DAM/DAW Programación

Tarea “paqXpress”

Con objeto de crear una aplicación para la gestión de los envíos de una pequeña empresa de logística y mensajería, se decide aplicar un patrón de diseño STATE para la gestión de los de paquetes y los diferentes estados por los que pasan

MODELO

Las entidades principales del modelo del sistema serán:

PAQUETE

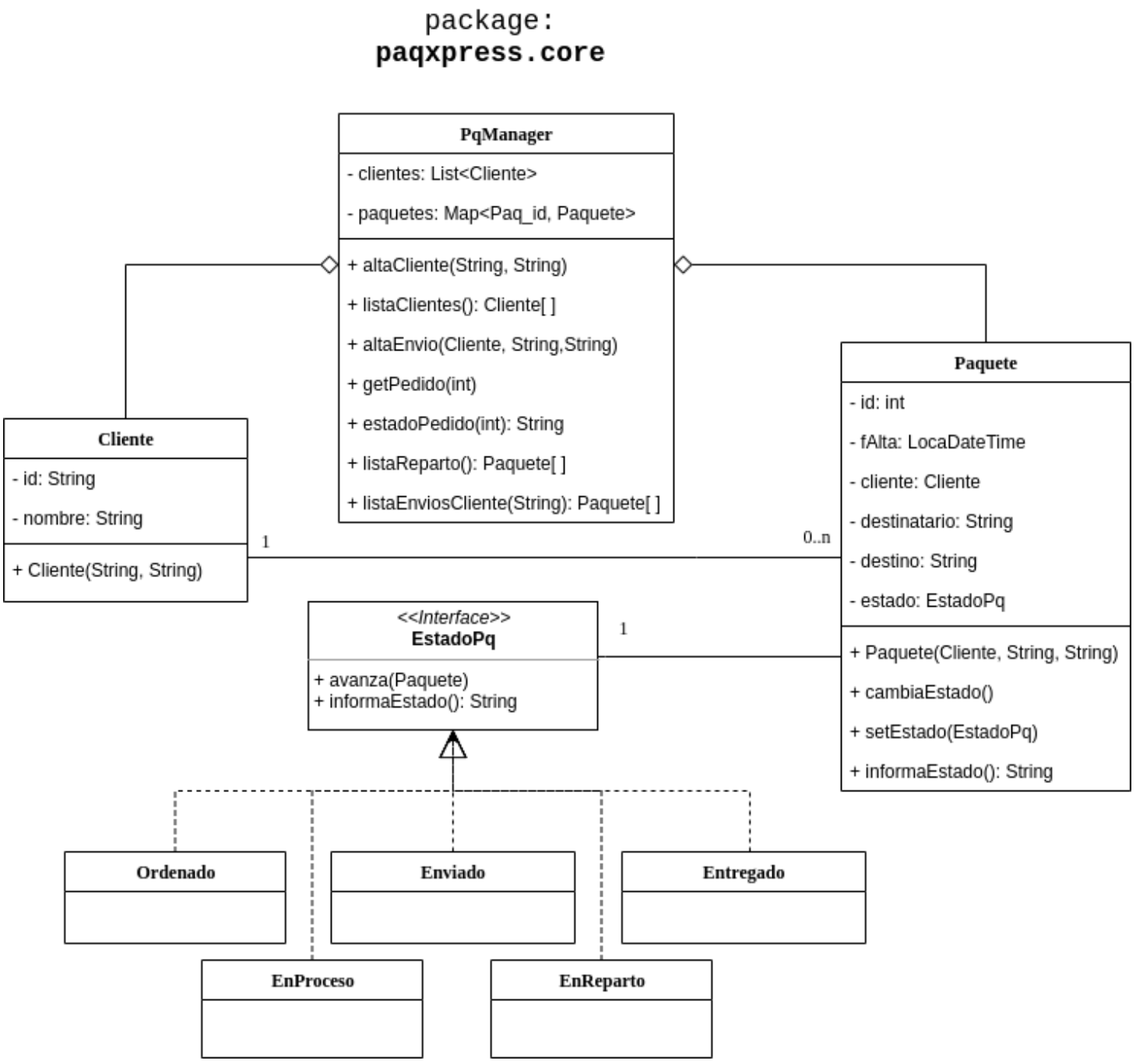
- Cada paquete estará identificado por un **id de envío** numérico secuencial que se genera de forma **automática** al introducirlo en el sistema. Dicho **id** comenzará en 1000 siendo 1001 el id para el primero de los envíos
- Cada paquete estará asociado a un cliente de la empresa de logística. Además, se registrará la fecha/hora de entrada en el sistema, nombre del destinatario y la dirección de envío del mismo
- Un paquete pasará por los siguientes estados (y en ese único **orden**):

Estado	Descripción
Ordenado	Alta en el sistema del nuevo envío
En Proceso	El paquete ha sido recogido y se encuentra en proceso de envío
Enviado	El paquete ha sido enviado
En Reparto	El paquete se encuentra en reparto
Entregado	El paquete ha sido entregado en su destino

CLIENTE

- Por cada cliente se registrará su CIF/NIF (que servirá a efectos de identificación del mismo) y su nombre

