Escuela Superior Politécnica del Litoral (ESPOL)

Facultad De Ingeniería en Electricidad y Computación

CCPG1042

Diseño de Software

Primera Evaluación

II PAO 2024

Profesores: MSc. David Jurado, Dr. Carlos Mera Gómez

Compromiso de Honor

Yo, ……..……, matrícula ……..…… declaro que he sido informado y conozco las normas disciplinarias que rigen a la ESPOL, en particular el Código de Ética y el Reglamento de Disciplina. Al aceptar este compromiso de honor, reconozco y estoy consciente de que la presente evaluación está diseñada para ser resuelta de forma individual; que puedo comunicarme únicamente con la persona responsable de la recepción de la evaluación.

Acepto el presente compromiso, como constancia de haber leído y aceptado la declaración anterior y me comprometo a seguir fielmente las directrices que se indican para la realización de la presente evaluación. Estoy consciente que el incumplimiento del presente compromiso anulará automáticamente mi evaluación y podría ser objeto del inicio de un proceso disciplinario.

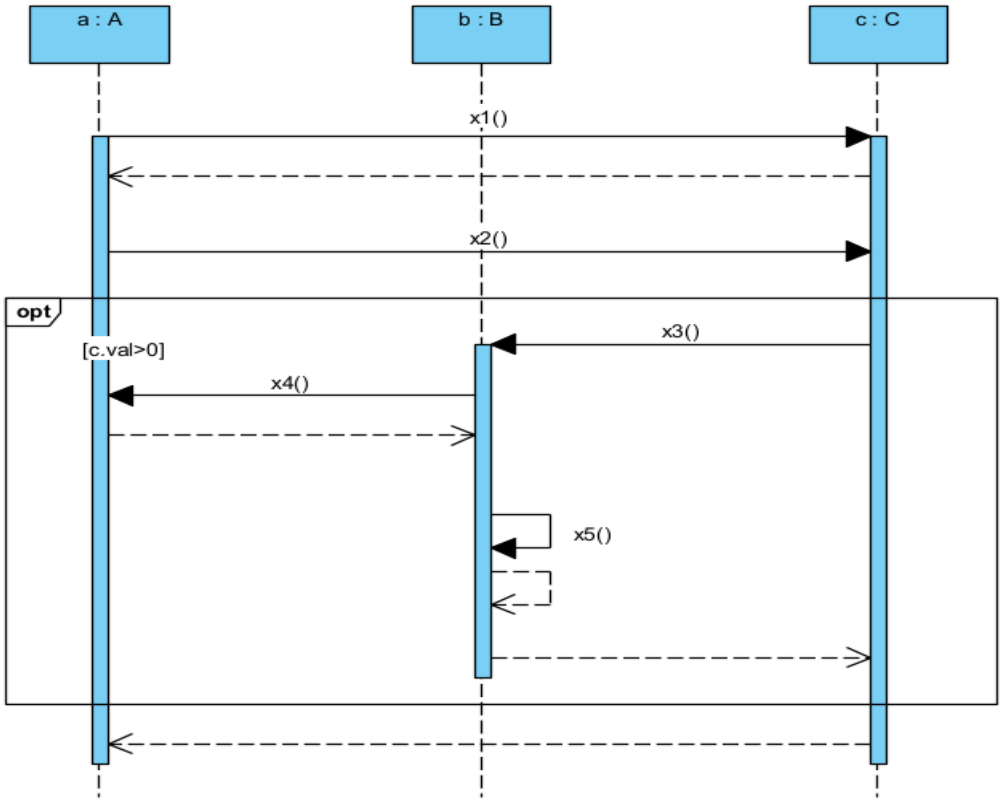
Firma: .....

Directrices:

- Duración del examen: 100 minutos.
- Por favor escriba su número de matrícula en la parte superior de cada hoja individual.
- Use pluma (no lápiz ni pluma con tinta roja).
- Por favor escriba de forma legible. Solo podemos calificar soluciones entendibles.
- Se prohíbe tener un teléfono móvil o cualquier otro tipo de dispositivo electrónico en su escritorio o en su ropa.
- El examen es a libro cerrado.
- Para preguntas de opción múltiple, se requiere la justificación de la respuesta seleccionada. Una corrección en la selección invalida la respuesta.

