COLLÈGE MONTMORENCY

Travail pratique #2

Pondération 25%

Date de remise : voir Moodle

Ce travail pratique sera validé par un test en classe (quiz) qui aura lieu après la remise du travail. Vous recevrez une seule, note qui comptera pour 25% de la note finale.

Une pénalité de 10% sera appliquée si les directives de remise ci-dessous ne sont pas respectées.

- Créez un package nommé **`tp02`** dans le package qui porte votre nom**.** Ce package contiendra toutes les classes de ce travail pratique.
- Vous devez respecter toutes les normes de programmation contenues dans le document intitulé **Normes de codage** disponible sur Moodle.
- Déposez le package `tp02` compressé en format .zip sur Moodle.
- Chaque classe doit comporter un en-tête conforme au modèle ci-dessous.

```
/**
 * Description de la classe.
 *
 * @author nom, prénom
 *
 */
```

Travail à faire

Vous devez implémenter un programme Java pour gérer un parc de véhicules usagés dans un concessionnaire automobile. Le parc a une capacité maximale de 400 véhicules.

Votre programme devra permettre :

- d'ajouter un véhicule,
- de supprimer un véhicule,
- de rechercher un véhicule par son numéro d'inventaire,
- d'afficher la liste des véhicules en inventaire,
- d'afficher la moyenne des prix des véhicules.

Les informations à enregistrer pour chaque véhicule sont:

- le numéro d'inventaire qui est unique,
- la marque,
- le modèle,
- l'année,
- le prix.

Ces informations devront être stockées dans une liste implémentée à l'aide d'un tableau d'objets.

Vous devez implémenter les classes suivantes: `Vehicule`, `ListeVehicules` et `GestionParc`

1. La classe `Vehicule`

La classe `Vehicule` représente un véhicule du parc. Elle contient les attributs privés suivants:

- `numeroInventaire` (`String`) : numéro d'inventaire.
- `marque` (`String`) : marque du véhicule.
- `modele` (`String`): modèle du véhicule.
- `annee` (`int`) : année de fabrication du véhicule.
- `prix` (`double`): prix du véhicule.

Implémentez les méthodes suivantes.

- **Les accesseurs (getters) et mutateurs (setters) pour chaque attribut, avec les validations suivantes:**
  - `numeroInventaire` doit comporter exactement 7 caractères.
  - `marque` doit être une chaîne de 2 à 15 caractères (ex. : Toyota, Honda, Mazda).
  - `modele` doit être une chaîne de 2 à 15 caractères (ex. : Corolla, Civic, CX-3).
  - `annee` doit être comprise entre 2000 et l'année en cours (ex. : 2011, 2019).
  - `prix` doit être compris entre 0 et 250 000 (ex. : 25 900, 19 850)..
  -

  Utilisez des constantes privées et statiques pour ces valeurs (exemple : `private static final int MIN_CAR = 2;`).
- **Trois constructeurs :**

1. Un constructeur à 4 paramètres pour initialiser `numeroInventaire, marque, modele et annee`. Appelez les mutateurs quand c'est possible.
2. Un constructeur à 5 paramètres qui initialise tous les attributs et appelle le constructeur précédent (à 4 paramètres).
3. Un constructeur par défaut (sans paramètres) qui initialise :
   - `numeroInventaire, marque` et `modele` à "Inconnu",
   - `annee` à 2000,
   - `prix` à 0.

   Ce constructeur **doit appeler un autre constructeur de la même classe**.

- **Une méthode** `String toString()` qui retourne une représentation du véhicule sous la forme :
  `2021001 Hundai Elantra 2019 12500,00$`
  Note: Utilisez `String.format()` pour formater la chaîne.

- **Des méthodes de lecture avec un** `scanner` :
  - `public void lireNumeroInventaire(Scanner clavier)`
    - Lit un numéro d'inventaire valide et l'affecte à l'attribut numeroInventaire.
  - `public void lireMarque(Scanner clavier)`
    - Lit une marque valide et l'affecte à l'attribut `marque`.
  - `public void lireModele(Scanner clavier)`
    - Lit un modèle valide et l'affecte à l'attribut `modele`.
  - `public void lireAnnee(Scanner clavier)`
    - Lit une année valide et l'affecte à l'attribut `annee`.
  - `public void lirePrix(Scanner clavier)`
    - Lit un prix valide et l'affecte à l'attribut `prix`.
- **Ajoutez une méthode** `main()` pour tester l'ensemble avec des données significatives. Testez tous les constructeurs et la méthode `toString()`.

2. La classe `ListeVehicules`

La classe `ListeVehicules` représente une liste de véhicules implémentée à l'aide d'un tableau d'objets de type `Vehicule`. Elle contient des méthodes permettant d'effectuer des opérations sur cette liste telles que l'ajout, la suppression et la recherche d'un véhicule.

Cette classe doit contenir les attributs suivants:

- `tabVehicules`: un tableau d'objets de type `Vehicule`.
- `nbVehicules`: un entier représentant le nombre actuel de véhicules dans la liste (et non la taille du tableau).

