UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS.
PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS
ELEMENTOS TEMÁTICOS PARA EL EXAMEN FINAL
ASIGNATURA: DISEÑO ARQUITECTURAL DE SOFTWARE Y PATRONES[1]

PARTE I (1.0/5.0)

Debe presentar el catálogo de los patrones de diseño trabajados durante el desarrollo de la asignatura corregidos y completos. Se revisarán de manera aleatoria y deben corresponderse con las versiones ajustadas y mejoradas de los mismos de acuerdo a las recomendaciones realizadas en las sesiones de clase.

PARTE II (2.0/5.0)

Para el examen final de la asignatura debe usted estudiar, resolver los retos, revisar la ejecución de los programas[2] y elaborar los modelos funcionales, estructurales y dinámicos de cuatro patrones de control de concurrencia. La especificación de los retos, documentación de apoyo y código fuente de los ejemplos se pueden descargar a través del aula virtual. Sobre estos patrones de diseño se le harán preguntas que debe resolver basándose en los conceptos propios de la temática, de los modelos de ingeniería y de lo estudiado durante el curso al respecto.

PARTE III (2.0/5.0)

Se le harán dos preguntas de las que aparecen enumeradas a continuación:

1. **Establezca de manera sucinta la diferencia entre arquitectura empresarial y arquitectura de software.**
   La arquitectura empresarial hace referencia a un complejo proceso en el que se alinean datos, aplicaciones e infraestructuras con la finalidad de cumplir un objetivo dentro de una organización, por otro lado, la arquitectura de software es solamente una pequeña parte de este proceso. Ambas se usan para la descripción de elementos, conexiones, comportamientos, roles, conectores y restricciones de un sistema. AE: Trabaja con sistemas empresariales. Suele ser principalmente descriptiva. Se usa para la comprensión de una empresa, permitiendo cubrir cada uno de los elementos que la conforman. Usa un modelo de referencia. Descripción de sistemas empresariales, trabaja con factores humanos, económicos, etc. AS: Trabaja con sistemas de software. Pueden ser prescriptivos (ayuda a entender qué tenemos que hacer para obtener los resultados que queramos en el futuro.) y descriptivos (describe lo que ha pasado con estadísticas, gráficos, tablas e informes). Se usa para el entendimiento, implementación, manutención y evolución de un sistema. Trabaja con sistemas de entorno computacional, ambiente técnico.

2. **¿Cuáles se pueden considerar las dimensiones definitorias de lo que se denomina arquitectura de software?**
   - Características Arquitectónicas:
     - También denominadas atributos de calidad, esta dimensión agrupa las siguientes características:
       - Disponibilidad: Capacidad del sistema para estar operativo y accesible cuando se requiere.
       - Confiabilidad: Grado en que el sistema realiza sus funciones correctamente y sin fallos durante un período determinado.
       - Capacidad de Realización de Pruebas: Facilidad para verificar y validar el sistema mediante pruebas automatizadas o manuales.
       - Escalabilidad: Capacidad de manejar crecimiento en carga de trabajo (usuarios, datos, etc.) sin degradar el rendimiento.
       - Seguridad: Protección contra accesos no autorizados, vulnerabilidades y garantía de integridad de los datos.
       - Agilidad: Adaptabilidad para incorporar cambios rápidamente (requisitos, tecnologías, etc.).
       - Tolerancia a Fallos: Capacidad de continuar operando incluso ante fallos parciales del sistema.
       - Elasticidad: Escalabilidad dinámica (ej.: en la nube) para ajustar recursos según demanda.
       - Capacidad de Recuperación: Tiempo y esfuerzo requeridos para restaurar el sistema tras un fallo.
       - Rendimiento: Eficiencia en el uso de recursos (tiempo de respuesta, throughput, etc.).

