TP JAVA n°6: Feuille de route pour la mise en place des uses cases `CreerPatient` et `ListerPatients` dans le cadre d'une application interactive

Dans cette feuille de route, vous retrouverez étape par étape les modifications à apporter à votre application `CabinetMedical` pour mettre en place une application interactive à partir des patterns `MVC/DAO/DTO`.

Etape 1 : Mise en place dans l'architecture logicielle de la Vue (`IHM`) et du Contrôleur (`Ctrl`):

Vous devez mettre en place 2 nouvelles classes :

- dans le paquetage `com.iut.cabinet.`**`presentation`** une classe **`GererPatientIHM`** qui permettra de mettre en place le représentation visuelle à l'écran de l'application (***vue***). Cette classe est disponible sous la zone libre, importez la.
- dans le paquetage `com.iut.cabinet.`**`application`** une classe **`GererPatientCtrl`** qui assurera l'interface entre l'utilisateur et le modèle (contrôleur). Créer cette classe.

Pour exécuter l'application, créer dans le package `com.iut.cabinet.`**`essai`**, la classe **`EssaiCabMed_console`**. Implémenter la méthode `main` de la classe qui ne sera composée que de l'instruction suivante : **`new GererPatientIHM();`**
Exécuter alors la classe `EssaiCabMed_console`.

Astuce : *Afin de faciliter de la phase de développement de l'application, vous pouvez coder directement la méthode `main` dans la classe `GererPatientIHM` (aller voir en fin de classe… elle s'y trouve déjà…)*

Etape 2 : Association du Contrôleur à la Vue utilisée:

Dans la classe `GererPatientIHM`, déclarer l'**attribut** suivant:

```
GererPatientCtrl ctrlUseCase;
```

Pour que le contrôleur soit créé au démarrage de la vue, l'instanciation de l'objet *`ctrlUseCase`* devra donc être la ***toute première instruction du constructeur*** de la classe `GererPatientIHM`.

```
ctrlUseCase = new GererPatientCtrl();
```

ou à défaut, on peut écrire directement en une seule ligne la déclaration et l'instanciation :

```
GererPatientCtrl ctrlUseCase = new GererPatientCtrl();
```

Etape 3 : Mise en place des `DTO` (Data Transfert Object) dans la couche de service (`user`)

Importer dans le paquetage `com.iut.cabinet.`**`user`**,
les classes **`PersonneDTO`**, **`PatientDTO`** et **`AdresseDTO`** disponibles sur la zone libre.
Consulter le code de ces classes.

Etape 4 : Mise en place des `Helper` dans la couche applicative pour favoriser la conversion des objets métiers ↔ `DTO`

Depuis la zone libre, importer dans le paquetage *`com.iut.cabinet.`***`application`** *:*

→ la classe **`HelperPatient`** qui propose les méthodes suivantes :

```
public static PatientDTO toPatientDTO (Patient unPatient) {…}
public static Patient toPatient (PatientDTO unPatientDTO) {…}
```

→ ainsi que la classe **`HelperAdresse`**

→ ainsi que la classe **`HelperException`**. Une `HelperException` sera lancée si on transmet en paramètre d'entrée une référence nulle. En effet, plus loin dans le programme, il pourrait y aurait une erreur à l'exécution si on appelait un `getteur` sur une référence nulle...

Remarque : *Notez également dans le code, que ce n'est que dans le `Helper`, lors de la transformation **`DTO→Metier`**, que les exceptions métier (**`CabinetMedicalException`**) risquent de se déclencher…*
Bien sûr, on traite pas les exceptions métier dans les `Helper`, mais on les fait les remonter…

Etape 5 : Création d'un objet `PatientDTO` dans la couche présentation (`GererPatientIHM`)

