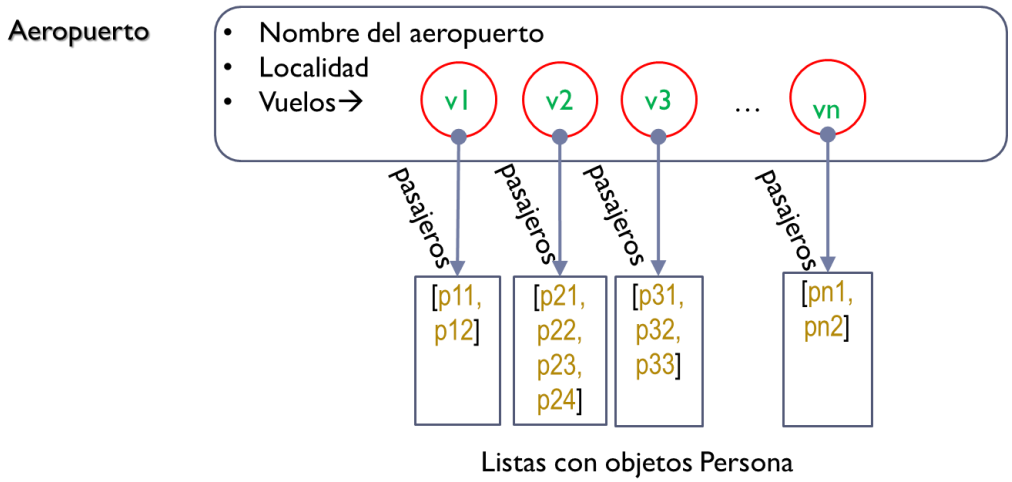
EL TIPO PERSONA, VUELO y AEROPUERTO

Consideraciones Iniciales

- Este es un enunciado que se irá incrementando a medida que avancemos en la teoría del 2º cuatrimestre del curso.
- Es **MUY IMPORTANTE** llevarlo al día, porque **numerosas cuestiones** avanzarán a partir de lo realizado en las anteriores, bien porque se ha resuelto en clase o se han dejado como tarea para casa.

Esquema gráfico de la estructura de datos del proyecto



Listas con objetos Persona

:

EJERCICIOS:

1. Importe el proyecto: T03_Aeropuerto. (vea las instrucciones que se facilitan en la sesión de hoy)
2. En fp.aeropuerto implemente un tipo ***Persona*** con las siguientes características:

PERSONA

Propiedades:

- String **dni**. Consultable.
- String **nombre**. Consultable.
- String **apellidos**. Consultable.
- LocalDate fechaNacimiento. Consultable

Métodos Contructores:

- Un constructor a partir de cada uno de los atributos (constructor canónico).
- Un constructor con apellidos y nombre en el que el dni quede con la cadena vacía (“”) y la fecha de nacimiento el 1 de enero de 1995.

Restricciones:

- Ningún atributo puede ser nulo.
- La fecha de nacimiento no puede ser mayor que la actual

El criterio de igualdad:

- Dos personas son iguales si tienen el mismo dni.

El criterio de ordenación:

- Por dni.

La representación como cadena:

- Todos los atributos básicos con el nombre del tipo y separados por comas.

3. En fp.aeropuerto,test ya tiene implementado una clase ***TestPersona01*** con algunos datos por cumplimentar. Si lo hace bien, al ejecutar el test debe salir lo siguiente:

```
Test constructor-1 con todos los parámetros
        Persona[dni=12345678A, nombre=Manolito, apellidos=Gafotas, fechaNacimiento=2000-01-15]
        Construido correctamente!!

Test constructor-1 con la fecha errónea
        Ha habido un error al construir la persona
        java.lang.IllegalArgumentException: fp.aeropuerto.Persona.<init>: la fecha no puede ser mayor que la
actual

Test constructor-1 con todos los parámetros nulos
        Ha habido un error al construir la persona
        java.lang.IllegalArgumentException: fp.aeropuerto.Persona.<init>: el parámetro 1 es nulo

Test constructor-2 con apellidos y nombre
        Persona[dni=, nombre=Manolito, apellidos=Gafotas, fechaNacimiento=1995-01-01]
        Construido correctamente!!

Test constructor-2 con parámetros nulos
        Ha habido un error al construir la persona
        java.lang.IllegalArgumentException: fp.aeropuerto.Persona.<init>: el parámetro 2 es nulo
```

4. Implemente en fp.aeropuerto el Tipo *Vuelo* con los siguientes requisitos:

VUELO

Propiedades:

