

## Travail pratique #2

### Objectif

L'objectif de ce travail est de mettre en pratique l'utilisation de plusieurs formulaires dans une application, l'utilisation des listes génériques, la composition d'objet, les relations entre les objets, les paramètres et les valeurs de retour de type objet, les fichiers texte, les énumérations, les types en C# ainsi que la gestion des exceptions.

### Contexte

*La Clinique du Coureur* est un organisme qui organise des courses à pied partout au Canada. Celle-ci fait appel à vous afin de mettre sur pied une application permettant de gérer l'inscription de participants à une course à pied, ainsi que la gestion des résultats de la course.

### Modalités

- Ce travail doit être fait **en équipe de deux**
- Cette partie du travail est sur 100 points et vaut pour **15% de la note finale**.
- Date de remise : **Voir date de remise sur LÉA**

### Fonctionnement de l'application

- L'application permettra à un utilisateur de créer, modifier et supprimer des courses
- Celle-ci permettra également d'ajouter, de modifier et de supprimer des participants à cette course.
- Il sera également possible de consulter les résultats de la course ainsi que certaines statistiques.

L'application devra également permettre la sauvegarde des données sur les courses et les coureurs dans des fichiers .CSV

## Interface de l'application

### Formulaire "FormCourse"

Ce formulaire est le formulaire principal de l'application. :

The screenshot shows a window titled 'FormCourse' with a search bar and a list of races. The list has four columns: NOM, VILLE, PROVINCE, and DATE. Below the list are three buttons: Ajouter, Modifier, and Supprimer.

NOM	VILLE	PROVINCE	DATE
COURSE DES 10 KM D'OTTAWA	Ontario	Ottawa	2024-09-26
MARATHON DE MONTRÉAL	Québec	Montréal	2024-09-26
COURSE DES CHUTES MONTMORENCY	Québec	Québec	2024-02-23
MARATHON DE VANCOUVER	Colombie-Britannique	Vancouver	2024-02-16
LA CLASSIQUE DU PARC LAFONTAINE	Québec	Montréal	2023-12-06
SEMI-MARATHON DE LA BAIE	Québec	Saguenay	2023-07-29
COURSE DU MONT ROYAL	Québec	Montréal	2022-08-16
ULTRA-TRAIL HARRICANA	Québec	La Malbaie	2022-08-16
TRAIL DU MONT ORFORD	Québec	Orford	2022-06-27
MARATHON DE TORONTO	Ontario	Toronto	2022-03-13

- **Liste des courses**

- Affiche la liste des courses en **ordre de date de la plus récente à la plus ancienne et en ordre alphabétique de nom.**
- 

- **Bouton Ajouter**

- ☐ Permet l'affichage du formulaire « **FormCourse** » pour créer une nouvelle course.
- ☐ La course ne doit pas être ajoutée s'il existe une course identique déjà existante dans la liste.
- ☐ La liste des courses doit être mise à jour suite à l'ajout d'une course et les données doivent être enregistrées.
- ☐ Un message de confirmation doit être affiché à l'utilisateur.

- **Bouton Modifier**

- ☐ Permet de modifier les informations de la course sélectionnée dans la liste des courses à partir du formulaire « **FormCourse** »
- ☐ La liste des courses doit être mise à jour à la suite de la modification et les données doivent être enregistrées.

- Un message de confirmation doit être affiché à l'utilisateur.

- **Bouton Supprimer**

- Permet la suppression de la course sélectionnée dans la liste des courses à partir du formulaire « **FormCourse** »
- Un message demandant à l'utilisateur de confirmer la suppression de la course doit être affiché.
- La liste des courses doit être mise à jour à la suite de la modification et les données doivent être enregistrées.