En fin de saisie (dans la méthode **`creerPatient`** de la classe **`GererPatientIHM`**), stocker les données saisies dans un objet `patDTO` de type `PatientDTO` (avec un ascendant `null` et un objet de type `AdresseDTO` que vous aurez préalablement également créé.)

```
PatientDTO patDTO = new PatientDTO(…constructeur à compléter avec les données
  saisies : consulter la classe PatientDTO pour connaître l'ordre des paramètres);
```

Etape 6 : Transmission au Contrôleur de l'objet `PatientDTO` (créé dans la Vue)

Afin de matérialiser l'envoi du message de la **Vue** au **Contrôleur**, vous devez effectuer dans la couche présentation (classe **`GererPatientIHM`**) à la fin de la méthode **`creerPatient`** c-a-d après la saisie :

- la création d'un objet de type ***`PatientDTO`*** (c'est le `patDTO` créé à l'étape 5 précédente),
- suivi de ***l'appel à la méthode du contrôleur*** : **`ctrlUseCase.creerPatient(patDTO);`**

Etape 7 : Réponse du Contrôleur au message envoyé par la Vue…
… ou le rôle du Contrôleur dans la création d'un `Patient` …

→ Déclarer dans la couche application (**classe `GererPatientCtrl`**), une méthode `creerPatient` :

```
public void creerPatient(PatientDTO unPatientDTO)
{ … }
```

→ 1. Créer un objet métier à partir du DTO :

Coder la méthode `creerPatient` en commençant par déclarer un objet `unPat` de type `Patient` et instancier-le en appelant la méthode `toPatient` de la classe `HelperPatient` prenant comme paramètre l'objet de type `PatientDTO` provenant de la vue :

```
Patient unPat; // Déclaration
unPat = HelperPatient.toPatient(unPatientDTO); // Instanciation
```

..***propager vers la couche appelante les exceptions*** susceptibles de se déclencher :

```
public void creerPatient(PatientDTO unPatientDTO)
            throws CabinetMedicalException, HelperException
{ … }
```

Etape 7 (suite) : Réponse du Contrôleur au message envoyé par la Vue…
… ou le rôle du Contrôleur dans la création d'un `Patient` …

→ 2. Assurer la persistance des objets métiers dans un Fichier à l'aide d'un `DAO`

Lorsque le support de persistance est un fichier, la création d'un `Patient` revient à ajouter un `Patient` dans une liste de persistance grâce au processus suivant :

- **a. charger** la liste de `Personne` initialement contenue dans le fichier
- b. créer un nouvel objet de type `Patient`
- c. ajouter cet objet à la liste
- **d**. **sauvegarder** la nouvelle liste (incluant le nouveau `Patient`)

Pour commencer nous traitons les points **b** et **c** de ce processus :
Dans la méthode `creerPatient` du contrôleur (`GererPatientCtrl`) :

- Déclarer `maListe` de manière à manipuler une liste de `Personne` au travers d'une *collection* :

```
Collection<Personne> maListe ;
```

- placer-vous après la création du `Patient unPat` (c-a-d après l'appel au `Helper`) et ajouter ce nouveau `Patient` à la collection (`maListe`).

Les points **a** et **d** qui concernent la sérialisation/désérialisation de la liste seront réalisés à l'aide d'une ***couche d'accès aux données*** (**`DAO`**). Cette nouvelle couche permet de rendre indépendant la couche métier et le support de persistance
Compléter la méthode `creerPatient` du contrôleur (`GererPatientCtrl`) :

- Pour la ***désérialisation***, compléter la déclaration de `maListe` :

```
Collection<Personne> maListe=PersonneDAOFichier.findAllPersonnes();
```

Une seule ligne suffit puisque la méthode `findAllPersonnes` renvoie une référence valide (non nulle) sur une Collection.
Relancer vers la Vue, l'éventuelle `CabinetTechniqueException`.

- En fin de méthode `creerPatient`, effectuer la ***sérialisation*** de la Collection à l'aide de la méthode **`storeAllPersonnes`** de la classe **`PersonneDAOFichier`**.