        - Desplegabilidad: Facilidad para implementar el sistema en diferentes entornos (desarrollo, producción, etc.).
        - Facilidad de Aprendizaje: Curva de aprendizaje para que nuevos desarrolladores o usuarios comprendan el sistema.
    - Decisiones de Diseño Arquitectónico
    - Estructura
    - Principios de Diseño
3. **Diferencie conceptualmente arquitectura de referencia descriptiva de arquitectura de referencia prescriptiva.** La arquitectura de referencia prescriptiva nos habla de la etapa de diseño, en la que se realiza la concepción del sistema sin tener en cuenta las tecnologías, es el modelo de dominio. Por otro lado, la arquitectura de referencia descriptiva nos habla del proceso de producción, en el que se toma en cuenta las tecnologías mediante las cuales se implementará la arquitectura prescriptiva. Es necesario que ambas estén en total sinergia, de lo contrario podría presentarse degradación arquitectural. Arq. Descriptiva: Dependiente de la tecnología, modela un sistema ya existente. Arq. Prescriptiva: Es independiente de la tecnología, puede modelarse en un ADL (Arq. Description Language), hace una descripción conceptual del sistema, es previa a la implementación del propio sistema.

4. **Enuncie diez estilos arquitectónicos de software y explique de manera sucinta uno de ellos.**
    - Centrado en datos
        - Repositorio pasivo: Clientes solamente pueden leer.
        - Repositorio activo: Sistemas construido a partir de componentes preexistentes. (Pizarra). Clientes relativamente independientes entre sí y del repositorio. Se pueden agregar clientes. Cambios en clientes no afectan a otros. Acoplar clientes disminuye el beneficio, pero mejora el desempeño.
    - Flujo de datos
        - Secuencia de lote.
        - Pipe & Filter:
            - Provee una estructura para los sistemas que procesan un flujo de datos. Luego cada paso de procesamiento se encapsula en un componente de filtro. Los datos pasan a través de tuberías (Pipe) entre filtros adyacentes.
            - Ventajas: Composición funcional, filtros son cajas negras, puede combinarse filtros y canales jerárquicamente, se puede componer concurrentemente y de forma distribuida ya que cada filtro procesa su entrada aislada del sistema.
            - Desventajas: Predominio de pensamiento secuencial, costo basado en el peor desempeño, puede tener abrazos mortales (si falla un filtro se cae el flujo), cada filtro opera individualmente eso genera costo de tiempo en respuesta.
    - Capas
        - Cliente / Servidor.
        - Máquina Virtual.
            - Busca portabilidad: Ejecutar funcionalidad no nativa al hardware o software sobre el que es implementado.
            - Se concreta en interpretadores, sistemas basados en reglas, “shells” sintácticos y procesadores de lenguaje de comandos.
            - Un interpretador adiciona flexibilidad, pero genera costos de desempeño por la computación necesaria involucrada en tiempo de ejecución.
            - Los datos de estado del programa brindan contexto a la interpretación.
    - Llamada y Retorno
        - Programa Principal y Subrutina.
        - Llamada procedimientos remotos.
        - Sistemas orientados a objetos.
        - Sistemas basados en capas.
    - Componentes independientes
        - Basados en eventos: Control por parte del modelo, modelo Publicar / suscribir, usa gestor de mensajes (patrón observador), desacoplamiento de componentes, facilita integración de componentes y procesos.
        - Basados en componentes de software con paso de mensajes.
        - Comunicación de procesos.
    - Memoria compartida
        - Blackboard.
        - Rule Based.

5. **Enuncie tres subestilos arquitectónicos del estilo "Invocación implícita".**
    - Publish-Subscriber: desean hacer broadcast de mensajes a los suscriptores
        - Usarlo si:
            - Los componentes están muy débilmente acoplados
            - Los datos de suscripción son pequeños y son eficientemente transportados
        - No usarlo si:
            - No se dispone de un middleware para soportar un alto volumen de datos
    - Basado en Eventos: Componentes independientes emiten y reciben eventos asincrónicamente comunicados a través de una red
        - Usarlo si:
            - Los componentes son concurrentes e independientes
            - Los componentes son heterogéneos y distribuidos en red
        - No Usarlo si:
            - Se requiere garantizar el procesamiento en tiempo real de los eventos
    - Blackboard: Múltiples componentes colaboran mediante un repositorio central ("pizarra"), donde escriben/leen datos reactivamente.
        - Usarlo si:
            - El problema requiere solución colaborativa (ej.: sistemas de diagnóstico médico, IA).
            - Los componentes son especializados y aportan conocimiento parcial.
        - Evitarlo si:
            - La coordinación centralizada introduce cuellos de botella.
6. ***ASDL* (*Architectural Style Description Language*, en el inglés) es una propuesta de Lenguaje de Descripción Arquitectural elaborada por _____ y se basa para la especificación estructural en _____ y para la especificación comportamental en _____.**
   Michael Rice y Stephen Seidman
   Lenguaje Z
   Álgebra de procesos (CSP)

