IIC 2143 Ingeniería de Software
Interrogación 2 - Semestre 2 /2019
Secciones 01 y 02

Responda cada pregunta en hoja separada
Entregue una hoja con su nombre para cada pregunta aunque sea en blanco
Tiempo: 2:15
Recuerden que están bajo el código de honor

Pregunta 1: (Modelos de Dominio)

Una empresa de arriendo de habitaciones de un complejo residencial se encuentra desarrollando una aplicación web para ofrecer sus servicios a través del mercado digital. La aplicación debe permitir que administradores, moderadores y arrendatarios puedan ingresar para realizar distintos tipos de operaciones. Todos deben definir un email, el cuál debe ser único, y un nombre para poder identificarlos. Solo los arrendatarios pueden arrendar habitaciones.

Todo arriendo debe obligatoriamente pasar a través de un contrato de arriendo, el cual define una fecha de inicio y una fecha de término. Dado un contrato de arriendo vigente, la aplicación debe permitir a los arrendatarios efectuar los respectivos pagos de sus arriendos en cuotas mensuales, las cuáles vencen en una determinada fecha. El valor de la cuota depende del monto acordado al momento de firmar el contrato, la habitación y servicios adicionales que el cliente pudo haber solicitado. Estos incluyen servicios como muebles, pago de cuentas (agua, luz, basura) incluido, y servicio de limpieza de la habitación. Cada uno de estos servicios debe contar con un valor para poder ser considerado por el cliente. La habitación por su lado debe contar con un número y una superficie. Cada pago efectivamente realizado por un cliente debe figurar en el sistema; sin embargo, nótese que el monto cancelado por el cliente no necesariamente corresponde al monto de la cuota ya que en caso de atraso deben agregarse intereses.

Todo cliente cuenta adicionalmente con una hoja de vida en la cual figura el registro de todas las transgresiones cometidas por él, incluyendo atrasos en los pagos, violación de las reglas de la comunidad y delitos comunes de los que se tenga conocimiento. Esto le permite a la agencia tomar estos hechos en consideración en futuras solicitudes de arriendo efectuados por el cliente.

Por último, con el fin de promover la nueva aplicación, se está pensando en implementar un sistema de referencias en el que arrendatarios puedan invitar a amigos para así conseguir descuentos para ambos. Como está concebida la funcionalidad, si un arrendatario se registra al sistema luego de ser referido por un amigo, la aplicación debe generarle un voucher de descuento a ambos por un determinado monto para ser usado cuando estimen conveniente.

Dibuje un modelo de dominio (usando la notación del diagrama de clases de UML) que ilustre las principales clases y relaciones (asociaciones, agregaciones, composiciones y generalizaciones) entre clases que se desprenden del enunciado. En las clases, incluya todos los atributos que se desprendan del enunciado; para las relaciones, especifique etiquetas y multiplicidades; añada generalizaciones donde sea pertinente. Especificar visibilidad y tipos de los atributos, además de navegabilidad de las relaciones es opcional.

Explicite todos los supuestos que consideró en la elaboración de su modelo.

Pregunta 2: (Patrones de Diseño)

El sistema computacional de una entidad bancaria chilena permite realizar transferencias bancarias solo entre cuentas corrientes originadas en Chile. Para ello, la aplicación en cuestión define la clase "CuentaCorriente", la cual define el atributo "pesos", el cual se puede leer y escribir libremente. Toda transferencia se hace a través de una instancia de la clase "Banco", la cual define el método *transferir(fuente, destino, monto)*, donde *fuente* y *destino* son instancias de "CuentaCorriente" y *monto* es un entero que representa el monto de la transferencia en pesos. Este método permite traspasar el monto especificado como parámetro desde la cuenta *fuente* hacia la cuenta *destino*.

Usted ha sido contratado para desarrollar la versión 2.0 de este sistema computacional, el cual entre sus múltiples funcionalidades debe permitir transferir dinero desde y hacia cuentas extranjeras que operen con dólares americanos. Los sistemas con los que usted debe interoperar son afortunadamente muy similares al sistema chileno, pues estos definen una clase "ForeignAccount" que también define un único atributo que se puede leer y escribir libremente, solo que en este caso el atributo se llama "dollars" y representa el monto de la cuenta en dólares americanos.

