Instituto de Tecnología ORT
Carrera de Analista de Sistemas
Materia: Taller de Herramientas de Programación

EXAMEN FINAL

Leé con cuidado el enunciado y por lo menos dos veces para resolver lo pedido. Pensá bien la estrategia de resolución antes de comenzar el desarrollo de lo que te solicitan. El objetivo de este examen es **evaluar la correcta aplicación de los conceptos y técnicas** vistos hasta el momento:

- Correcta implementación de constructores.
- Modularización reutilizable y mantenible con uso de métodos con correcta parametrización y correcto encapsulamiento, publicando *setters* y *getters* sólo cuando corresponda.
- Manejo de clases, enumerados y colecciones.
- Importación y exportación de proyectos Java desde Eclipse.

Enunciado

AppDePagos es una billetera electrónica cuyo objetivo es que los usuarios tengan en un solo lugar todos los servicios a pagar ordenados para su pago a tiempo. Como muchas billeteras, esta funciona con el método de depósito previo: el usuario deposita dinero en la billetera y con ese dinero va pagando los servicios a los que se registró (adicionalmente tiene otros beneficios que hoy no nos interesan).

La aplicación guarda todos los datos del **usuario** registrado que usa la aplicación (en el futuro permitirá llevar la billetera de más de un usuario). De este se sabe lo siguiente:

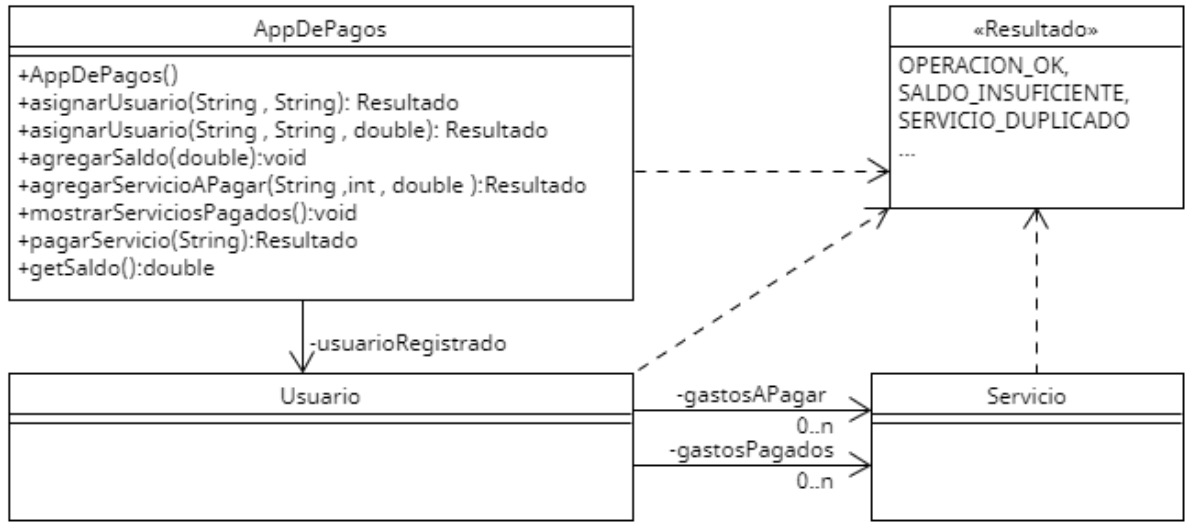
- CUIL-CUIT.
- Saldo disponible para pagos.
- Una colección de servicios a pagar. En ella solo es posible tener una factura de cada tipo de servicio cuando está pendiente de pago (no hay repetidos por servicio).
- Una colección de servicios pagados (acá sí puede haber servicios repetidos).

Se puede realizar una operación por servicio, la que puede ser registrar un servicio para un pago posterior o realizar un pago. También puede guardar dinero en la billetera, acto necesario para pagar un servicio. Los resultados de cada **operación** pueden verse parcialmente en el diseño UML (deben completarse).

De cada servicio se sabe:

- Nombre
- Número de comprobante.
- Importe.

El diseño básico e incompleto es el siguiente:



Nosotros formamos parte del equipo de desarrollo de la aplicación y nos piden que escribamos Java usando Eclipse las clases necesarias para cumplir con la siguiente funcionalidad:

- Un método para asignar el usuario que usa la aplicación: debe crearlo siempre y cuando no haya otro usuario registrado previamente. En caso de que la aplicación ya tenga un usuario registrado el debe devolver el mensaje de error correspondiente. Este método tendrá dos versiones: una agregando también un saldo (que debe ser mayor que cero) y otra solamente con el cuil y el nombre del usuario (en este caso el saldo debe quedar en cero).
- Un método con el que se pueda agregar un servicio a pagar. Al agregar un servicio debe chequearse que el mismo no exista en la lista de impagos. En caso de que exista debe devolver SERVICIO DUPLICADO (sino OK).
- Un método para que el usuario pueda pagar servicios, chequeando que tanto el usuario como el servicio existan y que alcance el saldo. En caso de fallar debe devolver el código de error correspondiente; en caso de realizar correctamente el pago además del OK el servicio debe quedar entre los **pagados**.
- Un método para agregar saldo de dinero al usuario (si la aplicación tiene un usuario asignado). El monto a agregar no puede ser menor a cero. Chequear lo que haga falta y devolver el OK o el error correspondiente.
- Un método para mostrar los servicios pagados por el usuario. Debe mostrar, al final, el monto total de lo abonado.

Completá el programa provisto (el main no debe ser modificado). Al completar el programa deberá verse algo como esto:

```
Registrando al usuario con cuil 12-12345678-8
Nombre: Albert Essen, Saldo: 10000.0
OPERACION_OK

Registrando al usuario con cuil 12-87654321-8
Nombre: Albin Otinto, Saldo: 50000.0
USUARIO_YA_EXISTENTE

Agregando saldo (3000.0)
OPERACION_OK

Agregando saldo (-3000.0)
ERROR_EN_MONTO

Agregar AYSA
OPERACION_OK

Agregar FLOW
OPERACION_OK

Agregar FLOW
SERVICIO_DUPLICADO

Agregar AYSA
SERVICIO_DUPLICADO

Agregar AYSA
SERVICIO_DUPLICADO

Pagar FLOW
OPERACION_OK
```

```
Pagar FLOW
SERVICIO_SIN_PENDIENTES

Pagar PATENTE
SERVICIO_SIN_PENDIENTES

Servicio pagado: FLOW
Total pagado: 6254.8
```