Université de Montréal

COURS IFT1142

TRAVAIL PRATIQUE #1

CONSIGNES

1. **Il faut remettre la version électronique. On n'accepte pas des fichiers envoyés par courriel.**
2. **La date de remise est sur Studium.**
3. **Tout travail remis en retard aura une pénalité de 10% par jour de retard.**
4. **Les travaux remis avec 5 jours et plus de retard seront refusés.**

GESTION D'UN CATALOGUE DE LIVRES

1. PRÉSENTATION DU TRAVAIL PRATIQUE

Titre : Gestion de livres (CRUD et requêtes de sélection)
Langage : JavaScript ES6+ (sans la POO)
Données : Fichier **livres.js** (module)
Technologies :

- **Application Node en mode console**
- **JavaScript ES6+** (sans la POO). Bonne utilisation des méthodes de la classe Array.
- **Fichier livres.js (**module**)** pour stocker les livres. Ce fichier se trouve dans Studium.

Le travail consiste à concevoir une application qui permet de **créer**, **lire**, **modifier** et **supprimer** des livres, ainsi que d'effectuer **quelques requêtes de sélection** (exemples plus bas).

2. STRUCTURE DES DONNÉES DANS LIVRES.JS

Chaque livre est représenté par un objet JSON de la forme suivante. Tous les objets sont dans un tableau **tabLivres**.

```
let tabLivres =[
 {
  "id": 400,
  "titre": "Une aventure d'Astérix le gaulois. Le devin",
  "idAuteur": 11,
  "annee": 1972,
  "pages": 48,
  "categorie": "bandes dessinées"
 },
```

Explication des attributs d'un objet livre :

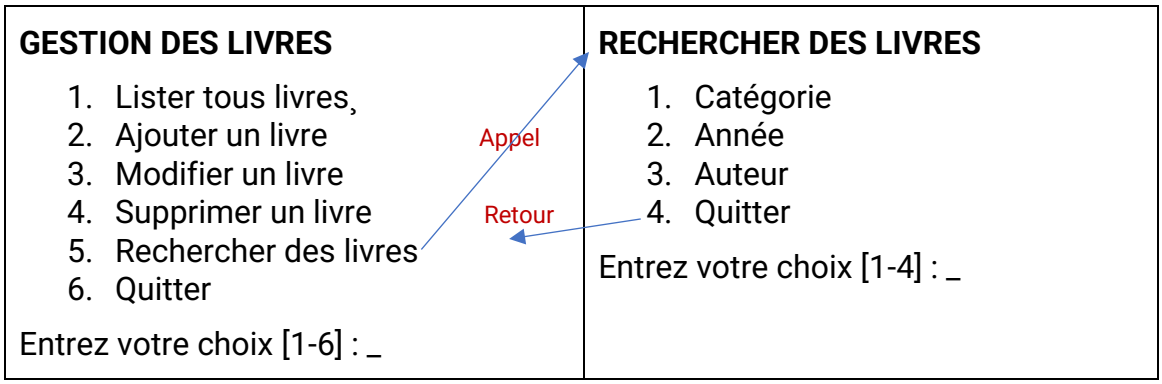
- **id** (entier) : identifiant unique du livre.
- **titre** (texte) : titre du livre.
- **idAuteur** (entier) : identifiant de l'auteur (ou tout autre code d'auteur).
- **annee** (entier) : année de publication.
- **pages** (entier) : nombre de pages du livre.
- **categ** (texte) : la catégorie du livre (ex. "bandes dessinées", "roman", "manga", etc.).

Tous les ajouts, modifications ou suppressions doivent être enregistrés dans le tableau **tabLivres**.

3. CONTENU ATTENDU (FONCTIONNALITÉS)

1. Menu d'accueil

- Doit y avoir toutes les options de traitement.
- Doit être dans le fichier **main.js**. Ce fichier aura aussi une autre fonction **main()**. L'exécution commencera par l'appel à cette fonction **main()**, qui à son tour appellera la fonction **menu()**.



