Fiche récap : Apprenez à programmer en Java

Développement

1. Manipuler des variables, des structures conditionnelles et itératives.
2. Créer de nouvelles classes avec les attributs et méthodes nécessaires
3. Tirer profit des capacités de la POO comme l'héritage et les interfaces.
4. Manipuler les collections pour organiser les données.
5. Gérer les exceptions pour stabiliser l'application.
6. Optimisez l'application grâce aux traces, à la programmation fonctionnelle et au multi threading.

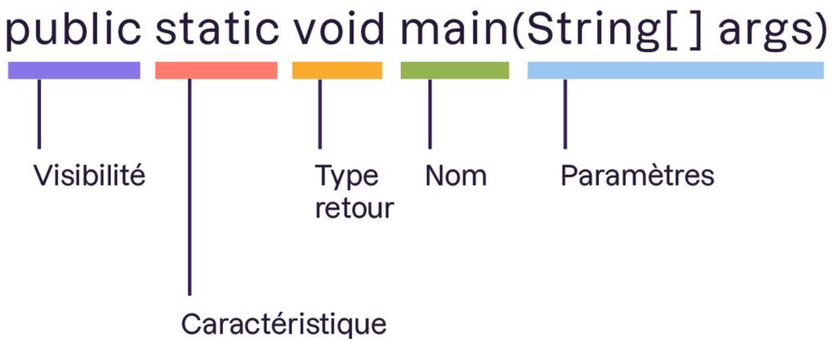
Code 💻

java

```java
public static void main(String[] args) {
    System.out.println("Hello World");
}
```

java

```java
List<String> list = new ArrayList<String>();
list.add("example");
list.forEach(System.out::println);
```



Définitions 🔎

Classe
Une classe est le plan d'un objet. Elle permet de définir les informations et les traitements de l'objet.

L'héritage
Une classe peut hériter d'une autre classe et ainsi accéder à ses attributs et méthodes. Cela permet également de redéfinir les méthodes de la classe héritée.

Interface
Une interface définit un contrat que des classes doivent suivre. Elle contient des méthodes abstraites sans implémentation qui devront être implémentées dans la classe.

Exception
Une exception est un événement qui perturbe le flux normal du programme. Elle est modélisée grâce à une classe et est gérée via les blocs `try`, `catch` .

Bonnes pratiques 👍

- ✅ Commenter le code pour faciliter la compréhension.
- ✅ Utiliser des noms de variables explicites et significatifs.
- ✅ Utiliser l'héritage pour réduire les duplications.
- ✅ Utiliser des interfaces pour définir les comportements.
- ✅ Utiliser des exceptions pour gérer les erreurs.
- ✅ Optimiser les performances grâce au multi-threading.
- ✅ Ajouter des traces grâce à un Logger.

Erreurs classiques 👎

- ❌ Omettre les commentaires dans le code.
- ❌ Utiliser des noms de variables non explicites.
- ❌ Dupliquer du code là où l'héritage permet de le factoriser.
- ❌ Écrire des classes sans abstraction, sans contrat.
- ❌ Ne pas gérer les exceptions correctement.
- ❌ Négliger l'optimisation des performances.
- ❌ Utiliser l'instruction System.out.println pour suivre l'exécution du code.