

**Voir à la fin de cet énoncé pour l'évaluation et remise de ce labo.**

## Objectifs

- POO
- Utilisation de fichiers texte et d'objets
- Utilisation de conteneurs
- Sérialisation
- Utilisation d'interfaces de base de Swing et AWT

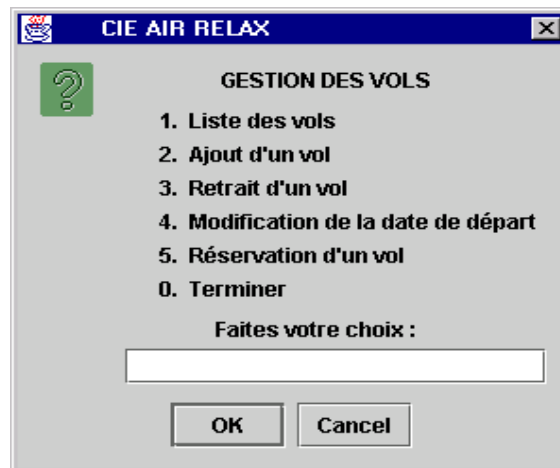
## Sujet

On vous engage afin de réaliser un projet de gestion des vols d'une compagnie aérienne. Votre projet sera utilisé par différentes compagnies aériennes. Chaque compagnie dispose d'un fichier (des vols déjà existants) dont **le nom est celui du nom de la compagnie suivie de l'extension .txt**. Chaque enregistrement des fichiers contient :

- Le numéro du vol
- La destination du vol
- La date de départ du vol
- Le nombre total des réservations à date

En partant, les enregistrements sont par ordre croissant du numéro du vol. **On ne connaît pas le nombre d'enregistrements qu'il y a dans le fichier.**

Dans votre projet, la classe qui contiendra le « main » devra faire afficher un menu : par exemple, pour la compagnie « **Cie Air Relax** » nous aurons le menu suivant, à l'aide de `JOptionPane.showInputDialog`



**N'oubliez pas de produire un message d'erreur si le choix n'est pas entre 0 et 5**

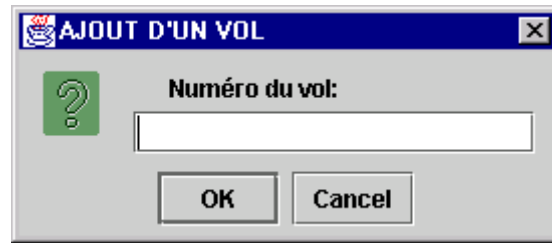
## Travail à faire

Créez un projet JAVA nommé **Labo1**. Dans ce même projet, au premier niveau vous allez créer un dossier nommé documents (la documentation du projet).

Ce projet va comporter 5 classes soient :

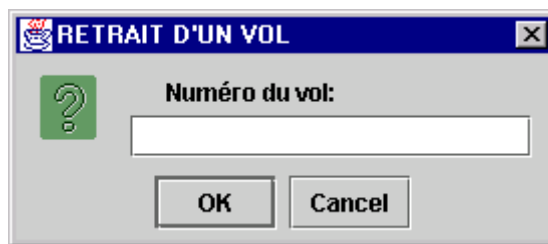
1. La classe **Date** ayant les attributs entiers **jour**, **mois**, **an**, le constructeur par défaut et celui avec 3 paramètres, les méthodes « **get** » et la méthode **toString** qui retourne la chaîne contenant les attributs séparés par "/" et en plus les attributs jour et mois dans le format décimal "00". Un exemple de classe Date avec validation vous sera donné en classe.
2. La classe **Vol** dont les caractéristiques sont :
  - attributs **d'instances** : numéro du vol, sa destination, la date de départ (objet de la classe Date), le numéro d'avion affecté à vol et le nombre total de réservations à date
  - le constructeur permettant d'initialiser les 5 attributs d'instances avec les valeurs reçues en paramètres. Le numéro de vol au départ sera 0. À la création d'un vol vous allez demander le numéro de l'avion à affecter à ce vol.
  - les méthodes « **set** » pour seulement le total des réservations et la date de départ
  - les méthodes « **get** » pour tous les attributs
  - la méthode **toString**( ) qui retourne une chaîne contenant les attributs séparés par des "\t"
3. La classe **Avion** que contiendra comme attributs d'instance : numéro de l'avion, type avion (exemple Boeing 747, Airbus 320 ou autre), nombre de places de l'avion et d'autres attributs si vous voulez.
4. La classe **Compagnie** dont les caractéristiques sont :
  - la constante de classe publique MAX\_PLACES correspondant au nombre de places que contient l'avion affecté à ce vol.
  - attributs d'instances :
    - le nom de la compagnie,
    - un **ArrayList listeVols** d'objets de la classe Vol (**on ne fait que déclarer la référence**)
  - attribut de classe :
    - le nombre de vols actifs (un compteur)
  - le constructeur à 1 paramètre qui permet d'initialiser le nom de la compagnie. De plus, le constructeur va instancier le **ArrayList listeVols** et ensuite appeler la méthode **chargerFichierTexte** pour faire l'ouverture et la lecture du fichier (Cie\_Air\_Relax.txt) situé dans **LEA** de façon à remplir **listeVols** d'objets Vol et de compter les vols. Ensuite, il ferme le fichier. Ce sera la seule fois que le fichier texte sera lu.