Sección A

1. Escriba el **mínimo** código fuente en Java para las clases A, B y C que capture el comportamiento del siguiente diagrama de secuencia UML. Los cuerpos de todos los métodos deben dejarse vacíos excepto para las llamadas indicadas en el diagrama. Constructores y propiedades (es decir, métodos get y set) de las clases no necesitan ser mostradas. **[16%]**



2. Elabore un **diagrama de secuencia** UML para la operación *calculateAssetsValue*. **[16%]**

```java
package exam2024_2;

import java.math.BigDecimal;
import java.util.Iterator;

public class AssetsCalculator {

    public BigDecimal calculateAssetsValue(Iterator<Asset> assets) {
        BigDecimal totalAssetsValue = new BigDecimal(0);
        while(assets.hasNext()) {
            Asset asset = assets.next();
            if(asset.isNew())
                totalAssetsValue = totalAssetsValue.add(asset.getValue());
            else
                totalAssetsValue = totalAssetsValue.add(asset.getDepreciatedValue());
        }
        return totalAssetsValue;
    }
}
```

3. A usted se le ha encargado el diseño de un sistema para reservas de habitaciones en un hotel. La descripción del proyecto es la siguiente:

Un hotel ofrece varios tipos de habitaciones: simple, doble y matrimonial. El sistema debe mantener un catálogo ya que hay varias habitaciones de cada tipo. Un cliente tiene el beneficio de reservar una habitación y pagarla hasta una semana previo a la estadía. En caso de reservas adicionales o reservas de múltiples habitaciones, estas deben pagarse inmediatamente. Una reserva siempre especifica número de huéspedes, fechas de inicio y fin de la potencial estadía; tiene una vigencia máxima de hasta tres meses. Una habitación puede estar disponible; ocupada por un viajero; reservada por un viajero; en mantenimiento o en limpieza post-ocupación. El administrador del hotel puede dar por terminada una estadía si el cliente incumple con las políticas del hotel.

   a. Elabore un **diagrama de clases** UML para este sistema. Haga uso de herencia, agregación o composición, según sea aplicable. Agregue cardinalidades a las relaciones. Agregue atributos y métodos cuando sea necesario. No es necesario incluir getters y setters de atributos. Modificadores de visibilidad (público, privado, etc.) no son requeridos. Declare cualquier asunción que considere necesaria. (La nota será dividida entre su selección de clases, su selección de tipos de relaciones y sus respectivas cardinalidades). **[30%]**

Sección B

4. Considere el caso de uso de ***Realizar pedido*** de un sistema de pedidos de Pizzas.

Acción del actor	Respuesta del sistema
1. Seleccionar pizza requerida.	2. Crear carrito de compras.
	3. Agregar pizza al carrito.
	4. Mostrar ingredientes adicionales.
5. Seleccionar ingredientes adicionales.	6. Agregar ingredientes a la pizza.
	7. Actualizar precio total del pedido.
	8. Mostrar vista general del pedido.
9. Seleccionar la opción “Pagar pedido”.	10. Mostrar formulario para ingresar pago.
11. Ingresar datos para pago.	12. Mostrar opción para “Confirmar pedido”.
13. Seleccionar “Confirmar pedido”.	14. Mostrar “Pedido realizado”.
	15. Enviar correo de confirmación de pedido.

   a. **Documente** este caso de uso utilizando una plantilla con al menos **seis** elementos, incluyendo actores, precondiciones, postcondiciones, flujo de eventos normal y flujo de eventos alternativo. Indique cualquier asunción **razonable** que realice. **[10%]**

5. ¿Cuál es la diferencia entre *Scattering* y *Tangling*? **[04%]**

6. Indique dos ejemplos de *cross-cutting concern* y justifique cada uno. **[06%]**

Sección C

7. Considere el código fuente en Java que se muestra a continuación.
   a. **Identifique** los principios SOLID que se están violando y **justifique** su respuesta. **[06%]**
   b. **Corrija** el código de tal manera que ya no se lo viole. Si lo considera necesario, usted puede crear interfaces, clases u operaciones. Puede utilizar un diagrama de clase UML. **[12%]**

```java
interface Playable {
    public void add(Playable item);
    public void remove(Playable item);
    public Playable getChild(int index);
    public void play();
}

class Media implements Playable {
    protected String title;
    protected String owner;
    protected String url;
    protected String type;

    public Media(String title, String owner, String url) {
        this.title = title;
        this.owner = owner;
        this.url = url;
        this.type = "video";
    }
    public void setType(String type){
        this.type = type
    }
    public void add(Playable item){
        throw new UnsupportedOperationException();
    }
    public void remove(Playable item){
        throw new UnsupportedOperationException();
    }
    public Playable getChild(int index){
        throw new UnsupportedOperationException();
    }

    public void play() {
        if(type == "video"){
            System.out.print("Reproduciendo video: ");
            System.out.println(title + " - Dirigido por: " + owner);
        }else if(type == "song"){
            System.out.print("Reproduciendo canción: " );
            System.out.println(title + " - Artista: " + owner);
        }
        System.out.println(url);
    }
}
class Playlist implements Playable {
    private String name;
    private List<Playable> components = new ArrayList<>();

    public Playlist(String name) {
        this.name = name;
    }
    public void add(Playable component) {
        components.add(component);
    }
    public void remove(Playable component) {
        components.remove(component);
    }
    public Playable getChild(int index) {
        return components.get(index);
    }
    public void play() {
        System.out.println("Reproduciendo playlist: " + name);
        for (Playable item : components) {
            item.play();
        }
    }
}
```