TP N02 : TYPESCRIPT

Exercice 1 : Interfaces et Types Avancés

Objectif

Comprendre et utiliser les interfaces, les types littéraux et les types d'union en TypeScript.

Instructions

1. **Définir des interfaces pour des formes géométriques :**
    a. Définis une interface Shape qui possède une méthode area retournant un nombre.
    b. Définis une interface Circle qui étend Shape et ajoute une propriété radius.
    c. Définis une interface Rectangle qui étend Shape et ajoute les propriétés **width** et **height**.
2. **Créer des classes implémentant ces interfaces :**
    a. Crée une classe CircleImpl qui implémente Circle et fournit une méthode pour calculer l'aire du cercle.
    b. Crée une classe RectangleImpl qui implémente Rectangle et fournit une méthode pour calculer l'aire du rectangle.
3. **Créer une fonction qui accepte n'importe quelle forme et calcule son aire :**
    a. Crée une fonction calculateArea qui prend en paramètre un objet de type Shape et retourne l'aire de la forme.
4. **Exemple d'utilisation :**
5. Instancie des objets CircleImpl et RectangleImpl.
6. Utilise la fonction calculateArea pour afficher l'aire des objets créés.

Exercice 2 : Types Génériques

Objectif

Comprendre et utiliser les types génériques en TypeScript.

Instructions

1. **Créer une classe générique Stack :**
    a. Crée une classe Stack qui accepte un type générique T.
    b. Implémente des méthodes push, pop, peek, et isEmpty pour gérer les éléments du Stack.
2. **Utiliser la classe Stack avec différents types :**
    a. Crée une instance de Stack pour les nombres et effectue quelques opérations de pile.
    b. Crée une instance de Stack pour les chaînes de caractères et effectue quelques opérations de pile.

Exercice 3 : Programmation Orientée Objet Avancée

Objectif

Mettre en pratique des concepts de la programmation orientée objet comme l'héritage, les classes abstraites, et les méthodes statiques.

Instructions

1. **Définir une classe abstraite Animal :**
    a. Crée une classe abstraite Animal avec un constructeur acceptant un nom et une méthode abstraite makeSound.
    b. Implémente une méthode move qui affiche un message indiquant que l'animal se déplace.
2. **Créer des classes dérivées Dog et Cat :**
    a. Crée une classe Dog qui étend Animal et implémente la méthode makeSound pour aboyer.

b. Crée une classe Cat qui étend Animal et implémente la méthode makeSound pour miauler.

3. **Créer une méthode statique pour compter le nombre d'instances :**
    a. Crée une classe AnimalCounter avec une propriété statique count et des méthodes statiques increment et getCount pour gérer le nombre d'instances d'animaux créées.
4. **Exemple d'utilisation :**
    a. Instancie des objets Dog et Cat et appelle leurs méthodes makeSound et move.
    b. Utilise AnimalCounter pour afficher le nombre d'animaux créés.

Exercice 4 : Manipulation Avancée des Types

Objectif

Utiliser des types conditionnels, les types utilitaires et les opérateurs de type avancés.

Instructions

1. **Définir un type conditionnel pour vérifier si un type est un tableau :**
    a. Crée un type conditionnel IsArray qui vérifie si un type donné est un tableau et retourne true ou false.
2. **Utiliser des types utilitaires pour rendre tous les champs optionnels et immuables :**
    a. Crée une interface Person avec des propriétés name, age, et address.

b. Utilise le type utilitaire Partial pour rendre tous les champs de Person optionnels.
c. Utilise le type utilitaire Readonly pour rendre tous les champs de Person immuables.

3. **Définir un type qui extrait les types des propriétés d'une classe :**
   a. Crée une classe Car avec des propriétés make, model, et year.
   b. Utilise l'opérateur de type keyof pour créer un type CarProperties qui représente les noms des propriétés de Car.