

Cours 420-4A6-LI Applications monopostes I Hiver 2020 Cégep Limoilou Département d'informatique	Tp 3 Travail intégrateur 25 %
---	-------------------------------------

Nom : _____

Gr : 1

Nom : _____

Objectif

- Créer, un modèle (diagramme) de classes pour illustrer les composants de votre application.
- Utiliser du FXML pour créer une interface graphique et des boîtes de dialogue personnalisées pour mettre à jour de l'information.
- Lier du FXML à du JavaFX.
- Respecter le modèle MVC avec FXML vu dans le cours.
- Utiliser des concepts à partir d'information prise sur le web.
- Utiliser une base de données et des fichiers d'information.

Activités à réaliser

- Vous avez à programmer une application de gestion de permis animalier.

Gestion de permis animaliers

Fichier Permis Listes de données Aide

Liste de numéros de permis

Permis

Numéro Date de début

Territoire Date de fin

Animal

Nom Type

Sexe ☐ Mâle ☐ Femelle ☒ Inconnu Poids en Kg

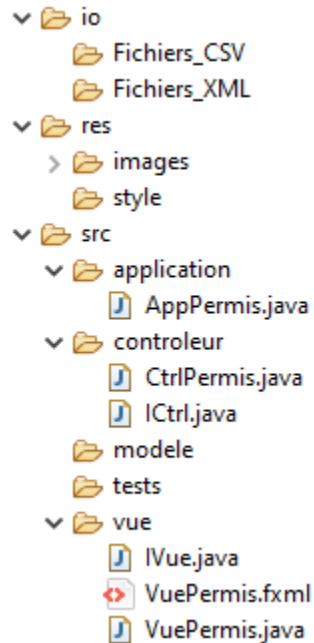
Date de naissance Couleur

Caractéristiques supplémentaires ☐ Vacciné ☐ Stérilisé ☐ Micropuce ☐ Dangereux

Nouveau Ajouter Modifier Quitter ?

À remettre et/ou à présenter

- Pour le code source, les ressources et les fichiers de données vous devez respecter minimalement l'arbre de packages, dossiers et de fichiers, suivant (les noms et le découpage). Je vous laisse décider du reste des ressources que vous aurez besoin :



- Pour la disposition des contrôles et leur mise en forme, il faut respecter l'image écran présentée ci-haut et les consignes du présent travail.
- Tout votre code doit-être documenté avec de la JavaDoc.
- Des tests unitaires JUnit doivent être faits pour vos classes métiers du package « modele ». Une preuve de couverture sera demandée.
- Une **démonstration** du fonctionnement de votre code devra obligatoirement être faite dans la semaine du **18 mai 2020**.
- Avant de faire votre démonstration, vous devez remettre votre code du projet « Eclipse » complet, compressé (.zip) dans un fichier du nom « noms des équipiers – Tp3.zip » en le déposant sur le réseau dans le dossier « Q:\!Dépôt\JGoulet\420-4A6-H20\Tp 3 – Code ».

Durée de l'activité

- Ce travail peut être réalisé en équipe de 2 maximum.
- Vous avez jusqu'au 22 mai à 12h pour faire votre démo. Nous allons devoir faire un horaire pour la planification des démos :o).

Avant de commencer

- Il faut avoir réalisé quelques exercices dans les formatifs 6 et 7.
- Il faut avoir lu les documents des notes de cours des semaines 8, 10 et 11 inclusivement.
- Il faut avoir compris les exemples fournis par le professeur pour les semaines 8, 10 et 11.

Étape 1 : Stratégie et diagramme de classe (non obligatoire, mais fortement recommandé)

- Votre première tâche consiste à fabriquer un diagramme de classes de toutes vos classes impliquées dans votre architecture MVC. Utilisez l'outil « **Visual Paradigm** » ou « **ObjectAid** » installé dans Eclipse (revoir, dans le formatif 2, les références à la fin du cas 1).

Cette étape est fortement recommandée pour vous permettre de prendre plusieurs décisions. En plus

de vous permettre de réfléchir à votre stratégie de conception, il vous permettra de déterminer les interfaces, les classes abstraites et les héritages. Bien entendu, il n'y a rien de parfait, faites de votre mieux. Je peux, si vous me le demandez, regarder votre diagramme avec vous. Commencez par lire le présent travail au complet, ce qui va vous permettre de faire un premier diagramme papier de haut niveau.

- Comme la section « À remettre ou à présenter » le précise, il n'y a rien à remettre à cette étape.