Durante la fase de análisis de su solución, su equipo de desarrollo se da cuenta de un detalle que puede ser importante: los montos en pesos de "CuentaCorriente" se guardan como enteros, mientras que los montos en dólares de "ForeignAccount" se guardan como números decimales con dos dígitos de exactitud. Sin embargo, al transformar pesos a dólares, suele ocurrir que los montos resultantes definen 3 dígitos decimales o más. Después de consultar con el cliente, se optó por programar el sistema para que cada vez que ocurra un caso similar durante una transferencia, el monto a setear en la cuenta extranjera se debe truncar a solo 2 dígitos decimales, y la diferencia, por pequeña que sea, se debe depositar en una cuenta del banco.

Dado lo anterior, utilizando el lenguaje Ruby, usted debe:

- Escribir la clase "Banco".
- Escribir un adaptador que permita que el método *transferir(fuente, destino, monto)* funcione con instancias de "ForeignAccount" y que incorpore la funcionalidad de redondeo aquí descrita.
- Escribir un breve script que emule una operación de transferencia de 50.000 pesos desde una "CuentaCorriente" con 100.000 pesos hacia una "ForeignAccount" con 1.000 dólares, a una tasa de cambio de 715,53 pesos/dólar.

Puede asumir que "Banco" guarda una referencia a la cuenta corriente del banco y que las clases "CuentaCorriente" y "ForeignAccount" vienen dadas (no tiene que implementarlas). Se le recuerda que Ruby define el método *floor(n),* que permite truncar números decimales a 'n' cifras significativas.

Pregunta 3: (Diagramas UML)

En una aplicación web usada por un sitio que se dedica a vender libros on-line se han identificado las siguientes clases de dominio:

- Producto: detalles del libro (nombre, isbn, precio)
- Cliente: detalles del cliente (rut, nombre, email, descuento)
- OrdenDeCompra: detalles de la compra (lo que estaba en el carrito)
- ItemOrden: detalle de un ítem de la orden (producto, cantidad)

Una de las responsabilidades de la OrdenDeCompra es calcular el total de la orden. Por ejemplo:

Orden No: 234123
Cliente: Jorge Campos Rut: 8.456.902-5

Cantidad	Producto	Precio	Total
2	23134	15.990	31.980
1	21341	20.000	20.000
Subtotal			51.980
Descuento Cliente 10%			5.198
Total Orden			46.782

Los precios de los productos se sacan directamente del modelo del producto. El descuento a aplicar está asociado al cliente.

a) Dibuje el diagrama UML de clases relativo al problema descrito. Incluya los atributos y los métodos que estime necesarios y que se puedan desprender del enunciado.

(3 puntos)

b) Dibuje un diagrama UML de secuencia que muestre detalladamente lo que ocurre cuando una instancia de la clase OrdenDeCompra como la del ejemplo de arriba (dos líneas de detalle) recibe un mensaje de calcular su valor.

(3 puntos)

Pregunta 4: (Conceptos Generales de Ingeniería de Software)

Indique si la afirmación es verdadera o falsa. Justifique en no más de 5 líneas las afirmaciones falsas.

(0.6 puntos por cada afirmación)

a) En el desarrollo de su proyecto semestral usted ha debido realizar planificación a nivel de iteración y de producto.

b) Si a pesar de sus estimaciones, al completar el segundo sprint usted evalúa que hay escasas opciones de poder entregar a tiempo el curso de acción de mejor pronóstico es negociar con el cliente para conseguir un poco mas de plazo.

c) Se ha estimado el tiempo de desarrollo en 6 meses. Dado que, de acuerdo con la ley de Parkinson, igual se terminará usando todo el tiempo disponible es mala idea planificar para 8 o 9 meses.

d) Los puntos de relato (story points) corresponden a una métrica de esfuerzo de desarrollo.

e) Mientras más líneas de código tenga el software resultante, mayor será la productividad de los desarrolladores.

f) Se ha estimado un esfuerzo de desarrollo de 24 meses hombre para un proyecto. Si contamos con 6 programadores el desarrollo tomará alrededor de 4 meses.

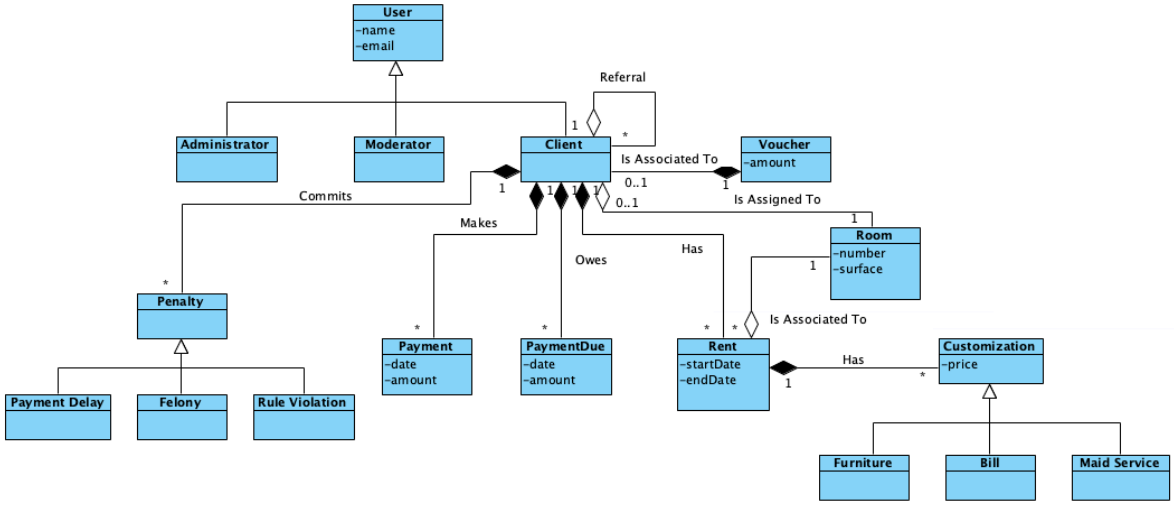
g) Un problema de la métrica de puntos de función es que es dependiente del lenguaje de programación que se utilice.

h) El equipo de desarrollo ha hecho una estimación de tiempos tanto optimista como pesimista. La optimista arrojó 10 semanas y la pesimista 20 semanas. Lo mas probable es que finalmente el proyecto tome alrededor de 15 semanas.

i) El MRF (minimum releasable features) determina el corte correspondiente al "will have" en el stack del backlog.

j) La mejor forma de dar estimaciones del tiempo de desarrollo es en semanas.

Pauta Pregunta 1:



Nota de Pauta:

Desglose de puntaje:

- Herencia de distintos usuarios con superclase común: 1 punto
- Herencia de castigos con superclase común y composición con cliente: 1 punto
- Se identifica separación de conceptos de “contrato de arriendo”, “deuda” y “pago efectivo”: 1 punto
- Herencia de customizaciones de un contrato de arriendo con superclase común y composición con contrato de arriendo: 1 punto
- Lógica de vouchers y referencias. En este ejemplo, cada cliente puede o no tener un voucher y se especifica una relación de agregación consigo mismo para denotar posibles referencias: 1 punto
- Relación entre contrato de arriendo, habitación y cliente. En estricto rigor, esto se modelaría mejor con una relación n-aria entre todas estas clases, pero el ejemplo en la solución propuesta es aceptable también: 1 punto

Pauta Pregunta 2:

```ruby
class Banco
  attr_reader :cuenta_banco

  def initialize
    @cuenta_banco = CuentaCorriente.new(0)
  end

  def transferir(fuente, destino, monto)
    if fuente.pesos >= monto
      fuente.pesos -= monto
      destino.pesos += monto
    end
  end

  def captar_redondeo(monto)
    @cuenta_banco.pesos += monto
  end

end

class AccountAdapter
  attr_reader :exchange_rate, :account

  def initialize(account, exchange_rate)
    @exchange_rate = exchange_rate
    @listeners = []
    @account = account
  end

  def pesos
    @account.dollars * @exchange_rate
  end

  def pesos=(value)
    dollars = value / @exchange_rate
    rounded_value = dollars.floor(2)
    difference = dollars - rounded_value
    @listeners.each do |listener|
      listener.captar_redondeo(difference * @exchange_rate)
    end
    @account.dollars = rounded_value
  end

  def add_listener(listener)
    @listeners.append(listener)
  end
end

bank = Banco.new
account1 = CuentaCorriente.new(100_000)
account2 = ForeignAccount.new(1000)
adapter = AccountAdapter.new(account2, 715.53)
adapter.add_listener(bank)

bank.transferir(account1, adapter, 50000)
puts("Account 1: #{account1.pesos} - Account 2: #{account2.dollars} - Bank:
#{bank.cuenta_banco.pesos}" )
```

Nota de Pauta:

Considerar el siguiente desglose de puntaje:

- Clase “Banco”: 1.5 puntos
- Adaptador: 3 puntos
- Script: 1.5 puntos

Durante la evaluación se precisó que el tema de redondeo debía considerarse para transferencias desde y hacia cuentas extranjeras. Sin embargo, como toda transferencia se hace a través del método *transferencia* para el cual los montos son siempre en pesos, solo el seteo de los montos en cuentas extranjeras es relevante.

En esta solución propuesta, la captación de las diferencias entre valores truncados y reales por la cuenta corriente del banco se materializa a través del uso del patrón Observer. Sin embargo, no es obligatorio el uso de este patrón para conseguir este efecto dado que no se especifica en el enunciado. Soluciones alternativas que cumplan el objetivo pero no usen Observer también son válidas. No obstante, se explicita claramente el uso del patrón Adapter para la otra parte del problema. Evaluar con 0 puntos el ejercicio si no se percibe el uso del patrón en ninguna parte (solo en caso de completa ausencia del patrón; si está incorrectamente implementado, aplicar descuentos que se estime conveniente).

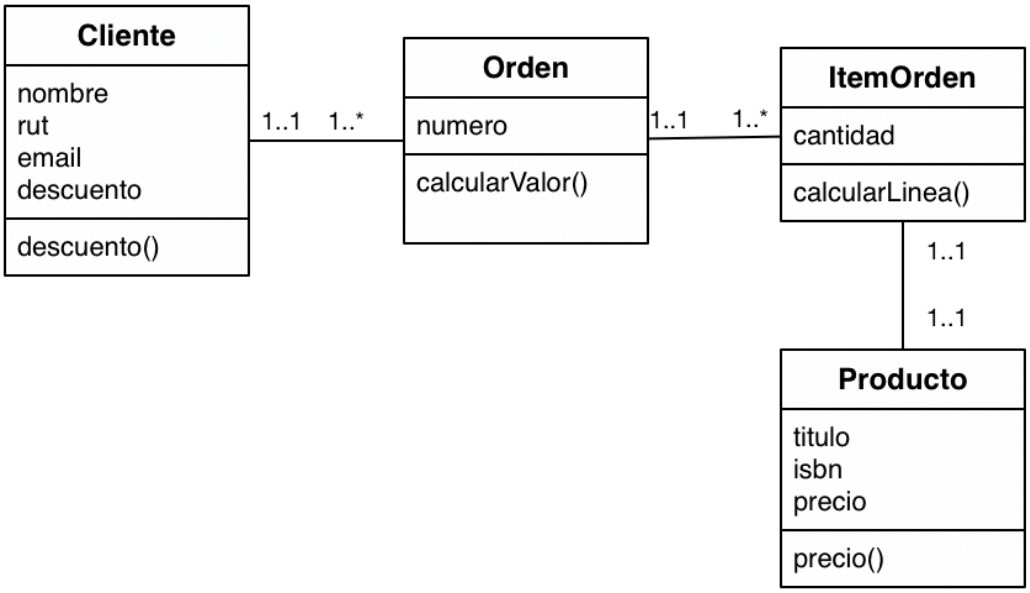
La validación de que el monto sea inferior a cantidad disponible en cuenta de origen en método para transferencia no es requerida en la solución del alumno.

Ser criterioso con errores de sintaxis. En general, uno o dos errores menores (como escribir mal el nombre de una función de ruby, u omitir un tag end) pueden omitirse sin necesidad de aplicar descuento. Solo si son muy numerosos o se trata de errores flagrantes descontar puntaje. Si por otro lado el alumno elabora su solución en Python o pseudo-código, evaluar todo el ejercicio con 0 puntos.

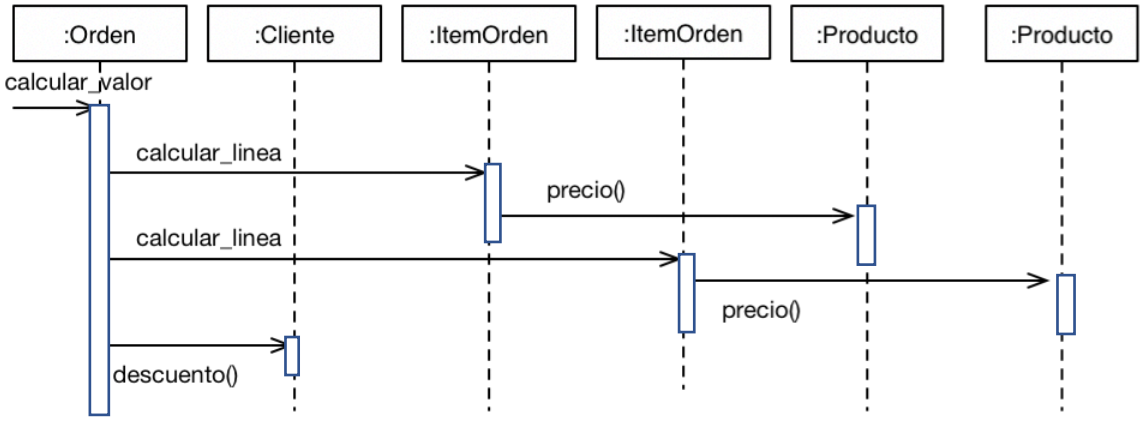
Pauta Pregunta 3:

a) Un posible diagrama es el siguiente. Podríamos haber incluido un rombo negro en la asociación de Orden con ItemOrden ya que se trata de una asociación de composición y un rombo sin rellenar entre ItemOrden y Producto ya que cada item hace referencia a un producto.

Los métodos esenciales son los que calculan el valor de la orden, el que calcula el monto de cada línea, el que entrega el precio de cada producto y el que entrega el porcentaje de descuento asociado al cliente.



b) El diagrama de secuencia muestra el caso en que hay dos líneas (ItemOrden) en la orden. Calcular_valor invoca calcular_linea para cada objeto ItemOrden de la orden, suma los resultados parciales y le resta el descuento (lo obtiene desde el cliente de la orden). Calcular_linea de ItemOrden debe obtener el precio desde el producto y multiplicarlo por la cantidad.



Nota de Pauta:

Se acepta para el diagrama de secuencia que el alumno use interaction frames para denotar la iteración sobre los ítems.

Pauta Pregunta 4

a) En el desarrollo de su proyecto semestral Ud ha debido realizar planificación a nivel de iteración y de producto.

   Falso - Se hace planificación a nivel de iteración y del release pero no del producto.

b) Si a pesar de sus estimaciones, al completar el segundo sprint Ud. evalúa que hay escasas opciones de poder entregar a tiempo el curso de acción de mejor pronóstico es negociar con el cliente para conseguir un poco mas de plazo.

   Falso - Por lo general tiene mas probabilidad de éxito una negociación por alcance (dejando algunos features fuera del release).

c) Se ha estimado el tiempo de desarrollo en 6 meses. Dado que, de acuerdo a la ley de Parkinson, igual se terminará usando todo el tiempo disponible es mala idea planificar para 8 o 9 meses.

   Falso - Dado que los costos de subestimar son mucho mayores que los del sobreestimar es mejor asumir el costo de algo de tiempo ocioso al riesgo de no poder cumplir.

d) Los puntos de relato (story points) corresponden a una métrica de esfuerzo de desarrollo.

   Falso - Es una métrica de tamaño al igual que los puntos de función o líneas de código.

e) Mientras más líneas de código tenga el software resultante, mayor será la productividad de los desarrolladores.

   Falso - En software se da el fenómeno de rendimientos decrecientes de modo que lo que ocurre es exactamente lo contrario.

f) Se ha estimado un esfuerzo de desarrollo de 24 meses hombre para un proyecto. Si contamos con 6 programadores el desarrollo tomará alrededor de 4 meses.

   Falso - En software los meses y los hombres no son intercambiables libremente.

g) Un problema de la métrica de puntos de función es que es dependiente del lenguaje de programación que se utilice.

   Falso - Justamente una de las ventajas respecto a las LOCs es que es independiente del lenguaje.

h) El equipo de desarrollo ha hecho una estimación de tiempos tanto optimista como pesimista. La optimista arrojó 10 semanas y la pesimista 20 semanas. Lo mas probable entonces es que finalmente el proyecto tome alrededor de 15 semanas.

Falso - Lo mas probable es que el proyecto se acerque mucho más a la estimación pesimista y no en el centro entre 10 y 20.

i) El MRF (minimum releasable features) determina el corte correspondiente al "will have" en el stack del backlog.

Falso - El MRF determina el punto de "must have". Es importante que este punto se ubique en la zona de "will have" o al menos fuera del "wont have".

j) La mejor forma de dar estimaciones del tiempo de desarrollo es en semanas.

Falso - Es mejor, cuando es posible, usar unidades mas grandes como meses o incluso trimestres ya que usar semanas transmite una idea engañosa de precisión