Méthodes à implémenter.

- **Constructeur :**
  - `public ListeVehicules(int maxVehicules)`
    - initialise le tableau `tabVehicules` avec `maxVehicules` cases;
    - initialiser `nbVehicules` à 0.
- **Méthodes d'information :**

- `public int taille()`
  - Retourne le nombre de véhicules actuellement dans la liste.
- `public boolean estVide()`
  - Retourne `true` si la liste ne contient aucun véhicule, `false` sinon.
- `public boolean estPlein()`
  - Retourne `true` si la liste est pleine (c'est-à-dire si `nbVehicules` atteint la taille du tableau), `false` sinon.

- **Méthodes de manipulation**
  - `public Vehicule obtenirVehicule(int indice)`
    - Retourne le véhicule situé à l'indice donné en paramètre.
    - Si l'indice est invalide, la méthode retournera `null` .
  - `public boolean ajouter(Vehicule vehiculeAjoute)`
    - Ajoute le véhicule reçu en paramètre dans la liste, si celle-ci n'est pas pleine.
    - Elle retourne `true` si l'ajout est réussi, `false` sinon.
  - `public void afficher()`
    - Affiche tous les véhicules contenus dans la liste en utilisant la méthode `toString()` de la classe `Vehicule`.
    - Le format d'affichage doit être :

```
2024001 Hundai          Elantra         2019   12500,00$
2024002 Toyota          Corolla         2018   10500,00$
…
```

  - `public int indexDe(String numeroVehicule)`
    - Retourne l'indice du véhicule dont le numéro d'inventaire est donné en paramètre.
    - Retourne -1 si le véhicule n'existe pas.
  - `public boolean supprimer(String numeroVehicule)`
    - Supprime le véhicule dont le numéro d'inventaire correspond au paramètre.
    - Retourne `true` si la suppression a eu lieu, `false` sinon.
  - Ajoutez une méthode **`main()`** pour tester toutes les fonctionnalités ci-dessus. Effectuez des tests significatifs.

3. La classe `GestionParc`

La classe `GestionParc` permet de gérer un parc de véhicules. Elle affiche un menu interactif dans la console (terminal) et traite les choix de l'usager dans la méthode principale (`main()`).

Menu affiché :

```
****** Gestion du parc automobile ******
1- Ajouter un nouveau véhicule
2- Supprimer un véhicule
3- Rechercher un véhicule par son numéro d'inventaire
4- Afficher la liste des véhicules
5- Afficher la moyenne des prix des véhicules
6- Quitter
Faites votre choix:
```

Attributs de la classe :

`clavier` : une variable `Scanner` pour la saisie utilisateur.

```
public static Scanner clavier = new Scanner(System.in);
```

Méthodes statiques à implémenter :

- `public static void main(String[] args)`
  - Déclare et initialise une liste de véhicules (objet `ListeVehicules`) avec une capacité de 400 véhicules**.**
  - Appelle `initialiserListeVehicules()` pour ajouter **6 véhicules** dans la liste (voir la méthode suivante)
  - Affiche le menu et attend l'entrée de l'utilisateur.
  - Si l'utilisateur saisit un choix invalide, affiche « `Choix invalide` » et réaffiche le menu après.
  - Exécute l'action correspondante au choix de l'utilisateur et réaffiche le menu après chaque action.

- `public static void initialiserListeVehicules(ListeVehicules listeVehicules)`
  - Ajoute les 3 véhicules suivants **ainsi que 3 autres véhicules de votre choix** dans la liste des véhicules . Aucune saisie clavier n'est nécessaire.

Numéro d'inventaire	Marque	Modèle	Année	Prix
2024001	Hyundai	Elantra	2019	12 500,00$
2024002	Toyota	Corolla	2018	10 500,00$
2024003	Honda	Civic	2024	-

Note : le prix du dernier véhicule n'est pas fourni.

  - Cette méthode ne doit être exécutée **qu'une seule fois au démarrage** pour initialiser la liste.

- `public static void afficherVehicules(ListeVehicules listeVehicules)`
  - Affiche tous les véhicules de la liste sous le format suivant :

```
Numéro  Marque          Modèle           Année Prix
-------------------------------------------------------
2024001 Hyundai         Elantra         2019  12500,00$
2024002 Toyota          Corolla         2018  10500,00$
2024003 Honda           Civic            2024      0.00$
…
```

  - Si la liste est vide, affiche « `Il n'y a aucun véhicule à afficher.` »

- `public static void ajouterVehicule(ListeVehicules listeVehicules)`
  - Demande à l'utilisateur de saisir les informations d'un véhicule.
  - Vérifie si :

    1. La liste est pleine : elle affiche « `Impossible d'ajouter des véhicules, le parc est plein.` »

2. Le numéro d'inventaire existe déjà : elle affiche « `Ajout impossible. Il existe déjà un véhicule avec ce numéro.` »
3. Sinon, ajoute le véhicule et affiche une confirmation sous forme « `Le véhicule de numéro d'inventaire 2024007 a été ajouté avec succès.` »

