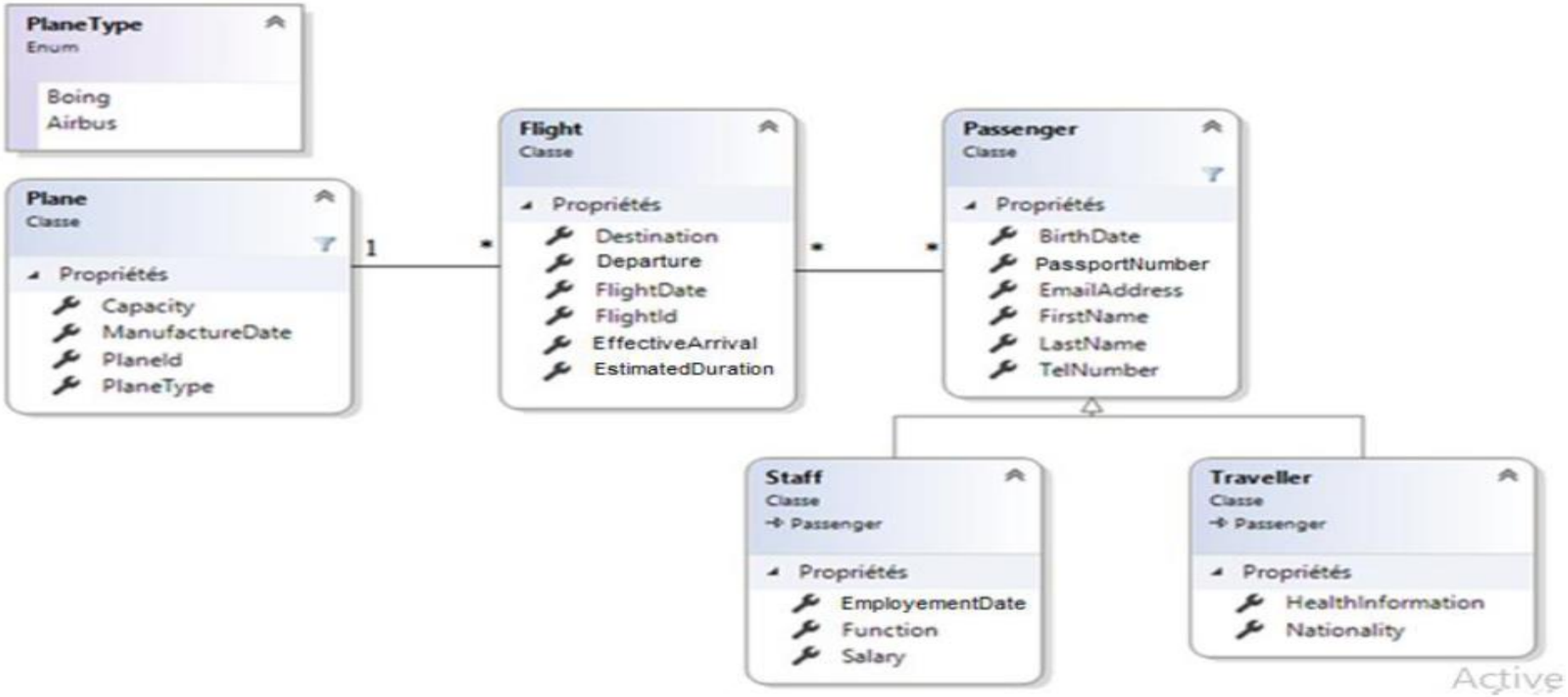
TP AIRPORT MANAGEMENT

Tarek Ayari

MISE EN PLACE DE LA SOLUTION:

Notre objectif principal est de développer une application pour la gestion des activités d'un aéroport, utilisant le framework .NET 6. Soit le diagramme de classe suivant:



PARTIE 1 : DIAGRAMME DE CLASSES

- Créer une solution vide nommé “AirportManagement” et y ajouter les projets suivants