### Formulaire "FormCourse"

Il est responsable de la création, de la modification et de la suppression d'une course. Vous devez reproduire le plus fidèlement possible le formulaire nommé **FormCourse** suivant :

#### Onglet « Information sur la course » :

The screenshot shows a web application window titled 'Course'. Inside, there's a form titled 'Modifier une course'. The form has two tabs: 'Information sur la course' (active) and 'Coureurs'. The 'Information sur la course' tab contains the following fields:

- Nom: MARATHON DE MONTRÉAL
- Nbr. participants: 22
- Ville: Montréal
- Temps de course moyen: 03:36:18
- Province: Québec (dropdown menu)
- Date: 26 septembre 2024 (calendar icon)
- Type: Sentier (dropdown menu)
- Distance: 45

At the bottom of the form are two buttons: 'Modifier' and 'Annuler'.

- **Nom** : Champ obligatoire. Doit contenir un minimum de 3 caractères.
- **Ville** : Champ obligatoire. Doit contenir un minimum de 4 caractères.
- **Province** : Champ obligatoire. Contient la liste des provinces du Canada.
- **Date** : Champ obligatoire.

- ☐ **Type** : Champ obligatoire. Contient la liste des types de courses.
- ☐ **Distance** : Champ obligatoire. Doit être un nombre réel supérieur ou égal à 1.
- ☐ **Nb. participants** : Champ en lecture seul affichant le nombre de participants.
- ☐ **Temps course moyen** : Champ en lecture seul affichant le temps de course moyen. Ne doit pas afficher les millisecondes (hh:mm:ss).

### Onglet "Coureurs"

DOSSARD	NOM	CATÉGORIE	TEMPS	RANG
12	LAVOIE, SOPHIE	F40-49	02:39:50	1
20	LEFEBVRE, CAMILLE	F20-29	03:00:11	2
21	TREMBLAY, MARTIN	F40-49	03:05:16	3
22	GAUTHIER, SARAH	F30-39	03:11:33	4
14	BOUCHARD, OLIVIER	M20-29	03:30:32	5
13	BERGERON, THOMAS	M50-59	03:51:02	6
19	BERGERON, MARTIN	F30-39	04:00:41	7
9	PELLETIER, CHARLES	M20-29	04:07:38	8
8	MARTEL, LAURA	F30-39	04:08:41	9
2	GIRARD, SAMUEL	F50-59	04:27:43	10
6	BOUCHARD, ÉMILIE	M40-49		

- Cet onglet affiche les participants à la course. Ceux-ci doivent être affichés en **ordre croissant du temps de course**. **Les coureurs ayant abandonné la course doivent se trouver à la fin de la liste et ne pas avoir de rang ni de temps officiel**. Le texte contenu dans la liste doit être en majuscule.
- **Liste des résultats :**
  - Affiche les participants à la course. Ceux-ci doivent être affichés en **ordre croissant du temps de course**. **Les coureurs ayant abandonné la course doivent se trouver à la fin de la liste et ne pas avoir de rang ni de temps officiel**. Le texte contenu dans la liste doit être en majuscule.

- **Bouton Ajouter**

- ☐ Permet l'ajout d'un coureur à la course à partir du formulaire « **FormCoureur** »
- ☐ Le coureur ne doit pas être ajouté à la liste si le numéro de dossard est déjà utilisé par un autre coureur ou s'il existe déjà un coureur avec les mêmes informations.
- ☐ La liste des résultats doit être mise à jour à la suite de l'ajout d'un coureur.
- ☐ Un message de confirmation doit être affiché à l'utilisateur.

- **Bouton Modifier**

- ☐ Permet de modifier les informations du coureur sélectionné dans la liste des résultats à partir du formulaire « **FormCoureur** ».
- ☐ Un message de confirmation doit être affiché à l'utilisateur.
- ☐ La liste des résultats doit être mise à jour à la suite de la modification.

- **Bouton Supprimer**

- Permet le retrait d'un coureur de la course.
- Un message demandant à l'utilisateur de confirmer la suppression du coureur doit être affiché.
- La liste des résultats doit être mise à jour à la suite de la suppression.

- **Bouton Ajouter, Modifier, Supprimer** : Bouton permettant de confirmer l'ajout, la modification ou la suppression d'une course selon l'état du formulaire.