La méthode d'instance **insérerVol** permet de saisir le numéro du nouveau vol à l'aide de `JOptionPane.showInputDialog` sous la forme :



Elle fera appel à la **rechercherVol** (une méthode privée de la classe Compagnie) qui reçoit en paramètre le numéro du vol, car pour pouvoir ajouter un nouveau vol, le numéro du vol ne doit pas déjà exister (prévoir un message d'erreur approprié si ce n'est pas le cas). Si le numéro est accepté, saisir alors la destination, la date de départ (`JOptionPane.showInputDialog`), le numéro de l'avion et mettre à 0 le total des réservations et insérer le vol au bon endroit dans le tableau.

- La méthode d'instance **retirerVol** permet de saisir le numéro du vol à l'aide `JOptionPane.showInputDialog` sous la forme :

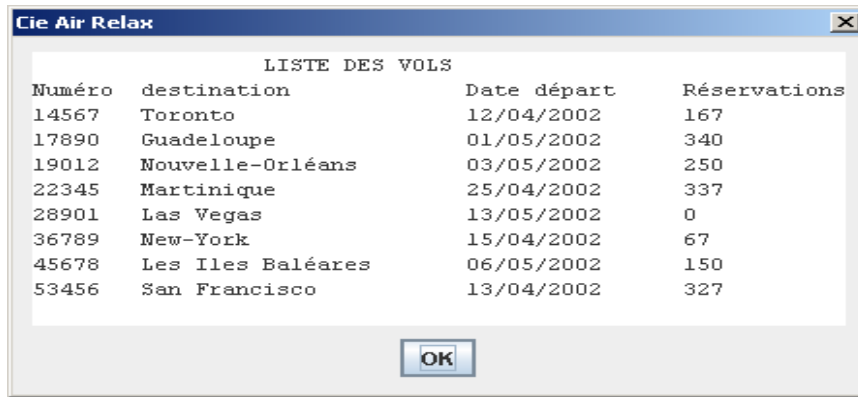


Elle fera appel à **rechercherVol** car pour pouvoir retirer un vol, le numéro du vol doit être existant (prévoir un message d'erreur approprié si ce n'est pas le cas). Si le numéro existe, à l'aide d'une seule fenêtre `JOptionPane.showInputDialog`, faire afficher sa destination, sa date et le total des réservations et demander la confirmation du retrait : « **Désirez-vous vraiment retirer ce vol (O/N) ?** » Si oui, retirer le vol de **listeVols** et ne pas oublier de diminuer le nombre de vols actifs.