📌 **Exemples de sortie :**

✅ **Ajout réussi :**

```
Faites votre choix: 1
Entrez le numéro d'inventaire du véhicule: 2024007
Entrez la marque du véhicule: Toyota
Entrez le modèle du véhicule: RAV4
Entrez l'année du véhicule: 2024
Entrez le prix du véhicule: 48250
Le véhicule de numéro d'inventaire 2024007 a été ajouté avec succès.
```

❌ **Ajout refusé (véhicule existant) :**

```
Faites votre choix: 1
Entrez le numéro d'inventaire du véhicule: 2024001
Ajout impossible. Il existe déjà un véhicule avec ce numéro.
```

❌ **Ajout refusé (liste pleine) :**

```
Faites votre choix: 1
Impossible d'ajouter des véhicules, le parc est plein.
```

Note: pour faire ce test, procédez comme suit dans l'ordre.

1. Modifiez temporairement la taille de la liste à 6.
2. Exécutez le programme. À ce moment la liste est pleine .
3. Choisissez l'option 1 du menu pour tenter d'insérer un nouveau véhicule.

- `public static void supprimerVehicule(ListeVehicules listeVehicules)`
  - Si la liste est vide : elle affiche : « `Il n'y a aucun véhicule à supprimer.` »
  - Sinon, elle :
    - Demande le numéro du véhicule à supprimer,
    - Vérifie si :

1. La liste est vide : elle affiche : « `Il n'y a aucun véhicule à supprimer.` »
2. Le numéro est introuvable : elle affiche : « `Il n'y a aucun véhicule avec ce numéro d'inventaire` »
3. Sinon, elle demande une confirmation sous forme : « `Voulez-vous vraiment supprimer le véhicule de numéro d'inventaire 2021001 (O/N)?` »
4. Si l'utilisateur confirme (`O` ou `o`), elle supprime le véhicule et affiche un message sous forme: «`Le véhicule de numéro d'inventaire 2021001 a été supprimé avec succès.`»

5. Sinon, elle annule la suppression et affiche un message sous forme : « `La suppression du véhicule de numéro d'inventaire 2021001 a été annulée.` »

📌 **Exemples de sortie :**

✅ **Suppression d'un véhicule existant:**

```
Faites votre choix: 2
Entrez le numéro du véhicule à supprimer: 2024001
Voulez-vous vraiment supprimer le véhicule de numéro d'inventaire 2024001
O/N? O
Le véhicule de numéro d'inventaire 2024001 a été supprimé avec succès.
```

✅ **Tentative de suppression d'un véhicule existant avec annulation de la suppression**.

```
Faites votre choix:2
Entrez le numéro du véhicule à supprimer: 2024001
Voulez-vous vraiment supprimer le véhicule de numéro d'inventaire 2024001
O/N? N
La suppression du véhicule de numéro d'inventaire 2024001 a été annulée.
```

❌ **Tentative de suppression d'un véhicule inexistant.**

```
Faites votre choix: 2
Entrez le numéro du véhicule à supprimer: 2024999
Il n'y a aucun véhicule avec ce numéro d'inventaire.
```

❌ **Tentative de suppression d'un véhicule dans une liste est vide.**

```
Faites votre choix: 2
Il n'y a aucun véhicule à supprimer.
```

- `public static void rechercherVehicule(ListeVehicules listeVehicules)`
  - Si la liste est vide : elle affiche « `Il n'y a aucun véhicule à rechercher.` »
  - Sinon, elle :
    - Demande le numéro du véhicule à rechercher.
    - Vérifie si le véhicule se trouve dans la liste :
      - Si oui, elle affiche ses informations
      - Sinon, elle affiche un message sous forme « `Il n'y a aucun véhicule avec le numéro d'inventaire 2024999` ».
      -

📌 **Exemples de sortie :**

✅ **Rechercher un véhicule qui existe**

```
Faites votre choix: 3
Entrez le numéro du véhicule à rechercher: 2024001
Véhicule trouvé:
Numéro  Marque          Modèle           Année  Prix
--------------------------------------------------------
2024001 Hyundai         Elantra          2019   12500,00$
```

❌ **Rechercher un véhicule qui n'existe pas.**

```
Faites votre choix: 3
Entrez le numéro du véhicule à rechercher: 2024999
Il n'y a aucun véhicule avec le  numéro d'inventaire 2024999.
```

❌ **Rechercher un véhicule dans une liste vide**

```
Faites votre choix: 3
Il n'y a aucun véhicule à rechercher.
```

Note: dans ce dernier cas, on ne demande pas le numéro d'inventaire à l'utilisateur.

- `public static void moyenneVehicules(ListeVehicules listeVehicules)`
  - Calcule et affiche le prix moyen des véhicules. L'affichage devrait être sous la forme: « `Le prix moyen des véhicules est: 11500.00$`»
  - Si la liste est vide, le message d'erreur suivant sera affiché: « `Il n'y a aucun véhicule.`»