1. AM.UI.Console : Projet de type application Console (.NET 6.0)
2. AM.ApplicationCore : Projet de type Bibliothèque de classe (.NET 6.0)
3. Sous le projet AM.ApplicationCore, créer le dossier « Domain » et y implémenter les différentes classes du diagramme de classes ci-dessus
4. Ajouter la référence de projet Core dans le projet Console

PARTIE 1 : DIAGRAMME DE CLASSES

4. Représenter l'héritage entre la classe Passenger et les deux classes Staff et Traveller
5. Implémenter les propriétés qui représentent les différents attributs.
6. 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**

PARTIE 1 : DIAGRAMME DE CLASSES

7. Implémenter le diagramme de classes suivant :
    a. Créer les relations entre les différentes classes et décorer les propriétés de navigation par le mot clé Virtual
    i. One to many entre les entités « Plane » et « Flight»
    ii. Many to many entre les entités « Flight» et « Passenger»

8. Réimplémenter la méthode ToString() pour toutes les classes qui retourne les propriétés de chaque classe

PARTIE 2 : INSTANCIATION DES OBJETS

6. 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.
7. Créer le constructeur suivant pour la classe Plane.
   - public Plane (PlaneType pt, int capacity, DateTime date)
   - Puis créer un autre avion en utilisant ce constructeur.
   - Supprimer le constructeur crée précédemment et instancier un autre avion en utilisant les initialiseurs d'objet.
   - → C'est plus simple et intuitif d'utiliser les initialiseurs d'objets.

PARTIE 3 : POLYMORPHISME DE SURCHARGE

8. Dans l'entité **Passenger,** créer les deux 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. Tester les deux méthodes dans le program Main en créant un objet Passenger.

PARTIE 4 : POLYMORPHISME D'HERITAGE

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

- «I am a passenger » si l'objet est de type «Passenger».
- «I'm a traveller"» si l'objet est de type «Traveller».
- «I'm a staff member» si l'objet est de type «Staff».

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

PARTIE 4 : INTERFACE ET SERVICE

10. Sous le projet « AM.ApplicationCore », créer les deux dossiers **Interfaces** et **Services**.
11. Créer l'interface **IServiceFlight** dans le dossier Interfaces et la classe **ServiceFlight** dans le dossier Services.
12. Dans la classe ServiceFlight, créer la propriété public List<Flight> Flights {get ; set ;} et l'initialiser à une liste vide.

- Nous formulerons les requêtes de ce TP en se basant sur cette liste comme source de données.

- N'oubliez pas de mettre à chaque fois la signature de chaque méthode dans l'interface

PARTIE 5 : PRÉPARATION DES DONNÉES DE TEST

13. Ajouter dans le projet « AM.Domain » la classe statique **TestData** qui contient les données statiques de test du tableau suivant.

- **Planes:**

Planes		
PlaneType	**Capacity**	**ManufactureDate**
Boing	150	03/02/2015
Airbus	250	11/11/2020

PARTIE 5 : PRÉPARATION DES DONNÉES DE TEST

- **Staff:**

Staff					
FirstName	**LastName**	**EmailAddress**	**BirthDate**	**EmploymentDate**	**Salary**
captain	captain	Captain.captain@gmail.com	01/01/1965	01/01/1999	99999
hostess1	hostess1	hostess1.hostess1@gmail.com	01/01/1995	01/01/2020	999
hostess2	hostess2	hostess2.hostess2@gmail.com	01/01/1996	01/01/2020	999

PARTIE 5 : PRÉPARATION DES DONNÉES DE TEST

- **Travellers:**

Travellers					
FirstName	**LastName**	**EmailAddress**	**BirthDate**	**HealthInformation**	**Nationality**
Traveller1	Traveller1	Traveller1. Traveller1@gmail.com	01/01/1980	No troubles	American
Traveller2	Traveller2	Traveller2. Traveller2@gmail.com	01/01/1981	Some troubles	French
Traveller3	Traveller3	Traveller3. Traveller3@gmail.com	01/01/1982	No troubles	Tunisian
Traveller4	Traveller4	Traveller4. Traveller4@gmail.com	01/01/1983	Some troubles	American
Traveller5	Traveller5	Traveller5. Traveller5@gmail.com	01/01/1984	Some troubles	Spanish