7. **¿Qué consideraciones son relevantes en un modelo de valoración de calidad de software?**
8. **¿Qué atributos de calidad podrían ser útiles para valorar la calidad de software "Open Source"?**
9. **¿Qué abstracción favorece el modelado modular usando lenguaje Z? Explique sucintamente y de manera precisa.**
   La abstracción que favorece el modelado modular es la especificación formal a través de los esquemas, ya que expresa que hace un sistema y en qué orden lo hace sin describir como lo hace (sin una tecnología específica).

10. **Enuncie los patrones de diseño creacionales. ¿Cuál es su objetivo? Se le presentará una situación problémica de software donde pueda aplicar uno de ellos. Usted deberá identificar cuál patrón de diseño podría ser el más adecuado y sustentar su respuesta.**
    - Método factoría:
        - A veces, es posible que un objeto de aplicación solo sepa que necesita acceder a una clase desde dentro de la jerarquía de clases, pero no sabe exactamente qué clase del conjunto de subclases de la clase principal debe seleccionarse.
    - Fábrica Abstracta
        - Cuando un objeto cliente desea crear una ejemplificación de una clase de un conjunto de clases dependientes relacionadas sin tener que saber qué clase concreta específica se va a instanciar. Cuando se tienen familias de clases que cuentan con una característica de afinidad entre varias subclases de ellas.
    - Singleton
        - A veces, puede ser necesario tener una y solo una instancia de una clase determinada durante la vida útil de una aplicación. Puede ser porque una sola instancia de la clase es suficiente.
    - Builder
        - Se utiliza cuando la creación de objetos es compleja (y compuestos) y una serie de pasos que constituyen el objeto pueden ser implementados de diferentes maneras produciendo diferentes representaciones del objeto. Se emplea en GUI por lo general.
    - Prototype.
        - Cuando un cliente necesita crear un conjunto de objetos que son similares o difieren entre sí solo en términos de su estado y es costoso crear tales objetos en términos de tiempo y procesamiento involucrados.

Como alternativa a la construcción de numerosas fábricas que reflejan las clases que se van a instanciar (como en el método Factory)

Su objetivo es: lidiar con la creación de objetos de una manera uniforme, simple y controlada; permitiendo la encapsulación de detalles de las clases que serán ejemplificadas y aquellas que las ejemplificarán, todo lo anterior con la finalidad de reducir el acoplamiento.

11. **Enuncie los patrones de diseño comportamentales. ¿Cuál es su objetivo? Se le presentará una situación problémica de software donde pueda aplicar uno de ellos. Usted deberá identificar cuál patrón de diseño podría ser el más adecuado y sustentar su respuesta.**
    - Método plantilla
        - Se emplea en situaciones donde hay un algoritmo en el cual algunos de sus pasos pueden ser implementados de múltiples maneras.
    - Estrategia
        - Cuando se desean configurar algoritmos en tiempo de ejecución por parte de quien usa la aplicación o por parte del programador quien implementa condiciones internamente que determinan cual algoritmo usar.
    - Estado
        - Se emplea cuando se es de interés modelar diferentes estados por los que puede pasar una clase, en donde en cada estado se pueden desarrollar diferentes actividades o algoritmos.
    - Comando
        - Si se busca parametrizar objetos con una acción a realizar. Encapsula la petición de un objeto, permitiendo parametrizar a los clientes con diferentes operaciones, hacer cola o llevar un registro de las peticiones y poder deshacerlas.
    - Intérprete
        - Se emplea cuando se desea desarrollar pequeños interpretadores, donde se recibe un lenguaje con gramática y semántica (¿contexto?) asociados que genera la ejecución de instrucciones basadas en ese lenguaje.
    - Observador
        - Se emplea cuando se desea tener el reporte de un cambio de estado de un objeto de interés (observado) por parte de otros objetos (observadores)
    - Memento
        - Se emplea cuando se cuenta con un estado operable en una aplicación OO y que en un momento haya una lectura o salida inesperada y se desee guardar el estado en el que se interrumpe la ejecución para iniciarla donde se encontraba anteriormente.
    - Mediador.
        - Define un objeto que encapsula cómo interactúan una serie de objetos. Promueve el bajo acoplamiento al evitar que los objetos se refieran a otros explícitamente, permite variar la interacción de forma interdependiente.

