

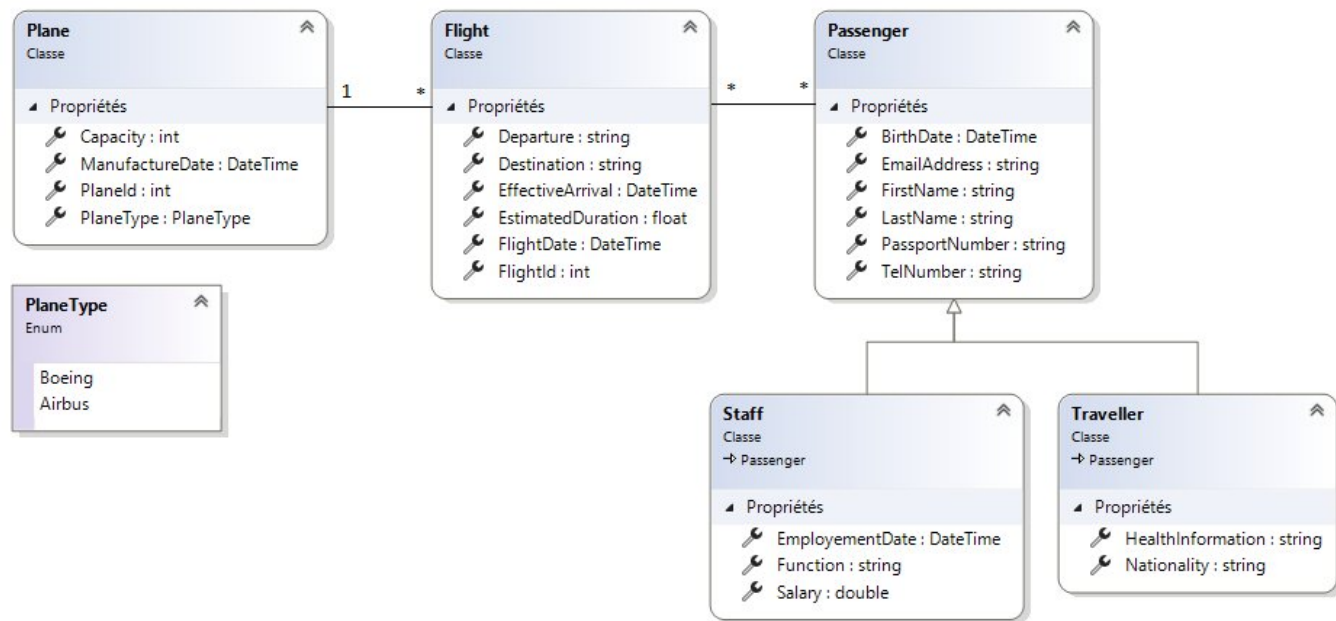
Chapitre 2 : Langage C#

Partie 1

SOMMAIRE


I.	Création des entités.....	2
II.	Création d'un service.....	3

On se propose de réaliser une application de gestion des activités d'un aéroport, définie par le diagramme de classes ci-dessous



I. Création des entités

1. En utilisant *VISUAL STUDIO 2022*, créer un nouveau projet nommé **AM.ApplicationCore** en tenant compte de ceci :
 - Le projet est de type bibliothèques de classes.
 - La solution est nommée **AirportManagement**.
 - Le framework cible est .NET 6.0 (le même pour tous les projets).
 - Supprimer la classe créée par défaut.
 - Aide : Suivre les étapes mentionnées au niveau du support du cours du chapitre 1 (Initiation à VS 2022).
2. Dans ce projet créer un dossier nommé **Domain** et créer dedans le type **Plane** en le rendant publique.
3. Créer la propriété auto-implémentée **Capacity**.
4. Dans le même dossier, créer le type **PlaneType**.
5. Ajouter les autres propriétés de l'entité **Plane**.
6. Faire le même travail pour les types **Flight** et **Passenger**.
7. Au niveau du projet, désactiver l'option appropriée pour ne plus afficher les messages de ce type (l'option se nomme *Nullable*) :

 CS8618 Le propriété 'Departure' non-nullable doit contenir une valeur non-null lors de la fermeture du constructeur. Envisagez de déclarer le propriété comme nullable.

8. Au niveau du type **Plane**, créer le constructeur par défaut et un autre qui permet l'affectation des propriétés *PlaneType*, *Capacity* et *ManufactureDate*. Les deux doivent être publiques.
9. Compiler le projet et vérifier que vous n'avez pas d'erreurs ni d'avertissements.

II. Création d'un service

10. Créer un nouveau dossier nommé **Interfaces** et y ajouter une interface publique nommée **IBasicFlightService**.
11. Sans implémenter un corps, ajouter à cette interface la méthode **ShowFlights(string filterType, string filterValue)** qui affiche les vols en fonction du type de filtre et sa valeur. Le filtre se fait pour les propriétés **Destination**, **FlightDate** et **FlightId**. Si le filtre est inconnu, la méthode doit générer une exception de type **ArgumentException** avec le message « Unknown filter ».
12. Créer un nouveau dossier nommé **Services** et y ajouter une classe publique nommée **BasicFlightService** qui contient un constructeur acceptant un paramètre de type **ICollection** nommé *source*. Faire le nécessaire pour stocker ce paramètre.
13. Au niveau de cette classe, implémenter l'interface créée précédemment (la description est déjà décrite précédemment). Le paramètre déjà stocké sera utilisé comme source du filtre. Aide :
 - Le parcours de la source se fait via le mot clé **foreach**.
 - L'affichage se fait en utilisant la méthode **Console.WriteLine**.
 - La conversion vers un type se fait en utilisant la méthode **Parse** du type destination.
14. Essayer d'optimiser le code pour les filtres **FlightDate** et **FlightId**.