Étape 2 : Programmation et validation des classes métiers

- Programmer vos classes en lien avec les données de la partie « modèle » du MVC, les classes de manipulation des données avec la BD et de lecture et écriture dans les fichiers.
- Programmer vos classes en lien avec l'application pour permettre de répondre aux événements de la vue (ajout, modification et suppression de données) de la partie « modèle » du MVC, les classes de traitements des données, de la gestion de certaines des listes et les classes de données transitoires.
- Les tests unitaires doivent être faits pour vos classes du package « modele ». Une preuve de couverture de code sera demandée.

Étape 3 : Réalisation et programmation du fonctionnement de l'interface

Les composants :

- Reproduire la vue telle que présentée dans cette figure et utiliser une feuille de style pour améliorer son apparence (le css est obligatoire, mais son contenu est laissé à votre convenance). Une preuve sera demandée pour voir l'utilité de votre feuille de style.

1. Ce champ permet de rechercher dans la liste des numéros de permis. La recherche se fait lorsqu'on appuie sur le bouton (loupe) ce qui positionne le numéro de permis visible dans la liste et présente les valeurs de ce permis dans les champs de la fenêtre, c'est le permis courant.
2. Le champ « Numéro » de permis, il n'est pas éditable, il est attribué par votre le système et doit être unique. Les numéros supprimés ne sont pas réutilisés.
3. Les champs « Date de début », « Date de fin » et « Date de naissance » sont de vrais champs date (composant date). Ils ne sont pas éditables, mais la date peut être modifiée à l'aide du composant.

Lors de l'ajout d'un nouveau permis, à l'initialisation des champs, la date de début prend la date du jour et la date de fin prend la valeur de la date de début plus 1 an.

Attention :

- La date de début doit toujours être plus ancienne que la date de fin de 1 mois au moins.
 - La date de naissance de l'animal doit toujours être plus ancienne ou égale à la date de début.
 - Si la date de naissance n'est pas connue, laissez le champ vide.
4. Ce bouton permet de supprimer l'enregistrement courant avec une demande de confirmation à l'utilisateur. Le permis est supprimé de la BD et de la liste de numéro de permis. Le nouveau permis courant est le numéro précédent de celui supprimé dans la liste. Si la liste est vide, donc aucune sélection possible, les champs du permis sont vides.
 5. Les champs « Territoire » et « Type » sont des listes déroulantes non éditables. Si le territoire ou le type ne sont pas définis dans les données, le champ doit afficher le mot « Inconnu ». Le bouton à côté du champ permet d'ouvrir une boîte de dialogue personnalisée qui permet de faire sa gestion (ajout et modifier seulement) de nouveaux territoires ou de nouveaux types. Après un ajout ou une modification, la liste est mise à jour.

Voici les valeurs à mettre dans les tables (BD) pour les 2 listes de départ :

Liste de territoires au départ du système :

Inconnu, Vimont, Sainte-Dorothée, Sainte-Rose, Saint-François, Chomedey, Pont-Viau, îles-Laval, Laval-des-Rapides, Laval-Ouest, Laval-sur-le-Lac, Saint-Vincent-de-Paul, Duvernay, Auteuil, Fabreville

Liste de type au départ du système :

Inconnu, Chat, Chien

Le territoire par défaut est « Inconnu » et le type par défaut est « Chat ».

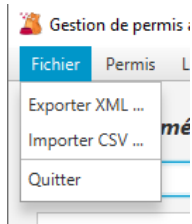
NOTE : Voici un lien à partir duquel on vous présente comment lier et administrer les données d'une boîte de dialogue personnalisée en FXML : <https://code.makery.ch/fr/library/javafx-tutorial/part3/>.