    Su objetivo es: lidiar con los detalles de la asignación de responsabilidades entre objetos por lo tanto son importantes para describir los mecanismos de comunicación entre objetos, además de ofrecer en tiempo de ejecución la selección de mecanismos de selección como lo es el caso del patrón estrategia.

12. **Enuncie los patrones de diseño estructurales. ¿Cuál es su objetivo? Se le presentará una situación problémica de software donde pueda aplicar uno de ellos. Usted deberá identificar cuál patrón de diseño podría ser el más adecuado y sustentar su respuesta.**
    - Bridge
        - Cuando se desea "re utilizar" las clases de una jerarquía ya existente (de implementación). Se desea utilizar un puente para unir dos jerarquías en tiempos de ejecución por medio de una asociación con un atributo a la jerarquía de implementación.
    - Adaptador
        - Cuando a un cliente le interesa que no se varíe o se cambie la firma de una operación que desea realizar y que a su vez desea implementar una operación cuya firma no coincide con la firma empleada anteriormente. Cuando una clase existente provee una funcionalidad requerida por un cliente, pero su interfaz o protocolo de mensajería no es lo que el cliente espera.
    - Decorador
        - Cuando se desea ampliar funcionalidad en una jerarquía de clases sin emplear el concepto de Herencia de la POO.
    - Cadena de Responsabilidad

- Útil cuando se da un Flujo de trabajo donde se la da la potestad a alguien. Flujo de trabajo en el ambiente empresarial: como fluye determina autorización en diferentes instancias de autoridad en la organización para que sea aprobado.

- Proxy
  - Cuando se desea satisfacer un requerimiento no funcional (relacionado con atributos de calidad como el desempeño, confiabilidad, etc.).
  - Se desea intermediar o representar un objeto.
  - El proxy puede realizar lo que requiera antes de redirigir el llamado al objeto que se desea referenciar.
  - Virtual: Permite la creación de objetos bajo demanda (Creación perezosa de objetos).
  - Contador o de Conteo (Counting): Permite un mecanismo de auditoría (como creación de logs o toma de tiempos o de accesos al objeto) antes de ejecutar algún método sobre el objeto destino.
  - 

Tipo	Propósito
Proxy remoto	Provee acceso a un objeto localizado en un espacio de direcciones diferente.
Proxy virtual	Permite la creación bajo demanda de objetos intensivos en memoria.
Proxy caché o servidor	Provee la funcionalidad requerida para almacenar los resultados de las operaciones sobre el objeto destino que fueron usadas más recientemente y que son frecuentes
Proxy firewall	Protege objetos destino de objetos cliente perniciosos o viceversa
Proxy de control de acceso	Controla el acceso de diferentes clientes al objeto destino en diferentes niveles
Proxy de sincronización	Permite acceso concurrente a un objeto destino por parte de diferentes objetos cliente
Proxy de referencia inteligente	Previene el borrado accidental del objeto destino cuando hay clientes concurrentes con referencias de éste.
Proxy contador o contable	Provee un mecanismo de auditoría antes de ejecutar algún método sobre el objeto destino

- Fachada
  - Cuando el cliente debe interactuar con un subsistema de clases y existe un alto protocolo de mensajería que se desea simplificar. Útil en un escenario remoto (ej. Validación en un servidor) para reducir el prot. De mensajería, disminuyendo el caudal de datos.
- Objetivo: Lidian con la delegación de responsabilidades de otros objetos, además de facilitar la comunicación interobjeto cuando existe algún objeto no accesible finalmente también permiten agregar objetos y asignarles recursos de la manera adecuada.