Afin de respecter le plan type détaillé, n'oubliez pas d'informer l'utilisateur après la désérialisation que le nouveau patient a bien été rajouté…

Etape 8 : Traiter dans la Vue (couche présentation : `GererPatientIHM`) les exceptions lancées par le Contrôleur

→ Dans la méthode `creerPatient` de la classe `GererPatientIHM` , il faut désormais attraper les éventuelles exceptions et informer l'utilisateur de la survenue d'une erreur (pour l'instant, on se contentera d'afficher le message embarqué par l'exception).
Aidez-vous de l'assistant d'Eclipse (petite croix rouge) pour générer automatiquement les `try..catch` et compléter les `catch` en affichant le message délivré par de l'exception.

→ A l'aide par exemple d'un booléen et d'une boucle `while` modifiez ensuite votre code pour pouvoir recommencer la saisie tant qu'une exception est récupérée (c-a-d tant que l'on passe dans un bloc `catch` …)

Etape 9 : *Tester votre application ! ! ! !*

→ Dans la méthode `creerPatient` du contrôleur (`GererPatientCtrl`), juste après l'appel du `Helper`, procéder tout d'abord à un affichage « temporaire » (c-a-d uniquement utilisé durant la phase de développement) de votre objet métier `Patient` (`unPat`) pour contrôler la validité de l'étape de création d'un objet métier par le contrôleur (étapes 5,6 et 7 du diagramme de séquence).
Rappel de nir « valides » :

`260058700112367` `297112A10102401`
`191128708545628` `168072B12345652` … *et votre propre nir ! ! !*

→ Afin de faciliter vos tests, vous pouvez récupérer sur la zone libre **`JeuEssaiPatient.txt`**, qui est un jeu d'essai à personnaliser, qui pourra être utilisé durant votre phase de développement afin de vous éviter de perdre trop de temps dans la saisie des caractéristiques…
Effectuer quand même avant au moins un test complet avec la saisie des données …

Etape 10 : *Implémentation du cas Lister tous les patients*

→ Dans la **Vue** (**`GererPatientIHM`**), implémenter pour l'instant la méthode **`listerPatients`** de la manière suivante :

```
public void listerPatients()
      {
        // Déclaration d'une Collection de PatientDTO
          Collection<PatientDTO> maListe=null;

        // Récupération de maListe par appel à la méthode listerPatient
        // du contrôleur de use case
        // … à vous de coder cet appel …

          for(PatientDTO unPatientDTO : maListe)
          {
            // Affichage restreint d'informations concernant le patient
            System.out.println("-------------------------------------");
            System.out.println(" NIR : "+ unPatientDTO.getNir());
            System.out.println(" Nom : "+ unPatientDTO.getNom());
            System.out.println(" Prenom : "+ unPatientDTO.getPrenom());
            System.out.println(" DateNaissance : "+
                                  DateUtil.toString(unPatientDTO.getDateNaissance()));
            System.out.println("-------------------------------------");
          }

      }
```

→ Dans le **Contrôleur** (**`GererPatientCtrl`**) :

- Ecrire une méthode qui permet de renvoyer la collection de `Patient` contenue dans le fichier. Cette méthode aura la signature suivante :

```
public Collection<PatientDTO>listerPatients() {// … à vous de coder }
```

La collection que vous allez récupérer depuis le DAO est une collection de `Personne` et nous souhaitons transmettre une **collection de `Patient`**… Tenez-en compte dans votre code …

- Vous relancerez les éventuelles exceptions à la **Vue** qui les traitera…

→ ***Tester votre application ! ! ! !*** Commenter tous les affichages que vous aviez écrit dans la méthode `creerPatient` du contrôleur (`GererPatientCtrl`).
Pour tester votre application, il ne vous reste plus qu'à jouer entre les options 1 et 4 du menu :
`1.Creer un patient et 4.Lister tous les patients`