2. CRUD (sur tabLivres) :

- **C** (Create) : Ajouter un livre en demandant les informations via la console.
- **R** (Read) : Lister tous les livres, afficher leurs informations dans la console.
- **U** (Update) : Modifier un livre via son id (en récupérant les infos du livre).
- **D** (Delete) : Supprimer un livre via son id (après confirmation).

3. Requêtes de sélection :

Vous devez implémenter les trois requêtes (via les options du menu) pour filtrer ou sélectionner des livres, par exemple :

- Sélection par **catégorie** (afficher uniquement les livres d'une catégorie donnée).
- Sélection par **année** (afficher uniquement les livres publiés après une certaine année).
- Sélection par **auteur** (selon l'idAuteur, etc.).

L'affichage du résultat de chaque requête se fera selon le format suivant :

Dans la console selon le format suivant (les données doivent être bien alignées) :

```
                                LISTE DES LIVRES
ID      Titre                          IdAuteur   Année    Pages      Catégorie
-------------------------------------------------------------------------------------
xxx     xxxxxxxxxxxxxxxxxxxxxx     xx          xxxx       xxxx       xxxxxxxxxxx

……..

Nombre de livres : xxxxx
```

Ceci est le format pour afficher tous les livres. Si nous voulons afficher tous les livres de la catégorie **Roman**, alors le titre sera **LISTE DES LIVRES DE LA CATÉGORIE Roman** et la colonne **Catégorie** n'est plus nécessaire. Essayiez d'utiliser une seule fonction en envoyant en paramètres des informations pour savoir quel titre et colonnes à afficher. Nous voulons éviter la redondance du code.

4. **Ergonomie :**
   - Gestion d'un menu via la console (terminal).
   - L'interface doit être **intuitive** et **cohérente**.
5. **Barème d'évaluation (sur 100%)**

Critères	Points
Ergonomie	10%
– Utilisation efficace du menu	
– Navigation claire et cohérente	
– Présentation des résultats et demande de saisie de données	
Fonctionnalités CRUD & Requêtes (7x10%)	70%
– Ajout, Modification, Suppression	
– Liste correct des livres	
– Trois requêtes de sélection	
Structure du code	20%
– Code commenté	
– Utilisation du fichier **livres.js**	
– Respect de la structure de projet demandée.	
– Clarté et organisation (fonctions, modules, utilisation des méthodes de la classe Array de JavaScript, , etc.)	
Total	**100%**

5. PÉNALITÉS SPÉCIFIQUES

1. **-10%** par consigne non respectée (ex. pas de menu, format d'affichage non respecté, etc.).
2. Toute option du menu qui ne fonctionne pas **= 50% maximum** des points attribués à la question, l'autre 50% étant accordé pour la partie du code qui a été implémentée.

Exemple : Si la requête de sélection par catégorie est un des points notés à 10, mais l'option du menu ne fonctionne pas, vous obtiendrez au maximum 5 points, et seulement si le code interne pour filtrer les livres est présent et bien fait.

6. LIVRABLES ATTENDUS

- Le dossier du projet nommé TP1 selon la structure de projet donnée en classe.
- Le projet doit être remis dans Studium dans le lien REMISE TP1

7. CONSEILS / RECOMMANDATIONS

- **Commenter** votre code : chaque fonction doit être documentée, ainsi que les blocs les plus importants.
- Respecter le format du projet.
- Testez fréquemment vos opérations :
  - Ajouter un livre,
  - Vérifier qu'il est bien écrit dans livres.js,
  - Modifier ce même livre,
  - Supprimer ensuite, etc.
- Restez **cohérent** sur les noms de variables et gardez ce même nom partout dans votre projet (pensez au dictionnaire de données), la structure, la navigation, etc.

EN RÉSUMÉ :

Vous devrez produire une **application console avec Node** permettant de gérer une liste de livres (fichier **livres.js**), avec un **CRUD complet** (Create, Read, Update, Delete) et **trois requêtes de sélection** (filtrage). L'**interface** d'accueil sera réalisée via un menu qui apparait dans la console. Vous serez évalué sur l'**ergonomie** (10%), la **qualité et l'exhaustivité** des fonctionnalités (70%), et la **structure et clarté du code** (20%), tout en respectant les consignes pénalisantes mentionnées.

Bonne réalisation et bon courage ! 😊