13. **Enuncie los patrones de diseño sobre colecciones. ¿Cuál es su objetivo? Se le presentará una situación problémica de software donde pueda aplicar uno de ellos. Usted deberá identificar cuál patrón de diseño podría ser el más adecuado y sustentar su respuesta.**
    - Flyweight
      - Cuando se cuenta con un conjunto de datos intrínsecos (NO varían dependiendo el contexto del objeto) y no se desea almacenar dichos datos en cada uno de los objetos a ejemplificar. Cuando se tiene un gran número de objetos que comparten la misma información y que no varía dependiendo su contexto.
    - Visitador
      - Cuando se desea externalizar una funcionalidad sobre una colección de objetos, de modo que dicha funcionalidad no sea lograda o implementada en cada una de las clases. Cuando no se desea implementar una funcionalidad en cada una de las clases de una jerarquía si no en una clase externa.
    - Iterador (Interno y Externo)
      - Permite al objeto cliente acceder a los contenidos de una colección de datos u objetos (Contenedor) de manera secuencial, sin tener ningún conocimiento acerca de la representación interna de sus componentes
    - Compuesto
      - Cuando un componente u objeto puede ser clasificado en una o dos categorías (Componentes Individuales o Compuestos), este patrón brinda una interfaz común para ambas categorías.

    Su objetivo es: Permitir la interacción con colecciones de objetos, permitiendo la composición de clases para crear estructuras más robustas, evitando la existencia de información duplicada y creando operaciones para trabajar con la colección de manera heterogénea.
14. **Explique el concepto de *Templates* a la luz de *ASDL (Architectural Style Description Language, en el inglés).***
    Representa los componentes que se pueden incluir dentro de la arquitectura. Se clasifican en tres grupos.

- Las plantillas primitivas: Aquellas plantillas cargadas por la librería por defecto
- Las plantillas de referencia: Colección de plantillas que pueden ser usadas para realizar sistemas de software
- Interfaces de módulos compuestos encapsulados: Colección de primitivas

15. **Explique el concepto de *Settings* a la luz de *ASDL (Architectural Style Description Language, en el inglés).***
    Representan arquitecturas como una unidad autocontenida sin conexiones externas que ya han sido construidas mediante el uso de Templates como nodos computacionales. (Cada nodo corresponde a componentes dentro de la arquitectura de software). Se construyen a partir de la ejemplificación de los templates.
16. **Explique el concepto de *Units* a la luz de *ASDL (Architectural Style Description Language, en el inglés).***
    Representa la jerarquía del sistema, es decir poder encapsular settings con la finalidad de poderlos usar como templates.
17. **¿Para qué sirve *CSP (Communicating Sequential Processes)* en ASDL?**
    CSP es una herramienta que se usa para describir los procesos de comunicación y el procesamiento de los mensajes, así como definir los elementos de entrada y salida de una plantilla. Utiliza el álgebra de procesos para modelar el comportamiento interno de un elemento definido dentro de la arquitectura, así como el tratamiento y direccionamiento de mensajes.
18. **¿A qué se denomina jerarquización y refinamiento en el área de conocimiento de arquitecturas de software?**
    - Jerarquización: cómo está conformado por dentro un componente en una arquitectura. La forma en que se estructura la comunicación y las invocaciones de operaciones entre los componentes.
    - Refinamiento: Es el mapeo desde una especificación formal a una tecnología. Representa la evolución desde lo más abstracto de un modelo de AS a lo más concreto y específico.
19. **Diferencie los enfoques de especificación axiomática y algebraica en el área de conocimiento de arquitecturas de software.**
    Enfoque axiomático: Desarrolla aserciones en la lógica de predicados que caracterizan los posibles estados de un cálculo. Las acciones de un programa se ven como transformadores predicados que mueven el cálculo de un estado a otro. Usa notación algebraica.
    Enfoque algebraico: Consiste en que las interacciones entre un sistema y su entorno se pueden modelar a través de una abstracción matemática llamada proceso.
20. **Enuncie los patrones arquitectónicos más conocidos. Explique de manera sucinta uno de ellos.**
    - Programación por capas
    - Tres niveles
    - Pipeline o Pipe and Filter.
    - Invocación implícita
    - Arquitectura dirigida por eventos
    - Peer-to-peer
    - Arquitectura orientada a servicios
    - Objetos desnudos
    - **MVC (Modelo Vista Controlador)**
        - El funcionamiento de este se basa en tener un controlador el cual será el mediador de aplicar todos los cambios que se realicen en el modelo y cómo estos se reflejan en la Vista, comúnmente se acompaña de el uso de base de datos, ya que este patrón no tiene establecido el uso de esta como parte de su diseño.
    - **Arquitectura en pizarra:**
        - Los datos compartidos pueden ser: Un repositorio pasivo, un repositorio activo (pizarra).
        - Es un estilo arquitectural que da una solución estructural a la integración.
        - El repositorio activo demuestra esto permitiendo:
            - Sistemas construidos desde componentes preexistentes logran integrarse a través de blackboard mechanism.
            - Los clientes son relativamente independientes cada uno del otro y el repositorio de datos también.
            - Nuevos clientes pueden añadirse.
            - Cambiar la funcionalidad de un cliente no afectaría a los demás. Acoplar a los clientes disminuye este beneficio, pero puede mejorar el desempeño.
    - **Arquitectura orientada a servicios (SOA):**
        - Un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de propiedad. Proporciona un medio uniforme para ofrecer, descubrir, interactuar y usar capacidades para producir los efectos deseados consistentes con las condiciones y expectativas medibles.