6. Le champ « Poids » peut prendre les valeurs réelles entre 0 à 500 kg. La valeur 0 ne correspond à aucune valeur.
7. Le champ « Couleur » est une liste déroulante éditable. Vous pouvez donc choisir une couleur à partir des couleurs déjà disponibles des autres animaux du système (BD) ou directement entrer la couleur voulue. La nouvelle couleur sera sauvegardée avec les données de l'animal et ajoutée à la liste.
8. La case à cocher « Dangereux » mettra son texte en **rouge et gras** si elle est sélectionnée, pour permettre d'attirer l'attention de l'utilisateur.
9. Les boutons « Nouveau » et « Ajouter » permettent d'ajouter un nouveau permis. Ils sont mutuellement dépendants. Lorsque je suis sur un enregistrement déjà existant le bouton « Nouveau » est disponible, mais pas le bouton « Ajouter », c'est le mode normal. Si je clique sur le bouton « Nouveau » le système me propose un nouveau numéro de permis, met les autres champs à des valeurs par défaut, désactive le bouton « Nouveau » et active le bouton « Ajouter », c'est le mode ajout. Si je clique sur le bouton « Ajouter », la filtration des données s'effectue (notification s'il y a des problèmes) et si tout est conforme, le nouveau permis est enregistré dans la BD et son numéro est ajouté à la liste de permis dans l'application. Il y a aussi une notification pour les données non valides. Après l'ajout, l'application revient en mode normal avec le nouveau permis, comme permis courant.
10. Le bouton « Modifier » est disponible seulement si une des valeurs du permis courant a été modifiée et qu'on est en mode normal. Lorsqu'il est cliqué, la filtration des données s'effectue (notification s'il y a des problèmes), la validation aussi avec notifications et si tout est conforme, les modifications du permis sont enregistrées dans la BD. L'application revient en mode normal avec le permis modifié, comme permis courant et le bouton « Modifier » désactivé.
11. La liste de permis permet d'afficher, en ordre croissant, tous les numéros de permis enregistrés dans la BD. Le numéro du permis courant (sélectionné) est toujours visible dans la liste. À chaque changement

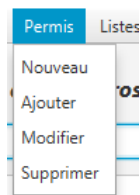
de sélection dans la liste, les champs du permis courant se mettent à jour. C'est aussi vrai pour la recherche et l'ajout d'un nouvel enregistrement.

- Le sexe par défaut est « Femelle ».
- Le bouton « Quitter » quitte l'application avec le consentement de l'utilisateur.
- Le bouton « ? » présente un court texte de ce que fait cette application dans une boîte de dialogue d'information

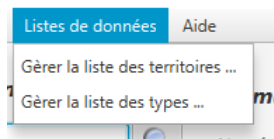
Les menus :




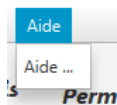
- « Exporter XML ... » permet de sauvegarder tous les permis contenus dans la BD dans un fichier XML selon un format prédéterminé.
- « Importer CSV ... » permet de lire un fichier CSV, du même format que celui utilisé dans les formatifs 1 et 3.
- Même action que le bouton « Quitter ».



- Les 4 actions de ce menu ont le même comportement et font les mêmes actions que les boutons de l'interface.



- Les 2 actions de ce menu ont le même comportement et font les mêmes actions que les boutons «  » de l'interface qui sont adjacents aux listes de territoires et de types.



- Même action que le bouton « ? », de l'interface graphique.

XML en Java et le format du fichier XML :

- Les règles d'écriture pour le XML sont très spécifiques et simples, l'essentielle de ces règles est présenté dans le document « Spécifications XML.pdf » disponible dans le dossier de ce Tp.
- Pour vous permettre d'écrire ou de lire un document XML en java, je vous réfère à un document sur le web <https://cynober.developpez.com/tutoriel/java/xml/jdom/>. Comme indiqué dans cette référence vous aurez besoin de JDOM une librairie que vous allez intégrer à votre code Java. Voici le lien pour télécharger cette dernière <http://www.jdom.org/index.html>. Téléchargez la version 2.0.6.
- Je propose ici, un format de fichier XML :

```
<?xml version="1.0" encoding="UTF-8"?>

<animaux>
  <animal type="chat" sexe="mâle">
    <permis numero="66666">
      <date_debut jour="26" mois="08" année="1973"></date_debut>
      <date_fin jour="26" mois="08" année="1974"></date_fin>
    </permis>
    <territoire>Sainte-Rose</territoire>
    <nom>pompon</nom>
  </animal>
</animaux>
```

```
<date_naissance jour="15" mois="06" année="1973"></date_naissance>
<couleur>tabby gris</couleur>
<poids metrique="kg">2.1</poids>
<caracteristiques>
  <vaccine>1</vaccine>
  <sterilise>1</sterilise>
  <micropuce>0</micropuce>
  <dangereux>0</dangereux>
</caracteristiques>
</animal>
<animal type="chien" sexe="femelle">
  <permis numero="88888">
    <date_debut jour="13" mois="07" année="1983"></date_debut>
    <date_fin jour="26" mois="08" année="1984"></date_fin>
  </permis>
  <territoire>Sainte-Rose</territoire>
  <nom>pompon</nom>
  <date_naissance jour="26" mois="05" année="1983"></date_naissance>
  <couleur>brun chocolat</couleur>
  <poids metrique="kg">6.3</poids>
  <caracteristiques>
    <vaccine>1</vaccine>
    <sterilise>1</sterilise>
    <micropuce>0</micropuce>
    <dangereux>1</dangereux>
  </caracteristiques>
</animal>
.
.
.
</animaux>
```