- La méthode d'instance **modifierDate** permet de saisir le numéro du vol dont on veut modifier la date, à l'aide de `JOptionPane.showInputDialog` (la barre de titre contiendra le titre **MODIFICATION DE LA DATE DE DÉPART**). Elle fera appel à **rechercherVol**. Si le numéro n'existe pas prévoir le message d'erreur approprié, sinon à l'aide d'une seule fenêtre `JOptionPane.showInputDialog` faire afficher sa destination, sa date de départ actuelle et demander l'entrée de la nouvelle date (dans la forme JJ/MM/AAAA).
- La méthode d'instance **reserverVol** permet de saisir le numéro du vol que le client désire réserver, à l'aide de `JOptionPane.showInputDialog` (la barre de titre contiendra le titre **RÉSERVATION D'UN VOL**). Elle fera appel à **rechercherVol**. Si le numéro n'existe pas prévoir le message d'erreur approprié, sinon s'il reste de la place dans l'avion, à l'aide d'une seule fenêtre `JOptionPane.showInputDialog` faire afficher sa destination, sa date de départ, le nombre de places restantes dans l'avion et demander l'entrée du nombre de places que le client désire

réserver et faites la réservation. S'il n'y a plus de place, faire afficher à l'aide de `JOptionPane.showMessageDialog` le message adéquat.

- La méthode d'instance **listerVols** qui fait afficher la liste des vols existants à l'aide de `JTextArea` [fonte courier, de style `Font.PLAIN` et de taille 12] à transmettre à `JOptionPane.showMessageDialog`, sous la forme (la barre de titre contiendra le nom de la compagnie):



Numéro	destination	Date départ	Réservations
14567	Toronto	12/04/2002	167
17890	Guadeloupe	01/05/2002	340
19012	Nouvelle-Orléans	03/05/2002	250
22345	Martinique	25/04/2002	337
28901	Las Vegas	13/05/2002	0
36789	New-York	15/04/2002	67
45678	Les Iles Baléares	06/05/2002	150
53456	San Francisco	13/04/2002	327

- La méthode d'instance **sauvegarderFicherObjets** qui ouvre le fichier de la compagnie en écriture et enregistre le `ArrayList listeVols` dans le fichier **Cie\_Air\_Relax.obj**.
5. La classe **GestionCompagnie** qui contient le « main » dont l'objectif est de créer un objet de la classe **Compagnie** dont le nom doit être fourni par l'utilisateur en utilisant une fenêtre `JOptionPane.showInputDialog` (vous devez taper **Cie Air Relax**). Par la suite, le menu doit s'afficher tant et aussi longtemps que l'utilisateur n'aura pas fait le choix 0. Lorsqu'il fait le choix 0, on doit appeler la méthode **sauvegarderFicherObjets**.
  6. Dans une méthode **chargerVols** vous allez tester si le fichier **CieAirRelax.obj** existe. Si tel est le cas vous allez lire ce fichier et affecter le résultat à le `ArrayList listeVols` sinon vous allez lire le fichier **CieAirRelax.txt**. Ceci arrivera seulement la première fois, après le fichier d'objets sera créé lorsque vous choisirez l'option 0 du menu.
  7. Vous devez définir dans la classe **Vol** la méthode **CompareTo** qui sera utilisée par trier votre liste de vols.

<https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html>

8. Vous devez utiliser **`Collections.binarySearch()`** de Java pour faire la recherche d'un vol.

<https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html>

**NOTA** : Ce travail peut être fait par des équipes de maximum trois étudiants.

**Déposer votre travail compressé dans LEA.**

## Critères de correction

-50% si rien est fonctionnel et le code contient des erreurs

**TOUTE** consigne non respectée dans cet énoncé vous fera perdre 10% de la note attribuée à celui-ci.

Qualité du code et utilisation de la POO	15%
Fonctionnalités du menu, il y en a 5 x 12% chaque	60%
Utilisation et gestion du menu, validation, indentation du code, commentaires	10%
Ergonomie (présentation des résultats et écran de saisie de données)	10%
Qualité de copies écran du rapport de tests	5%

### REMISE :

Faites des copies écran de : **vosre menu**, résultat de **l'option1**, **option 2**, **option 3**, **option 4** et **option 5**. Mettre ces titres dans votre rapport et après les copies écran. Vous devez faire aussi des copies écran des valeurs saisies. Copiez ces images dans un document Word nommé « **labo1Tests** ». Ces tests doivent suivre l'ordre du menu. Indiquez dans ce même document toute question qui ne fonctionne pas et donner une idée du provient le problème. **Je regarderai le code pour pouvoir vous donner quelques points au lieu de zéro. Si pas de copie écran -50% de la question. Si une question ne fonctionne pas et vous n'avez pas indiqué dans votre rapport alors vous aurez zéro pour la question.**

**Déposez « labo1Tests » dans le dossier « documents » en format PDF absolument.**