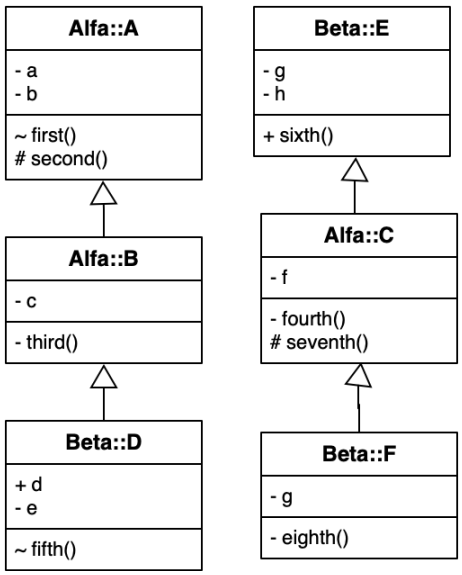
IIC 2143 Ingeniería de Software
Semestre 2021-2
Profesores M. Peralta y J. Navón

Puedes entregar tu prueba en un solo documento pdf con todas las respuestas o una carpeta comprimida con un pdf para cada pregunta. En cualquier caso, el archivo que subas a la plataforma debe tener por nombre: "I2_PrimerApellido_SegundoApellido". Por ejemplo: I2_Perez_Yoma.pdf o I2_Quiroz_Ema.rar

Código de honor: El trabajo debe ser **estrictamente individual**. Puedes consultar tus apuntes, si lo quieres, pero no puedes pedir ayuda ni físicamente ni en forma remota (correo, mensajería). Tampoco puedes trabajar en grupo o en pareja. Cualquier violación a este código que se detecte no solo será sancionada con la nota mínima sino con el sumario respectivo.

Problema 1 (24 puntos)

A continuación se muestra el diagrama UML de una pieza de software. En ellas se han indicado métodos y atributos con sus respectivas visibilidades. En cada caso indique si la afirmación es verdadera (V) o falsa (F) y agregue la razón (una sola línea para justificar)



a) En el método third de la clase B se puede acceder al atributo c
b) En el método fifth de la clase D se puede acceder al método second
c) En el método fifth de la clase D se puede acceder al método third
d) En el método fourth de la clase C se puede acceder al método third
e) En el método fourth de la clase C se puede acceder al método first
f) En el método third de la clase B se puede acceder al método sixth
g) En el método fourth de la clase C se puede acceder al atributo d
h) En el método sixth de la clase E se puede acceder al método fifth
i) En el método eight de la clase F se puede acceder al método seventh
j) En el método eight de la clase F se puede acceder al método fourth
k) En el método sixth de la clase E se puede acceder al atributo d
l) En el método eight de la clase F se puede acceder al atributo g

Problema 2 (20 puntos)

La discografía de un artista incluye varios álbums. Cada álbum incluye varias canciones. Queremos modelar las clases Discografía, Album y Song de la forma en que se indica a continuación

- una discografía tiene un único atributo artista y un método tiempo_total que entrega el total de minutos correspondiente a toda la discografía del artista
- un álbum tiene un único atributo nombre y un método tiempo_total que entrega el total de minutos correspondiente a las canciones del álbum
- una canción tiene dos atributos: título y duración y un método duración que entrega la duración en minutos de la canción
- queremos sacar partido del patrón composite de modo que el método duración entregue el total de minutos de una canción, álbum o de toda la discografía del artista dependiendo del objeto de que se trate

a) (10 pts) Dibuje un diagrama de clases UML que muestre la solución

b) (10 pts)Escriba el código Ruby que lo implemente

Problema 3 (26 puntos)

Se tiene el siguiente código Ruby que permite calcular el salario neto en distintos países. *amount* es el sueldo bruto y lo que cambia es el tratamiento de los impuestos en cada país:

```
def net_salary(amount, country)
    taxes = case country
        when "Ukraine"
            (amount * 0.05) + 313
        when "U.S."
            (amount * 0.2) + 100
        when "Poland" amount * 0.3
        else
            0
        end
    amount - taxes
end
```

Por ejemplo:

```
net_salary(1000, "Poland") # => 700.0
net_salary(1000, "Ukraine") # => 637.0
```

a) (16 pts)Efectue un proceso de *refactoring* de este código utilizando el patrón de diseño "Estrategia" (Strategy) para que el código sea mas extensible. Escriba un segmento de código que ilustre el funcionamiento para calcular el sueldo en Polonia y en Ukrania como lo hacía el antiguo código.

b) (10 pts) Haga diagramas de clases y de secuencia que ilustren la forma en que funciona el código de ejemplo de la parte a).