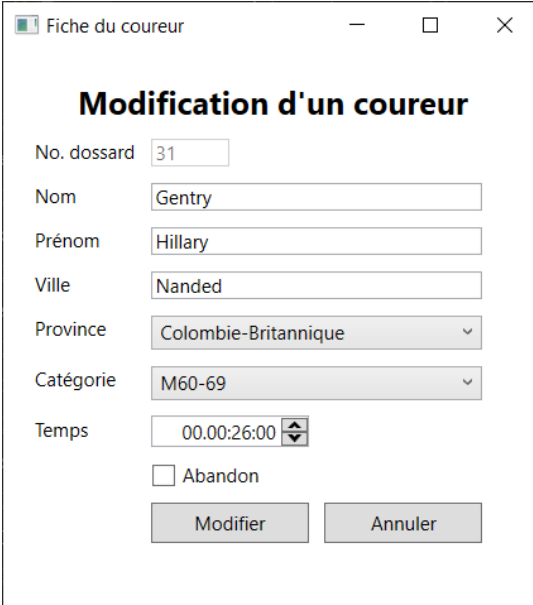
- Le texte du bouton doit représenter l'état du formulaire.
- Si le formulaire n'est pas valide alors un message doit être affiché à l'utilisateur lui indiquant toutes les erreurs.
- Lors d'une suppression, le formulaire est dans l'état de suppression, alors **tous les champs doivent être désactivés**. Un message demandant la confirmation de la suppression par l'utilisateur doit être affiché.

- **Bouton Annuler**

- Permet la fermeture du formulaire en annulant l'action.

### Formulaire "FormCoureur".

Ce formulaire permet l'ajout, la modification et la suppression d'un coureur. Vous devez reproduire le plus fidèlement possible le formulaire nommé **FormCoureur** suivant :



The screenshot shows a window titled "Fiche du coureur" with standard window controls. Inside, the title "Modification d'un coureur" is centered. The form contains the following fields and controls:

- No. dossard: Text input with value "31".
- Nom: Text input with value "Gentry".
- Prénom: Text input with value "Hillary".
- Ville: Text input with value "Nanded".
- Province: Dropdown menu with "Colombie-Britannique" selected.
- Catégorie: Dropdown menu with "M60-69" selected.
- Temps: Time input with value "00:00:26:00" and a spin button.
- Abandon: Unchecked checkbox.
- Buttons: "Modifier" and "Annuler".

Voici les détails du formulaire :

- **Titre du formulaire** : Texte obligatoire. Afficher le titre du formulaire selon l'état dans lequel se trouve le formulaire :
  - Ajouter un coureur
  - Modifier un coureur
  - Supprimer un coureur
- **Dossard** : Numéro du dossard du coureur. Champ obligatoire. Actif seulement dans le cas de l'ajout d'un nouveau coureur. Doit être un nombre entier supérieur à 1.
- **Nom** : Nom du coureur. Champ obligatoire. Doit contenir un minimum de 3 caractères.
- **Prénom** : Prénom du coureur. Champ obligatoire. Doit contenir un minimum de 3 caractères.
- **Ville** : Ville du coureur. Champ obligatoire. Doit contenir un minimum de 4 caractères.

- **Provinces** : Contient la liste des provinces du Canada. Champ obligatoire.
- **Categories** : Contient la liste des catégories. Champ obligatoire.
- **Temps** : Temp de course du coureur au format "hh:mm:ss".
- **Abandon** : Indique si un coureur a abandonné la course. Si la case à cocher est sélectionnée alors le temps de cours doit être remis à "00:00:00" et le champ est désactivé.
- **Bouton Ajouter, Modifier, Supprimer** : Bouton permettant de confirmer l'ajout, la modification ou la suppression d'un coureur selon l'état du formulaire.
  - Le texte du bouton doit représenter l'état du formulaire.
  - Si le formulaire n'est pas valide alors un message doit être affiché à l'utilisateur lui indiquant toutes les erreurs.
  - Lors d'une suppression, le formulaire est dans l'état de suppression, alors **tous les champs doivent être désactivés**. Un message demandant la confirmation de la suppression par l'utilisateur doit être affiché.
- **Bouton Annuler**
  - Permet la fermeture du formulaire en annulant l'action.

