Suite du projet concessionnaire automobile - Fichier JSON et CLI

Pour ces exercices, nous allons réutiliser le code de l'exercice précédent, pour l'améliorer.

03-fonctions-packages.4.0

Exercice 1 : Sauvegarde et Chargement de l'Inventaire en JSON

Objectif : Implémenter des fonctionnalités pour sauvegarder et charger l'inventaire de voitures au format JSON.

Instructions :

- Ajoutez des méthodes `SaveToFile(filename string) error` et `LoadFromFile(filename string) error` à la structure `Dealership`.
- Utilisez le package `encoding/json` pour convertir l'inventaire en JSON et vice-versa.
- Testez la sauvegarde et le chargement en utilisant des fichiers JSON.

Résultats attendus :

- Un programme capable de sauvegarder l'inventaire dans un fichier JSON et de le charger depuis ce fichier.

Exercice 2 Simplifié : Création d'une Interface de Ligne de Commande pour Gérer un Inventaire de Voitures

But de l'exercice : Construire une interface de ligne de commande (CLI) en Go pour gérer un inventaire de voitures.

Exemple de code Go :

```go
package main

import (
  "flag"
  "fmt"
  "os"
)

func main() {
  // Configuration des commandes et arguments
  addCmd := flag.NewFlagSet("add", flag.ExitOnError)
  brand := addCmd.String("brand", "", "Marque de la voiture")
  year := addCmd.Int("year", 0, "Année de la voiture")
  color := addCmd.String("color", "", "Couleur de la voiture")

  // Traitement des commandes
  switch os.Args[1] {
  case "add":
    addCmd.Parse(os.Args[2:])
    fmt.Println("Ajout d'une voiture:")
    fmt.Printf("Marque: %s, Année: %d, Couleur: %s\n", *brand, *year, *color)
    // Ajout de la voiture à l'inventaire
  default:
    fmt.Println("Commande non reconnue")
```

```
  }
}
```

Tâches à réaliser :

1. Votre programme doit démarrer en lisant un fichier JSON. Utilisez l'argument `-file` pour spécifier ce fichier.
2. Le programme doit sauvegarder l'inventaire dans ce fichier JSON avant de se fermer.
3. Ajoutez la commande `init` pour créer un fichier JSON vide.
4. Ajoutez la commande `add` pour ajouter des voitures. Les arguments sont :

- `-brand` : Marque
- `-year` : Année
- `-color` : Couleur
- `-engine` : Type de moteur

5. Ajoutez la commande `remove` pour supprimer une voiture. Argument attendu :

- `-index` : Index de la voiture à supprimer

6. Ajoutez la commande `search` pour chercher des voitures. Arguments attendus :

- `-brand` : Marque
- `-year` : Année
- `-color` : Couleur
- `-engine` : Type de moteur

7. Ajoutez la commande `display` pour afficher l'inventaire.

Résultats espérés :

Un programme CLI qui permet d'effectuer les opérations ci-dessus en utilisant des commandes en ligne.

```
$ go run main.go -file inventory.json init
$ go run main.go -file inventory.json add -brand "Audi" -year 2018 -color "Noir" -engine 4
$ go run main.go -file inventory.json add -brand "BMW" -year 2019 -color "Blanc" -engine 6
$ go run main.go -file inventory.json add -brand "Mercedes" -year 2017 -color "Rouge" -engine 4
$ go run main.go -file inventory.json display
$ go run main.go -file inventory.json remove -index 0
$ go run main.go -file inventory.json display
$ go run main.go -file inventory.json search -brand "BMW" -year 2019
```