PARTIE 5 : PRÉPARATION DES DONNÉES DE TEST

- **Flights:**

Flights					
FlightDate	**Destination**	**EffectiveArrival**	**Plane**	**EstimatedDuration**	**Passengers**
01/01/2022 15:10:10	Paris	01/01/2022 17:10:10	Airbus	110	All created travellers
01/02/2022 21:10:10	Paris	01/02/2022 23:10:10	Boing	105	
01/03/2022 5:10:10	Paris	01/03/2022 6:40:10	Boing	100	
01/04/2022 6:10:10	Madrid	01/04/2022 8:10:10	Boing	130	
01/05/2022 17:10:10	Madrid	01/05/2022 18:50:10	Boing	105	
01/06/2022 20:10:10	Lisbonne	01/06/2022 22:30:10	Airbus	200	

PARTIE 5 : PRÉPARATION DES DONNÉES DE TEST

- Dans la même classe, créer la liste statique **List<Flight> listFlights** et l'initialiser avec tous les vols crées précédemment.
- Dans le projet console, créer une instance de la classe ServiceFlights puis affecter listFlights à la propriété Flights de cette classe service.
- Tester toutes les méthodes qui suivent en se basant sur ces données de test.

PARTIE 5.1 : ITÉRATIONS / STRUCTURES CONDITIONNELLES

11. En utilisant la boucle For, implémenter la méthode **GetFlightDates (string destination)** dans la classe **ServiceFlight** qui retourne la liste des dates de vols d'une destination passée en paramètre

- Tester la méthode GetFlightDates () avec plusieurs paramètres
- Reformuler la fonction en utilisant foreach, Re-Tester la méthode GetFlightDates ()

PARTIE 5.2 : ITÉRATIONS / STRUCTURES CONDITIONNELLES

11. Implémenter la méthode **GetFlights(string filterType, string filterValue)** qui affiche les vols en fonction de type de filtre et sa valeur. Le type de filtre représente un attribut de la classe Flight.

- Par exemple GetFlights("Destination", "Paris") permettra d'afficher les vols dont la valeur de Destination est égale Paris.
- Tester la méthode GetFlights () avec plusieurs paramètres

LINQ

Présentation
licrosoft PowerPoi

PARTIE 6 : LINQ

12. Implémenter les méthodes suivantes et les tester à chaque fois dans le projet Console:

- Reformuler la méthode **GetFlightDates(string destination)** en utilisant une requête LINQ.
- **ShowFlightDetails(Plane plane):** Afficher les dates et les destinations des vols d'un avion passé en paramètre
- **ProgrammedFlightNumber(DateTime startDate):** Retourner le nombre de vols programmés pour une semaine (7jours) à partir d'une date donnée

PARTIE 6 : LINQ

12. Implémenter les méthodes suivantes et les tester à chaque fois dans le projet Console:

- **DurationAverage(string destination):** Retourner la moyenne de durée estimées des vols d'une destination donnée
- **OrderedDurationFlights():** Retourner les Vols ordonnés par EstimatedDuration du plus long au plus court

PARTIE 6 : LINQ

- **SeniorTravellers(Flight flight):** Retourner les 3 passagers, de type traveller, les plus âgés d'un vol
- **DestinationGroupedFlights():** Retourner les vols groupés par destination et les afficher sous ce format

Destination Paris
Décollage : 03/05/2022 12 : 10 :00
Décollage : 05/05/2022 23 : 00 :00
Décollage : 10/05/2022 21 : 15 :00
Destination Madrid
Décollage : 01/05/2022 10 : 10 :00
Décollage : 02/05/2022 13 : 10 :00

PARTIE 7 : LES MÉTHODES D'EXTENSION

17. Créer la classe **PassengerExtension** qui étend la classe Passenger et et qui contient la méthode d'extension suivante.

- **UpperFullName(Passenger p) :** qui met en majuscule la première lettre du nom et du prénom d'un passager.
- Tester la méthode d extension dans l application Console