## Énumérations

### Énumération "Province" : Déjà fournie

- ☐ Contient la liste des provinces canadiennes avec leurs descriptions.

### Énumération "TypCourse" : Déjà fournie

- ☐ Contient la liste des types de courses possible.

### Énumération "Categorie" : Déjà fournie

- ☐ Contient la liste des catégories des coureurs représentant le sexe et l'âge des coureurs.

### Énumération "EtatFormulaire" : Déjà fournie

- ☐ Contient la liste des états possible d'un formulaire.





## Classes

### Classe statique "Utilitaire" : DÉJÀ FOURNIE

Classe statique contenant des méthodes utilitaires pour la lecture et l'écriture des données dans le fichier.

### Classe statique "UtilEnum" : DÉJÀ FOURNIE

Classe statique pour les énumérations. Celle-ci fournit des méthodes permettant d'obtenir les descriptions associées aux constantes des énumérations lorsque celles-ci sont disponibles.

### Classe "Coureur" : DÉJÀ FOURNIE ET À COMPLÉTER

Classe représentant un coureur. Consultez les commentaires XML déjà présents dans la classe pour plus d'information. Dans cette classe, **vous devez implémenter les éléments suivants** :

- ☐ Constantes publiques nécessaires à la validation.
- ☐ Validation dans les propriétés suivantes :
  - ☐ Dossard : Doit être une valeur supérieure ou égale à 1.
  - ☐ Nom : Ne doit pas être nul ou vide et doit contenir au moins 3 caractères.
  - ☐ Prénom : Ne doit pas être nul ou vide et doit contenir au moins 3 caractères.
  - ☐ Catégorie : Doit être une valeur correspondant aux valeurs de l'énumération.
  - ☐ Ville : Ne doit pas être nul ou vide et doit contenir au moins 4 caractères.
  - ☐ Province : Doit être une valeur correspondant aux valeurs de l'énumération.
  - ☐ Rang : Ne peut pas être modifié. Il est déterminé selon le temps de course du coureur.

**Vous devez lancer le bon type d'exception avec un message adéquat lorsqu'une contrainte n'est pas respectée.**

- ☐ Constructeur paramétré ayant pour l'initialisation de toutes les propriétés.
- ☐ Méthodes :
  - ☐ **public override string ToString()** : Retourne la représentation d'un coureur sous forme de chaîne de caractère de la manière suivante : **Dossard Nom, Prenom Catégorie Temps**. Cette représentation est utilisée l'alignement du texte pour l'affichage la liste des résultats.

- ☐ **CompareTo(Coureur other)** : Permet de trier les coureurs selon leur temps de course.
- ☐ **public override bool Equals(object? obj)** : Permet de comparer deux coureurs. Deux coureurs sont identiques s'ils ont le même nom, prénom, ville et province.
- ☐ **public static bool operator ==(Coureur coureurGauche, Coureur coureurDroit)** : Permet de comparer deux coureurs.
- ☐ **public static bool operator !=(Coureur coureurGauche, Coureur coureurDroit)** : Permet de comparer deux coureurs.
- ☐ Commentaires XML pour tous les constantes, attributs, constructeurs et méthodes. **N'oubliez pas d'indiquer si requis les commentaires sur les paramètres, valeurs de retour et exceptions.**

### Classe "Course" : DÉJÀ FOURNIE ET À COMPLÉTER.

Classe représentant course. Consultez les commentaires XML déjà présents dans la classe pour plus d'information. Dans cette classe, **vous devez implémenter les éléments suivants** :

- Constantes publiques nécessaires à la validation.
- Validation dans les propriétés suivantes :
  - ☐ Nom : Ne doit pas être nul ou vide et doit contenir au moins 3 caractères. Doit être converti en majuscule.
  - ☐ Ville : Ne doit pas être nul ou vide et doit contenir au moins 4 caractères.
  - ☐ Province : Doit être une valeur correspondant aux valeurs de l'énumération.
  - ☐ TypeCourse : Doit être une valeur correspondant aux valeurs de l'énumération.
  - ☐ Distance : Doit être une valeur supérieure à 1.