Problema 4 (30 puntos)

La compañía CormoránAzul es una aerolínea con una flota de 20 aviones de distintos tamaños. Cada avión se identifica con un modelo y patente, y tiene una configuración distinta de asientos de acuerdo a su capacidad. Los asientos se identifican con una letra (string de A a la Z) y un número de fila. Además un asiento tiene una categoría que puede ser prioritaria, de emergencia o regular. Las personas que tienen asientos prioritarios pueden abordar antes y encontrarán sus asientos posicionados delante del ala del avión, donde las turbulencias se sienten en menor medida. Los asientos de emergencia se encuentran sobre el ala, tienen más espacio para las piernas pero no se pueden reclinar. Finalmente los regulares son todos los asientos restantes de un avión que no cumplen las condiciones anteriores.

La compañía ofrece distintos vuelos. Un vuelo queda definido por un número, un aeropuerto de salida, un aeropuerto de llegada y una fecha. El vuelo está asociado a un avión de la flota, por tanto la cantidad de asientos disponibles dependerá del avión asignado. Existen dos tipos de pasajes que pueden comprar los pasajeros, con asiento asignado o no. Los precios varían de acuerdo a ello. Si eligen la opción de asiento asignado podrán elegir qué asiento tomar al momento de comprar el pasaje. El pasaje es único por pasajero y debe guardarse un e-mail de contacto del pasajero para posibles notificaciones.

Un vuelo puede sobre venderse cuando hay más pasajes vendidos que asientos disponibles. Cuando esto ocurre se debe realizar el proceso de reasignación de avión, el cual consiste en lo siguiente:

- Un coordinador de vuelos ve una lista con todos los vuelos existentes. Si hay un vuelo sobrevendido éste aparece en rojo.
- El coordinador tiene la opción de reasignar un avión al vuelo sobrevendido. Para ello hace click sobre el vuelo sobrevendido y es redirigido a una nueva lista de aviones disponibles con mayor capacidad.
- Una vez que lo asigna, existe un proceso automático que revisa si el asiento que un pasajero eligió, digamos el K5, sigue siendo de la misma categoría en el nuevo avión. Si no es así, se le debe enviar un e-mail explicando la situación y solicitando su aprobación al cambio.

Se te ha pedido ayuda como desarrollador de la parte backend de este proceso. Decides hacerlo en Rails, y sabes que deben existir los modelos airplane (avión), seat (asiento), flight (vuelo), y ticket (pasaje).

a) (10 pts) Dibuja un diagrama de clases en que quede claro las relaciones entre los modelos existentes. Si tu solución requiere otra clase o modelo distinto puedes agregarlo. No puedes omitir los mencionados.
b) (10 pts) Escribe los modelos y migraciones con las asociaciones necesarias.
c) (10 pts) Escribe un método por cada uno de los siguientes puntos utilizando query Interface de Rails y operando sobre su resultado.
    i) Escribe un método que retorne una lista de objetos con información de vuelos. Para cada vuelo debe aparecer el número, aeropuerto de salida y de llegada, fecha, modelo de avión, cantidad de asientos y si está sobrevendido o no.
    ii) Escribe un método que retorne una lista de aviones disponibles con capacidad mayor a la de un vuelo en particular flight. (Pueden asumir que si un avión está asignado a un vuelo ya no se encuentra disponible independiente de la fecha y hora)
    iii) Lista de e-mails de pasajeros que habría que notificar que su asiento no es de la misma categoría que antes al reasignar el vuelo V1 con el avión A2. El método debe recibir el vuelo V1 y el nuevo avión A2 como parámetros. (No es necesario usar esos nombres como parámetros)

En esta pregunta se te está pidiendo que escribas clases y métodos, NO que implementes una aplicación Rails. Con lo que sabes de Rails puedes escribirlo sin tener que programarlo para probarlo. El formato de la respuesta debe ser en PDF, ya sea código escrito o imágenes de código, pero debe ser legible y claro. No se aceptarán otros formatos como .rb o .zip, o cualquier otro formato que no sea .pdf.

Buena Suerte !