21. **Basándose en los modelos funcionales, estructurales y dinámicos explique el ejercicio de recuperación de diseño del ejemplo asociado al patrón de diseño "MEMENTO".**
    Se emplea cuando se cuenta con un estado operable en una aplicación OO y que en un momento haya una lectura o salida inesperada y se desee guardar el estado en el que se interrumple la ejecución para iniciarla donde se encontraba anteriormente.
    - Emplea dos clases:
        - Originator que tiene memoria de algún estado que es de interés para ella.
        - Memento: Tiene responsabilidad de tener memoria del estado de interés de originator (Extiende de Serializable de Java).
    - Si se cierra la app por algún evento, se puede iniciar esta donde se encontraba anteriormente.
    - En java se emplea el modificador Serializable que permite guardar el objeto.x- Originator usa memento para recordar cuando lo requiere!
    - MementoHandler: Se encarga de manipular el memento, de ejemplificar el objetoMemento en el que se requiere guardar el estado de interés y guardarlo en disco.

    - Sistema que convierte registros de clientes a sentencias SQL.
    - Lee del archivo de texto Data.txt los registros, los valida y si tienen algún problemaguarda el estado en el memento para que el usuario corrija el registro y retomar desde dicho registro.
    - Cuenta con la clase memento que indica el id del último proceso, empleaserialización en memento para guardar los objetos en disco. - MementoHandler maneja el memento - Clase memento guarda el objeto.
    - DataCoverter se encarga de crear el memento y obtener el ultimo procesado. ES ELORIGINATOR.
22. **Basándose en los modelos funcionales, estructurales y dinámicos explique el ejercicio de recuperación de diseño del ejemplo asociado al patrón de diseño OBSERVADOR.**

    Actualiza los reportes según la selección del usuario del departamento requerido, siendo ReportManager el observado, y YDTChart junto MonthlyReport los observadores. Es un observador de tipo push, donde al existir un cambio se notifica a los observadores.

