TP : Approfondissement TypeScript

Objectif : Aller plus loin que les types de base. Nous allons découvrir les outils puissants de TypeScript pour structurer nos données : les Enums, les Classes et les Génériques.

1. Exercice 1 : Les Enums (Énumérations)

Les Enums permettent de définir un ensemble de constantes nommées. C'est beaucoup plus propre que d'utiliser des chaînes de caractères magiques comme "EN_COURS" ou "TERMINE".

1. Créez une **Enum** nommée StatutCommande avec les valeurs :
   - EnAttente
   - Expediee
   - Livree
2. Créez une interface Commande :
   - id: number
   - statut: StatutCommande
3. Créez une fonction afficherEtat qui prend une commande.
   - Si le statut est Livree, affichez "Colis reçu !".
   - Sinon, affichez "En cours d'acheminement".

2. Exercice 2 : Classes et Encapsulation

TypeScript permet d'utiliser la Programmation Orientée Objet (POO) classique pour organiser le code.

1. Définissez une interface Livre avec :
   - titre (string)
   - auteur (string)
2. Créez une classe Bibliotheque.
   - **Propriété privée :** Ajoutez une propriété private catalogue: Livre[] initialisée comme un tableau vide. Le mot-clé private empêche de modifier le tableau directement depuis l'extérieur.
   - **Méthode publique :** Ajoutez une méthode ajouterLivre(nouveauLivre: Livre) qui pousse le livre dans le tableau.
   - **Méthode publique :** Ajoutez une méthode obtenirNombreLivres() qui retourne la taille du catalogue.
3. **Test :** Instanciez la classe, ajoutez deux livres, et essayez d'accéder directement à maBibliotheque.catalogue (VS Code doit vous bloquer).

3. Exercice 3 : Introduction aux Génériques ()

Imaginez que vous vouliez créer une "Boîte" qui peut contenir soit un texte, soit un nombre, soit un objet, mais qui garde ce type en mémoire. C'est le rôle des Génériques.

1. Créez une classe simple Boite<T>.
    - Elle a une propriété contenu: T.
    - Un constructeur qui prend une valeur de type T.
    - Une méthode regarder() qui retourne T.
2. **Utilisation :**
    - Créez une boiteAString : new Boite<string>("Bonjour").
    - Créez une boiteANumber : new Boite<number>(42).
    - Vérifiez que TypeScript comprend les types de retour de la méthode regarder().

4. Exercice 4 : Les "Utility Types" (Bonus)

TypeScript fournit des outils pour transformer les types existants.

1. Reprenez votre interface Livre (titre, auteur).
2. Créez une fonction mettreAJourLivre qui prend :
    - Un livre original (type Livre).
    - Des modifications. **Astuce :** Utilisez le type Partial<Livre>. Cela crée une version du type Livre où toutes les propriétés sont optionnelles.
3. La fonction doit retourner un nouveau livre fusionné (utilisez l'opérateur spread ...).

Pour tester

Lancez votre code avec la commande configurée ce matin :

```
npm start
# ou
npx ts-node src/index.ts
```