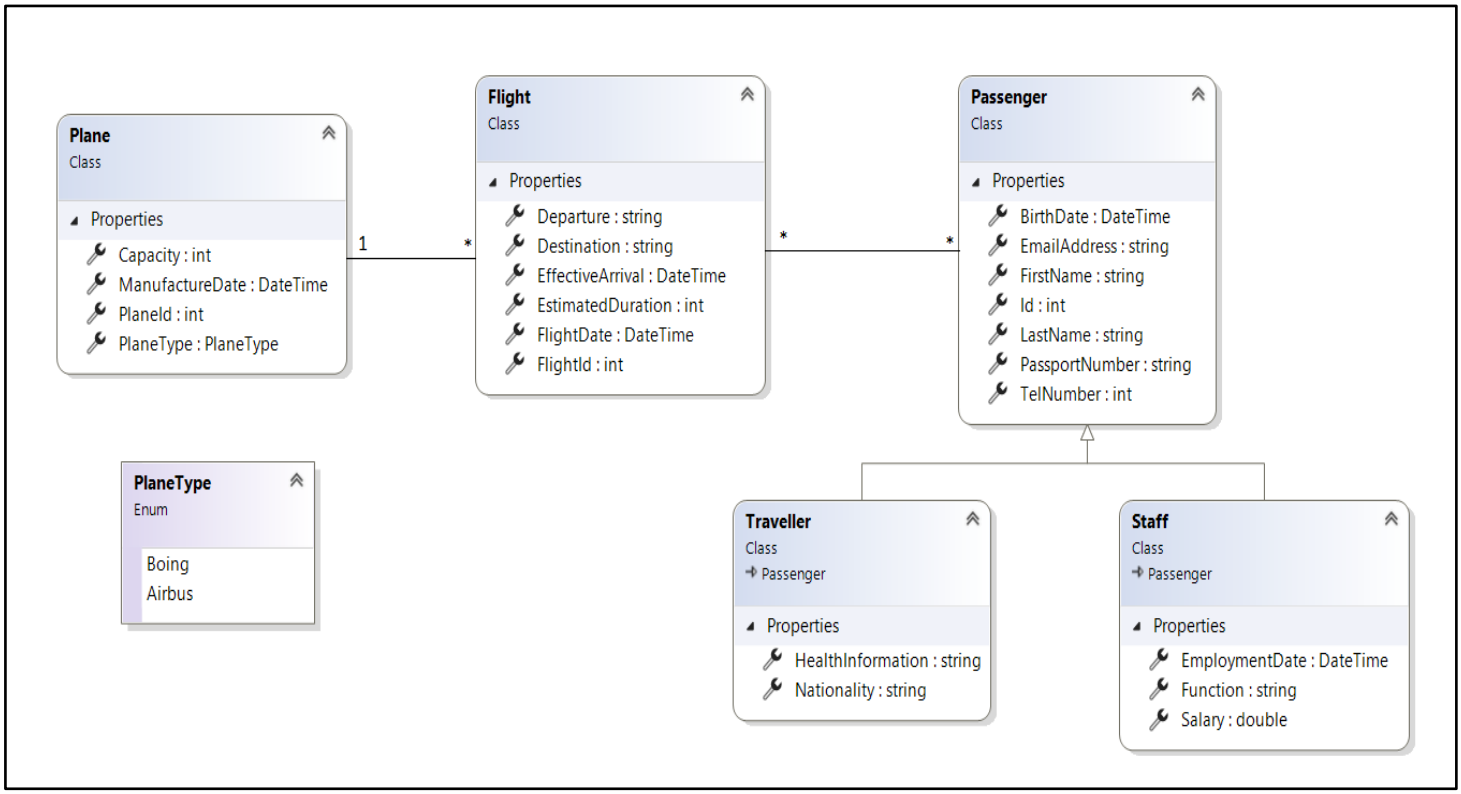
TP AirportManagement

Partie 1: Les principes de l'orienté objet

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



I Implémentation de la Couche de Données

1. Créer une Solution nommée **AirportManagement** qui contient les deux projets suivants
   AM.UI.Console : Projet de type application Console (.NET 8.0)
   AM.ApplicationCore : Projet de type Bibliothèque de classe (.NET 8.0)
   Notez bien que le projet **Console** doit référencer le projet **ApplicationCore**.
2. Sous le projet **AM.ApplicationCore**, créer le dossier **Domain** et y implémenter les différentes classes du diagramme de classes ci-dessus.
3. Représenter l'héritage entre la classe **Passenger** et les deux classes **Staff** et **Traveller**.
4. Implémenter les propriétés qui représentent les différents attributs et leurs accesseurs.
5. Représenter les relations au biais des objets de navigation.

a. Par exemple, la relation 1-* entre **Plane** et **Flight** sera représentée par les objets de navigation suivants ;
    i. – Une propriété de type **ICollection<Flight>** dans la classe **Plane**
    ii. – Une propriété de type **Plane** dans la classe **Flight**

6. Réimplémenter la méthode ToString() pour toutes les classes.

II Instanciation des objets

7. Créer un objet non initialisé de type **Plane** en utilisant le constructeur non paramétré de la classe, puis initialiser ses attributs à travers leurs propriétés.
8. Créer le constructeur suivant pour la classe **Plane**
    i. **public Plane (PlaneType pt, int capacity, DateTime date)**

Puis créer un autre avion en utilisant ce constructeur.

9. Supprimer le constructeur crée précédemment et instancier un autre avion en utilisant les initialiseurs d'objet. Que remarquez vous ?

III Le Polymorphisme

10. Polymorphisme par Signature

Dans l'entité **Passenger**, créer les trois méthodes **bool CheckProfile(…)** suivantes :

a. Une méthode pour vérifier le profile en utilisant deux paramètres: nom du passager et prénom du passager.
b. Une méthode pour vérifier le profile en utilisant trois paramètres: nom du passager, prénom du passager et email du passager.
c. Une méthode pour remplacer à la fois les deux méthodes précédentes.

11. Polymorphysme par héritage

a- Dans la classe **Passenger**, implémenter la méthode **PassengerType** qui affiche « **I am a passenger»**

b- Dans la classe **Staff** et **Traveller**, réimplémenter la même méthode pour qu'elle affiche respectivement « **I am a passenger I am a Staff Member**» et « **I am a passenger I am a traveller»**.

Utiliser l'implémentation de la méthode de la classe mère dans les 2 méthodes précédentes.

c- Tester la méthode **PassengerType** dans le projet console pour 3 instances de types **Passenger, Staff et Traveller**.