutateur** Les mutateurs (ou **modificateurs**) sont des méthodes qui modifient l'état d'un objet.
- **println** La méthode `System.out.println()` affiche son paramètre dans le terminal.
- **conditionnel** Une instruction conditionnelle choisit une des deux actions possibles à partir du résultat d'un test.
- **expression booléenne** Les expressions booléennes peuvent avoir une seule des deux valeurs : `true` (*vrai*) et `false` (*faux*). Elles permettent de choisir parmi les deux chemins dans une instruction conditionnelle.
- **variable locale** Une variable locale est une variable déclarée et utilisée dans une seule méthode. Sa portée et sa durée de vie sont limitées à celles de la méthode.

Chapitre 3 : Interactions entre objets
Résumé des concepts

- **abstraction** L'abstraction est la capacité d'ignorer les détails pour se concentrer sur un niveau supérieur du problème.
- **modularisation** La modularisation est le processus de division d'un tout en parties bien définies qui peuvent être élaborées et étudiées séparément et qui interagissent de façons bien définies.
- **les classes définissent des types** Le nom d'une classe peut être utilisé comme type pour une variable. Les variables dont le type est une classe permettent de stocker des objets de cette classe.
- **diagramme de classes** Le diagramme de classes représente les classes d'une application et leurs relations. Il donne des informations sur le code source et présente la vue statique d'un programme.
- **diagramme d'objets** Le diagramme d'objets montre les objets et leurs relations à un instant donné au cours de l'exécution de l'application. Il présente une vue dynamique d'un programme.
- **références aux objets** Les variables de type objet stockent des références aux objets.
- **type primitif** Les types primitifs en Java sont les types non objet. Les types primitifs **`int`**, **`boolean`**, **`char`**, **`double`**, **`long`** sont les plus communs. Les types primitifs n'ont pas de méthodes.
- **création d'objets** Les objets peuvent créer d'autres objets, en utilisant l'opérateur **`new`**.
- **surcharge** Une classe peut contenir plusieurs constructeurs ou plusieurs méthodes de même nom, à condition que chacun possède un ensemble distinct de types de paramètres.
- **appel de méthode interne** Les méthodes peuvent appeler d'autres méthodes de la même classe dans le cadre de leur implémentation. Il s'agit d'un appel de méthode interne.
- **appel de méthode externe** Les méthodes peuvent appeler des méthodes d'autres objets en utilisant la notation pointée. Il s'agit d'un appel de méthode externe.
- **débogueur** Un débogueur est un outil logiciel qui aide à étudier la manière dont s'exécute une application. Il peut être utilisé afin de retrouver des bugs.

Chapitre 4 : Groupement d'objets
Résumé des concepts

- **collection** Un objet collection peut stocker une quantité quelconque d'autres objets.
- **boucle** Une boucle peut être utilisée pour exécuter un bloc d'instructions à plusieurs reprises sans qu'il soit utile de les écrire plusieurs fois.
- **itérateur** Un itérateur est un objet qui fournit des fonctionnalités pour traiter de manière itérative (répétitive) tous les éléments d'une collection.
- **null** Le mot-clé Java réservé `null` est utilisé pour signifier « pas d'objet » quand une variable objet ne fait actuellement pas référence à un objet particulier. Un champ qui n'a pas été explicitement initialisé contiendra la valeur `null` par défaut.
- **tableau** Un tableau est une sorte de collection spéciale capable de stocker un nombre fixe d'éléments.

Chapitre 5 : Comportements plus complexes
Résumé des concepts

- **bibliothèque Java** La bibliothèque de classes standard Java contient de nombreuses classes très utiles. Il est important de savoir l'utiliser.
- **documentation de la bibliothèque** La documentation de la bibliothèque de classes Java présente les détails de toutes les classes de la bibliothèque. Utiliser cette documentation est essentiel pour faire un bon usage des classes de la bibliothèque.
- **interface** L'interface d'une classe décrit ce que fait la classe et la manière de l'utiliser sans montrer l'implémentation.
- **implémentation** La totalité du code source qui définit une classe s'appelle *l'implémentation* de cette classe.
- **immuable** Un objet est dit *immuable* (ou *inaltérable*) si son contenu ou son état ne peuvent être modifiés une fois qu'il a été créé. Les chaînes de caractères sont un exemple d'objets immuables.
- **association** Une association (*map*) est une collection qui stocke des paires clé/valeur comme entrées. Il est possible de rechercher les valeurs en fournissant la clé.
- **ensemble** Un ensemble (*set*) est une collection qui stocke chaque élément au maximum une fois. Il ne maintient aucun ordre particulier.
- **documentation** La documentation d'une classe doit être suffisamment détaillée pour que d’autres programmeurs puissent utiliser la classe sans avoir besoin de lire l'implémentation.
- **modificateurs d'accès** Les modificateurs d'accès définissent la visibilité d'un champ, d'un constructeur ou d'une méthode. Les membres publics sont accessibles de l'intérieur de la même classe et des autres classes ; les membres privés ne sont accessibles que depuis la même classe.
- **masquage de l'information** Le masquage de l'information est un principe selon lequel les détails internes de l'implémentation d'une classe devraient être cachés aux autres classes. Il permet une meilleure modularisation d'une application.
- **variable de classe, variable statique** Les classes peuvent avoir des champs. Ils sont connus sous le nom de *variables de classe* ou *variables statiques*. Un seul exemplaire d'une variable de classe peut exister à un instant *t*, indépendamment du nombre d'instances créées.