- String **código**. Consultable. Indica el código del vuelo.
- String **destino**. Consultable. Ciudad de destino del vuelo
- LocalDateTime **fechaHoraSalida**. Consultable. Fecha y hora de salida del vuelo.
- Duration **duración**. Consultable.
- Double **velocidad**. Consultable.
- Double **precio**. Consultable. Precio del billete.
- Integer **númeroPlazas**. Consultable. Número de plazas del vuelo.
- Boolean **conEscalas**. Consultable. Indica si el vuelo tiene, o no, escalas.
- List<Persona> **pasajeros**. Consultable. Lista con los pasajeros del vuelo.
- Integer **númeroPasajeros**. Consultable.
- Boolean **vueloCompleto**. Consultable. Indica si el vuelo está, o no, completo.
- Double **porcentajeOcupación**. Consultable.
- LocalDateTime **fechaHoraLLegada**. Consultable. Fecha y hora de llegada a destino.
- Compañía **compañía**. Se calcula a partir de los 3 primeros caracteres del código. Podrá tomar uno de los cuatro siguientes valores: IBE, LUF, RYA, VGL.

Métodos Constructores:

- Un constructor a partir de cada uno de los atributos (constructor canónico).

Restricciones:

- Ningún atributo puede ser nulo.

- El código debe tener 7 caracteres, los 3 primeros alfabéticos y los 3 último numéricos. El del centro serán un guion (-). Si hay error el mensaje es: "*El formato del código debe ser aaa-nnn*"
- El precio es mayor o igual que cero.
- El número de plazas debe ser mayor o igual que cero.
- La velocidad debe ser mayor o igual que cero.
- La duración es mayor que cero.
- El número de pasajeros es menor o igual que el número de plazas.

El criterio de igualdad

- Dos vuelos son iguales si tienen el mismo código, destino y la misma fecha de salida (sin la hora)

El criterio de ordenación:

- Por el código y desempatan por el destino y, en su caso, también por la fecha de salida (sin la hora)

La representación como cadena:

- El código, el destino, fecha y hora de salida, fecha y hora de llegada, número de plazas y número de pasajeros.

5. En fp.aeropuerto,test ya tiene implementado una clase ***TestVuelo01*** que si la ejecuta debe obtener lo siguiente:

```
Test constructor con todos los parámetros
      Vuelo [código=IBE-001, destino=Valencia, fechaHoraSalida=2023-08-01T09:00,
fechaHoraLlegada=2023-08-01T10:55 , Nro.pasajeros=3]
      Construido correctamente!!

Test constructor con algún parámetro nulo
      Ha habido un error al construir el vuelo
      java.lang.NullPointerException: Cannot invoke
"java.util.Collection.isEmpty()" because "coll" is null

Test constructor con el código malo
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: El formato
del código debe ser aaa-nnn

Test constructor con la duración negativa
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: La duración
debe ser >0

Test constructor con la velocidad negativa
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: La
velocidad debe ser >=0

Test constructor con el precio negativo
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: El precio
del billete debe ser >=0

Test constructor con las plazas negativas
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: El número
de plazas debe ser >0
```

```
Test constructor con las plazas menores que el número de pasajeros
      Ha habido un error al construir el vuelo
      java.lang.IllegalArgumentException: fp.aeropuerto.Vuelo.<init>: El número
de pasajeros debe ser <= número de  plazas
```

6. Practicamos métodos de Collectión y sus subinterfaces

Crear una clase ***TestPersona02*** para realizar operaciones de Collection.

1. Para ello primero creamos 5 personas con los datos:

```
("FRANCISCO MIGUEL","AARAB ORTIZ","12346678A",LocalDate.of(2005,1,15));
("ALBERTO","AGUILAR RALSTON","13457789B",LocalDate.of(2005,2,15));
("ALVARO","AGUILAR RALSTON","14568900C",LocalDate.of(2005,3,15));
("ADRIÁN","ALARCON MARTIN","15680011D",LocalDate.of(2005,4,15));
("PABLO","ALBA CONRADI","16791122E",LocalDate.of(2005,5,15));
```

2. Crea 3 listas de pasajeros vacías y añadir respectivamente las siguientes personas:

   pasaj1 con las 3 primeras personas
   pasaj2 con las 2 últimas personas
   pasaj3 con las 3 personas que ocupan las posiciones impares

3. Visualiza el número de elementos de las 3 listas.
4. Añade a ***pasaj1*** los elementos de ***pasaj3***
5. Visualiza el número de elementos de ***pasaj1***.
6. Crea un conjunto de pasajeros ***conj1***. y añádele los elementos de ***pasaj1***.
7. Visualiza el número de elementos de ***conj1***.
8. Añade a ***pasaj2*** los elementos de ***conj1*** desde ***la posición 1***.
9. Visualiza el número de elementos de ***pasaj2***.
10. Comprobar si el ***primer pasajero*** está en ***pasaj2***
11. Eliminar el ***primer pasajero*** de ***pasaj2***.
12. Visualiza el número de elementos de ***pasaj2***
13. Eliminar el ***pasajero p4*** de ***pasaj2***.
14. Visualiza el número de elementos de ***pasaj2***