**Vous devez lancer le bon type d'exception avec un message adéquat lorsqu'une contrainte n'est pas respectée.**

- Constructeur avec paramètre qui initialise toutes les propriétés ainsi qu'une nouvelle liste de coureurs.
- Méthodes :
  - ☐ **public void AjouterCoureur(Coureur coureur)** : Permet l'ajout d'un coureur à la liste des coureurs. Cette méthode doit lancer des exceptions dans les cas suivants :

- Le coureur ne peut pas être nul.
- Le numéro de dossard du coureur ne doit pas être utilisé par un autre coureur.
- Un coureur avec les mêmes informations (sans le numéro de dossard) existe déjà.

La liste des coureurs doit être triée à la suite de l'ajout du coureur.

- **public Coureur ObtenirCoureurParNoDossard(ushort noDossard)** : Permet d'obtenir un coureur à partir de son numéro de dossard. Si aucun coureur ne porte le numéro de dossard recherché, alors la valeur nulle est retournée sinon le coureur trouvé est retourné.
  - Cette méthode doit lancer une exception si le numéro du dossard est plus petit que 1.
- **public void SupprimerCoureur(Coureur coureur)** : Permet de retirer un coureur de la liste de coureurs. Cette méthode doit lancer des exceptions dans les cas suivants :
  - Le coureur ne doit pas être nul.
  - Le coureur doit exister dans la liste.
- **private TimeSpan CalculerTempsCourseMoyen()** : Permet de calculer le temps moyen de la course. Les coureurs ayant abandonné la course sont exclus du calcul. Retourne le temps moyen de la course.
- **public void TrierCoureurs()** : Permet de trier la liste des coureurs selon le temps de course en ajustant le rang des coureurs.
- **public override string ToString()** : Retourne la représentation d'une course sous forme de chaîne de caractère de la manière
- **public override bool Equals(object? obj)** : Permet de comparer deux courses. Deux courses sont identiques s'ils ont le même nom, date, ville, province, type et distance.
- **public static bool operator ==(Course courseGauche, Course courseDroite)** : Permet de comparer deux coureurs.
- **public static bool operator !=(Course courseGauche, Course courseDroite)** : Permet de comparer deux courses.

### Classe "GestionCourse" : À COMPLÉTER.

Classe permettant la gestion d'une course et ses coureurs. Consultez les commentaires XML déjà présents dans la classe pour plus d'information. Dans cette classe, **vous devez implémenter les éléments suivants** :

- Attributs :
  - **\_courses** (Courses) : Les courses.
- Propriétés :
  - ☐ Des propriétés **publiques en lecture** pour tous les attributs (get).
  - ☐ Des propriétés **publiques en écriture** pour tous les attributs (set).
- Constructeurs :
  - ☐ Un seul constructeur ayant comme paramètre le **chemin d'accès au fichier de courses et celui des coureurs** et qui charge les courses.
- Méthodes :
  - ☐ **private void ChargerCourse(string cheminFichierCourses, string cheminFichierCoureurs)** : Permet de charger les données de la course et ses coureurs. Doit utiliser la fonction ChargerDonnees de la classe Utilitaire. Une fois les données chargées dans la course, les coureurs sont chargés. Cette méthode doit lancer des exceptions si les noms des fichiers sont nuls, vides ou ne contiennent que des espaces.
  - ☐ **private ChargerCoureurs(Course course, string cheminFichierCoureurs)**: Permet de charger les coureurs dans la course. Doit utiliser la fonction ChargerDonnees de la classe

Utilitaire. Cette méthode doit lancer une exception si la course est nulle ou si le nom du fichier est nul, vide ou ne contient que des espaces.

- **public void AjouterCourse(Course course)** : Permet l'ajout de la course à la liste des courses. Doit lancer des exceptions dans les cas suivants :

- La course est nulle
- Une course ayant la même information existe déjà dans la liste.

La liste des courses doit être triée à la suite de l'ajout.

- **public bool SupprimerCourse(Course course)** : Permet de retirer une course de la liste des courses. Cette méthode doit lancer des exceptions dans les cas suivants :

- La course ne doit pas être nulle.
- La course doit exister dans la liste.