    Observer: Se emplea cuando se desea tener el reporte de un cambio de estado de un objeto de interés (observado) por parte de otros objetos (observadores),

    - Es usado en bolsas de valores y app. Distribuidas.
    - Subject es el observado.
    - Observadores les interesa observar si el subject cambió de estado o no.
    - Observadores son conocidos por el subject por medio de una asociación (Tipocolección o vector) que muestra un registro de los obs. a los cuales les interesa su cambio de estado.
    - Para que el subject sepa ello deben existir los métodos attach y detach que permiten a los observadores "suscribirse" en el subject para que los tengan en cuenta en el cambio de estado.
    - Observadores deben saber cuál es su sub., tienen una asociación para saber cuál es el observado (sub.) de su interés
    - **Existen dos modelos: Pull y Push**
        - Modelo Pull: Subject provee una interfaz de operación para que los observados puedan estar averiguando periódicamente si el cambió de estado, luego subj no se preocupa por indicar a los observados cuando cambió.
        - Modelo Push: El observado realiza un mensaje de difusión a todos los observados indicando que cambió de estado cada vez que ello ocurre. (Es el más utilizado).
        - Pull no se usa mucho puesto que depende del tiempo de actualización en los observadores y si ese tiempo es muy largo pueden perderse cambios de estado.
    - Sistema para visualizar informes de ventas de una tienda con varios departamentos.
    - 3 ventanas: 1 ventana con la selección del hardware (Observado o subj.) y 2 ventanas observadoras que varían dependiendo del estado de ella (Transacciones del mes y reporte de ventas anual).
    - El ReportManager es la clase observada, tiene un vector de observadores llamado observerList. Realiza la interfaz Observable que tiene los métodos de notificación (MODELO PUSH), registrarse a la lista o des registrarse.

1 **Preparó Henry Alberto Diosa Ph.D.**
2 **De ser necesario debe generar los programas principales que validen las técnicas de concurrencia.**
23. **Basándose en los modelos funcionales, estructurales y dinámicos explique el ejercicio de recuperación de diseño del ejemplo asociado al patrón de diseño ESTADO.**
    - Se cuenta con una jerarquía de estados que es una máquina de estados de la clase que es relevante.
    - En Ing. De Software se recomienda desarrollar la máquina de estados para entender el ciclo de vida de los objetos ejemplificados de esa clase.

- Diferencias: Máquina virtual modela el comportamiento intraobjeto, secuencia modela comportamiento interobjeto.
- Clase relevante hace las veces de contexto, tiene una referencia a los posibles estados (objState) el cual va variando en función en que el contexto varíe de estado debido a la lógica de negocio.
- Sistema que simula transacciones sobre una cuenta bancaria.
- Depósitos o retiro de cuenta con costo, sin costo y con sobregiro de la cuenta.
- Cuenta de negocios tiene una serie de estados cuando se desea depositar o retirar.
- State tiene un atributo de contexto que es la referencia o cargador a la cuenta de ahorros creada anteriormente.

24. **Basándose en los modelos funcionales, estructurales y dinámicos explique el ejercicio de recuperación de diseño del ejemplo asociado al patrón de diseño MÉTODO PLANTILLA.**
    Se realiza la validación de una tarjeta de crédito, teniendo en cuenta las diversas características que cada tarjeta de crédito tiene y las características que presenta cada franquicia entre Visa, MasterCard y Diners.
    - Se emplea en situaciones donde hay un algoritmo en el cual algunos de sus pasos pueden ser implementados de múltiples maneras.
    - En este escenario, el patrón es de los más usados y sencillos.
    - Sugiere mantener separado el contorno del algoritmo en un método diferente llamado como método plantilla dentro de la clase que puede ser referenciada como clase plantilla, permitiendo las diferentes especificaciones en diferentes subclases.
    - Se emplea una plantilla que se reúsa de manera repetitiva y representa un esquema básico que se establece, luego basado en esa plantilla se puede operar de manera repetitiva un algoritmo o proceso.
    - Se emplea un método que se puede reusar repetitivamente porque tiene una plantilla.
    - El método plantilla tiene el esquema de invocación de una serie de operaciones que pueden ser primitivas o extensiones basadas en especializaciones de clases donde se generan nuevas operaciones.
    - Dos enfoques: Clase abstracta (se tienen operaciones en común con la misma lógica) o interfaz (no se define la lógica de una operación, luego en cada subclase la lógica varía).

    - Validador de tarjetas de crédito.
    - Tarjetas de tipo Diners, Visa y Master
    - