IIC 2143 Ingeniería de Software
Interrogación 2 - Semestre 1 /2018 - Sección 02: Prof. J. Navón

Pregunta 1 - 20 pts (Estimaciones)
Conteste en la forma más breve y precisa posible

a) Indique 3 métricas asociadas al tamaño del software e indique fortalezas y debilidades
b) Explique para qué sirve y cómo funciona el "planning poker" también llamado "scrum poker"
c) Explique los riesgos de sobreestimar vs subestimar el esfuerzo. ¿Cuál sería su recomendación a la luz de lo anterior ?
d) Que se entiende por el fenómeno de "deseconomía de escala" en el desarrollo de software y cuál sería la causa

Pregunta 2 - 20 pts (Patrones de Diseño)
Se quiere implementar una clase Podcast que mantiene una lista de episodios y un método add_episode que permite incorporar nuevos episodios. Normalmente un Podcast mantiene un cierto número de subscriptores que deben ser informados cada vez que se agrega un nuevo episodio. Queremos que funciona mas o menos así:

```
>> developer_tea = Podcast.new
>> s1 = Subscriber.new("Jaime")
>> s2 = Subscriber.new("Rodrigo")
>> developer_tea.add_subscriber(s1)
>> developer_tea.add_subscriber(s2)
>> developer_tea.add_episode (Episode.new("T1-E3"))

Jaime: there is a new episode !
Rodrigo: there is a new episode !

>> developer_tea.remove_subscriber(s1)
>> developer_tea.add_episode (Episode.new("T1-E4"))

Rodrigo: there is a new episode !
```

a) (15 pts) Identifique el patrón de diseño relevante y escriba el código Ruby necesario (Solo lo estrictamente necesario para que funcione como el ejemplo). Debe escribir
   - La clase Podcast con los métodos necesarios
   - La clase Subscriber con los métodos necesarios
   - La clase Episode

b) (5 pts) Dibuje un diagrama de secuencia que muestre el funcionamiento de toda la secuencia de mas arriba (desde creación del postcast, creación de subscriptores, agregación de un nuevo episodio, eliminación de subscriptor y agregado de otro episodio)

Pregunta 3 - 20 pts (Modelo de Dominio)

El CAI ha decidido construir una aplicación web para que los alumnos puedan consultar sobre preguntas de pruebas de semestres anteriores de algún ramo específico. Un curso en un año y semestre dado puede haber tenido más de una sección, cada una con distinto profesor. En esa ocasión cada prueba puede haber un número de interrogaciones (típicamente 3) y un examen final. Cada prueba tiene un número variable de preguntas cada una de las cuales tiene un puntaje asociado. La prueba completa también tiene un puntaje máximo asociado. Así por ejemplo, esta pregunta (I2 de IIC2143 primer semestre de 2017, sección 02 (Prof. Navón) vale 20 puntos y la prueba 100. Las preguntas pueden además ser etiquetadas por los alumnos de acuerdo a características (desarrollo, UML, difícil). No hay límite en el número de etiquetas que pueden asignarse a una pregunta.

a) (15 pts) Construya un modelo de dominio que capture en la forma más completa posible la situación de modo de poder comenzar el diseño de la aplicación.

b) (5 pts) Cómo modificaría el modelo si se quieren guardar también las pautas o soluciones de las pruebas

Pregunta 4 - 20 pts (Patrones de Diseño)

Un retailer como Amazon tiene un procedimiento bastante típico para el checkout una vez que se ha puesto el o los productos en el carrito: ingresar la dirección de envío, ingresar el método de pago, revisar los items y poner la orden. Abajo se muestra parte del código y a la derecha un ejemplo de interacción con él.

```
class CheckoutProcess
 attr_reader :order
  def initialize(order)
   @order = order
  end
  def set_shipping_address(address)
   order.shipping_address = address
  end
  def set_payment_method(payment_method)
   order.payment_method = payment_method
  end
  def review_order
   puts "Order contains: #{order.items}"
   puts "Shipping Address is #{order.shipping_address}"
   puts "Payment Method is #{order.payment_method}"
  end
  def place_order
   order.confirm
  end
end
```

```
>> order = Order.new("Head First Design Patterns")
>> checkout = CheckoutProcess.new(order)
>> checkout.set_shipping_address("9th St. Apt xxx, San Francisco, CA")
>> checkout.set_payment_method("Chase Credit Card")
>> checkout.review_order
Order contains: Head First Design Patterns
Shipping Address is 9th St. Apt xxx, San Francisco, CA
Payment Method is Chase Credit Card
>> checkout.place_order
Your order has been place.
Your item: Head First Design Patterns
    will be shipped to 9th St. Apt xxx, San Francisco, CA
Your Chase Credit Card will be charged for the order.
```

Se quiere implementar un proceso abreviado llamado 1-click que usa direcciones y métodos de pago defaults para el cliente (ver abajo). Identifique el patrón de diseño relevante que permite implementar este comportamiento y escriba el código Ruby para que funcione exactamente en la forma ilustrada.

```
>> default_address = "9th St. Apt xxx, San Francisco, CA"
>> default_payment = "Chase Credit Card"
>> order = Order.new("Head First Design Patterns")
>> one_click_checkout = OneClickCheckout.new(order, default_address, default_payment)
>> one_click_checkout.click
Your order has been place.
Your item: Head First Design Patterns
    will be shipped to 9th St. Apt xxx, San Francisco, CA
Your Chase Credit Card will be charged for the order.
```

Pregunta 5 - 20 pts (Diagrama de Estados UML)

Típicamente, un sitio Web como por ejemplo Netflix ofrece vitrinear sin necesidad de logearse pero eventualmente si uno quiere hacer algo mas tiene que logearse o registrarse primero. En este caso el usuario manifiesta su intención de logearse (un click o un selector) después de lo cual se le presenta la opción de ingresar su email y password. Si las ingresa correctamente queda logeado y puede acceder a todas las funcionalidades. Si las credenciales son incorrectas se le comunica al usuario para que vuelva a hacerlo. Desde la misma página de login donde se piden las credenciales hay dos links que son frecuentemente utilizados: uno para el caso que el usuario haya olvidado el password, y uno para el caso en que el usuario necesite registrarse (no tenía cuenta).

Cuando el usuario pincha el link para el caso de haber olvidado el password se le manda a otra página que le permite indicar que se le envíe un correo. Después de que el usuario haya logrado resetear su password usando el correo enviado puede intentar nuevamente el login.

Cuando el usuario pincha el link de registrarse se le envía a una página de registro y si sus datos son aceptables quedará además logeado. Si los datos son incorrectos (passwords no coinciden) debe seguir insistiendo. Puede ocurrir que estando en proceso de registro, el sistema detecte que el correo indicado ya existe. En ese caso se le envía a la página de login. La figura ilustra en parte lo explicado antes

Modele esta interacción como un diagrama de estados UML.