TP Garage - Partie 2

Version : 1.0

A partir du premier TP Garage, nous allons ajouter des fonctionnalités supplémentaires à notre gestion de garage permettant d'ajouter dynamiquement des véhicules via un menu, et de sauvegarder ces informations dans un fichier.

Les objectifs étant :

- Utiliser les exceptions
- Créer nos propres exceptions
- Traiter des données saisies
- Effectuer des opérations sur des listes
- Utiliser le mécanisme de sérialisation pour enregistrer des informations dans un fichier

Première étape :

Afin de mieux gérer le garage et de pouvoir enregistrer les informations de celui-ci dans un fichier, nous allons réaliser un menu permettant d'effectuer les commandes suivantes :

- Afficher les véhicules
- Ajouter un véhicule
- Supprimer un véhicule
- Sélectionner un véhicule
- Afficher les options d'un véhicule
- Ajouter des options à un véhicule
- Supprimer des options à un véhicule
- Afficher les options
- Afficher les marques
- Afficher les types de moteurs
- Sauvegarder le garage
- Quitter l'application

Le menu pourra ressembler à celui-ci :

```
********** Gestion de Garage **********
1.  Afficher les véhicules
2.  Ajouter un véhicule
3.  Sélectionner un véhicule
4.  Supprimer le véhicule
5.  Afficher les options du véhicule
6.  Ajouter des options au véhicule
7.  Supprimer des options au véhicule
8.  Afficher les options
9.  Afficher les marques
10. Afficher les types de moteurs
11. Sauvegarder le garage
0.  Quitter l'application

Choix :
```

Une classe Menu permettra de mettre en place l'affichage du menu et toutes les commandes.

Afin de gérer le garage cette classe aura comme constructeur :

```
public Menu(Garage garage)
{
	Garage = garage;
}
```

Il s'agira du garage créé dans le programme principal.

Afin de stocker également des moteurs et des options, il faudra rajouter dans votre garage une liste de moteurs, une liste d'options et les méthodes AjouterMoteur et AjouterOption.

Une méthode Start() permettra de lancer le menu à partir du programme. Cette méthode affichera le menu, et bouclera tanque l'utilisateur n'aura pas sélectionné le choix 0 pour sortir de l'application.

Pour traiter les exceptions de l'application on mettra un bloc try … catch global autour du menu qui permettra de de catcher les Exceptions levées par les commandes du menu et d'afficher un message d'erreur à l'utilisateur.

Dans cette méthode Start() un switch case permettra de tester les valeurs saisie par l'utilisateur. Cette valeur étant récupérée sous forme de string il faudra convertir cette valeur en int afin de la tester dans le switch case.

Créer une méthode GetChoix() qui permettra :

- Récupérer le choix saisi par l'utilisateur
- Effectuer la conversion en int
- Retourner le résultat converti

Le conversion peut être effectuer par la méthode `Convert.ToInt32()`.

Cette méthode devra implémenter une première gestion d'exception pour gérer le cas où l'utilisateur ne saisit pas un nombre. L'exception levée lors de la conversion est la FormatException. Il faudra dans cette méthode catcher l'exception puis lever à nouveau la FormatException avec le message d'erreur : "Le choix saisie n'est pas un nombre".

Le message sera affiché par le bloc try … catch de la méthode Start.

Il faudra aussi afficher un message si l'utilisateur saisie un choix qui n'existe pas dans le menu.

Pour cela nous allons créer une Exception spécifique à notre application.
Créer cette Exception dans le fichier Menu.cs et nommer la MenuException. Utiliser le constructeur de l'exception pour spécifier un message qui pourra être "Le choix n'est pas compris entre 0 et 11".

Créer une méthode GetChoixMenu() qui utilisera la méthode GetChoix() pour récupérer le nombre saisie et qui vérifiera si le nombre est compris entre 0 et 11. Si ce n'est pas le cas il faudra lever l'exception MenuException.

Le message sera affiché par le bloc try … catch de la méthode Start.

Pour chaque opération du menu créer une méthode, qui sera chargée d'effectuer toutes les opérations nécessaires de la commande.

Par exemple pour la commande « 1. Afficher les véhicules », on créera une méthode :

```
Public void AfficherVehicules()
{

}
```

Qui sera appelé dans le switch case :

```
case 1:
       AfficherVehicules();
       break;
```

Créer toutes les méthodes correspondant à chaque fonction prévue dans le menu.

Coder toutes les fonctionnalités en essayant de mettre dans la classe Garage les fonctions de base et dans la classe Menu l'interface avec l'utilisateur.

Ajouter également de nouvelles exceptions pour traiter par exemple les cas où l'utilisateur sélectionne un véhicule qui n'existe pas, une marque qui n'existe pas, un moteur qui n'existe pas, …

Deuxième étape :

Modifier les différentes classes Vehicule, Voiture, Camion, … afin d'ajouter le mécanisme de sérialisation.

Pour rappel, il suffit d'ajouter l'attribut :

```
[Serializable]
```

A chaque classe.

Ajouter en vous inspirant du support de cours dans votre classe Garage les deux méthodes :

Enregistrer : Permet d'enregistrer un objet.
Charger : Permet de charger un objet.

Ajouter dans votre gestion de menu et dans votre programme la fonction de sauvegarde d'un garage. Le chargement du garage se fera au lancement de l'application. Si le garage est vide prévoir l'ajout et l'enregistrement automatique de véhicule par défaut.