- **public void EnregistrerCourses(string cheminFichierCourses, string cheminFichierCoureurs)** : Permet l'enregistrement des données de la course et les données des coureurs dans les fichiers correspondants. Cette méthode doit lancer des exceptions dans les cas où les noms des fichiers sont nuls, vides ou ne contiennent que des espaces.

## Fichiers texte

- Toutes les **données** pour ce travail proviennent de **deux fichiers texte au format CSV** (« **courses.csv** » et « **coureurs.csv** »). Ces fichiers doivent être placés dans le dossier « **C:\data-420-14B-FX\TP2** » qui doit être à la racine du lecteur « **C** ».

## Directives

- Votre travail final doit se trouver sur GitHub. Vous devez vous créer un nouveau **projet privé** et m'ajouter comme membre du projet.
- La **structure précise des classes** est fournie aux **annexes II (diagramme de classes)**; il est impératif de **respecter à la lettre** cette structure sans rien ajouter ou omettre. Vous devez aussi **respecter** les **identificateurs** et les **types** qui sont précisés dans ces annexes.
- Vous devez gérer les exceptions à l'aide des **try...catch** dans vos formulaires.
- Vous devez partir de la **solution de départ fournie par le professeur** (« 420-14B-TP2-A24 ») qui inclut le projet ainsi que certaines classes à compléter.

- Vous devez respecter les **conventions** fournies par le professeur pour tous les **identificateurs en C#** (variables, méthodes, classes, contrôles de formulaire, etc.) ainsi que les autres conventions de codage du cours Programmation Orientée Objet. Voir le fichier "**Conventions pour les identificateurs en C#.pdf**" et "**Normes de codage C#.pdf**" disponible sur le site du cours. De plus, tout identificateur doit avoir **un nom significatif** considérant le contexte dans lequel il apparaît.
- Vous devez mettre des **commentaires en format XML** pour tous les éléments suivants : **classes, énumérations, constantes, propriétés, constructeurs et méthodes**. À noter que pour les classes correspondant à des formulaires, **il n'est pas nécessaire de mettre des commentaires XML pour les méthodes liées aux événements dans l'interface graphique**; cependant, vous devez en ajouter sur toutes les méthodes que vous ajoutez la classe du formulaire.
- Vous devez faire des "**Commit**" par fonctionnalité ayant des noms significatifs et des "**Push**" de façon régulière sur votre dépôt Git.

## Précisions sur l'évaluation

Voici une ébauche du barème de correction. Il pourrait être modifié lors de la correction.

Éléments de compétence	Critères de performance	Points
00Q6 - Exploiter les principes de la programmation orientée objet		
5. Programmer des classes.	5.1. Choix approprié des instructions, des types de données élémentaires et des structures de données.	25
	5.2. Organisation logique des instructions.	
	5.4. Intégration correcte des classes dans le programme.	
	5.3. Programmation correcte des messages à afficher à l'utilisatrice ou à l'utilisateur. Valider les entrées d'un utilisateur. Attraper et lancer des exceptions.	10
	5.6. Respect de la syntaxe du langage. 5.7. Respect des règles de codage.	10
	5.5. Fonctionnement correct du programme.	35
6. Documenter la programmation.	6.1. Notation claire de commentaires dans le code informatique. 6.2. Notation claire de la documentation d'aide à la programmation.	10

7. Appliquer la procédure liée à la gestion des versions de programmes.	7.1. Configuration correcte du système de gestion de versions. 7.2. Soumission méthodique du code modifié.	10
<b>Pénalités</b> <ul style="list-style-type: none"><li>• Non-respect des gabarits fournis.</li><li>• Retard.</li><li>• Remise incorrecte du travail.</li><li>• Français.</li><li>• D'autres éléments selon le besoin.</li></ul>		

**Remise**

- **Aucune remise papier.**
- **À remettre sur LEA :** Un fichier nommé "**InfoGit\_{NoDA1}\_{NoDA2}.txt**" contenant l'URL de votre dépôt *GitHub* ainsi que le nom des coéquipiers.

## Annexe I : Diagramme de classes