Aplicando el patrón indicado, se ha diseñado el siguiente esquema base de clases que deberá ser implementado:
(puedes añadir métodos ***getter*** y ***toString*** a las clases **Paquete** y **Cliente**)



CLASES

- Todos los atributos serán privados

➢ **PqManager**

- Es la clase encargada de la gestión de los clientes y envíos
- Atributos:
  - **clientes**: **lista** dinámica en la que se almacenarán los diferentes clientes de la empresa a medida que se vayan dando de alta en el sistema
  - **paquetes**: **mapa** que nos permitirá almacenar cada envío y localizarlo por su identificador
- Métodos:
  - Se proporcionarán métodos para realizar las siguietes tareas:
    - alta de nuevo cliente
      - params: id(NIF ó CIF) y nombre del cliente
    - lista de clientes
      - devolverá un **array** con los clientes
    - alta envío
      - params: cliente que solicita el envío, nombre del destinatario y dirección de destino del mismo
    - get pedido
      - params: **id** del paquete
    - estado del paquete:
      - params: **id** del paquete del que se quiere consultar su estado
      - devolverá una cadena con la descripción del estado actual
    - lista de envíos en situación de reparto
      - devolverá un **array** con los paquetes en reparto
    - lista de envíos por cliente
      - params: id del cliente
      - devolverá un **array** de los paquetes enviados por el cliente indicado

- **EstadoPq**

    - Interfaz/Clases para los diferentes estados
    - Métodos:
        - **avanza(Paquete)**: cambia el estado del paquete indicado al siguiente en la secuencia de estados. Cada estado "sabe" cuál es el siguiente estado de la secuencia
        - **informaEstado( )**: devuelve la descripción del estado

- **Paquete**

    - Clase para almacenar la información de los envíos
    - Atributos:
        - **id**: valor numérico secuencial generado por la clase de forma **automática** en el momento de creación del envío. Dicha numeración se inicia en el valor 1000, siendo 1001 el id del primer envío.
        - **fAlta:** fecha/hora de alta del envío (LocaDateTime)
        - **cliente**: cliente que solicita el envío
        - **estado**: estado actual del envío
        - **destinatario**: nombre del destinatior del paquete
        - **destino**: dirección de de destino del paquete
    - Métodos:
        - **cambiaEstado( ):** solicita a su **estado actual** que cambie al siguiente en la secuencia de estados. Ten en cuenta que la clase Paquete no tiene información de qué estados existen (este patrón de diseño "desacopla" los paquetes de los estados por los que pasan). El esquema de las llamadas sería:

        - **setEstado(EstadoPq):** actuliza el estado actual del paquete al nuevo estado indicado
        - **informaEstado( )**: devuelve la descripción del estado

Aplicación: ConsoleApp

Como paso previo al desarrollo de una aplicación Web para la gestión de los envíos, se nos plantea el desarrollo de un prototipo basado en la consola para probar el correcto funcionamiento de nuestra estructura de clases.

Para ello, dentro del paquete **paqxpress.test** se creará una pequeña aplicación que constará de una única clase: **consoleApp** que importará y usará las clases descritas anteriormente

Dicha clase mostrará el siguiente menú de opciones:

```
########################################
#                                      #
#           p q X p r e s s            #
#                                      #
# --->--->--->--->--->--->--->--->---  #
#                                      #
# [1] - Alta cliente                   #
# [2] - Lista clientes                 #
# [3] - Alta nuevo envío               #
# [4] - Consultar estado envío         #
# [5] - Cambiar estado envío           #
# [6] - Listar envíos por cliente      #
# [7] - Listar envíos en reparto       #
# [X] - Salir                          #
#                                      #
########################################
Opción >
```

Este menú se ejecutará en un bucle infinito de forma que se puedan realizar sucesivas operaciones (la aplicación no finalizará hasta que se haya seleccionado la opción correspondiente)

La aplicación mantendrá dos ficheros, **clientes.dat** y **paquetes.dat**, en los que se almacenarán serializadas las colecciones correspondientes. Estos archivos se cargarán al iniciar la aplicación y se actualizarán en el momento de finalizar la misma.

La aplicación deberá proporcionar las siguientes funcionalidades:

- **Alta cliente:**
  - solicitará la entrada de los datos necesarios para el alta de un nuevo cliente

- **Lista clientes:**
  - mostrará los datos de los clientes dados de alta en el sistema

- **Nuevo envío:**
  - solicitará el identificador del cliente y, tras comprobar que existe en los clientes dados de alta, solicitará los datos restantes del nuevo envío y lo registrará en el sistema. La fecha/hora será la correspondiente al momento del registro del nuevo envío

- **Consultar envío:**
  - Tras solicitar un id de envío y comprobar que es válido, mostrará los datos del envío: fecha, cliente, destinatario, destino y situación actual en la que se encuentra

- **Cambiar estado envío:**
  - Tras solicitar un id de envío y comprobar que es válido, promoverá el estado de dicho envío al siguiente estado en la secuencia.

- **Listar envíos por cliente:**
  - mostrará los datos (fecha/hora, destino, situación) de los envíos del cliente indicado

- **Listar envíos en reparto:**
  - mostrará los datos (cliente y destino) de los envíos que se encuentren actualmente en situación de reparto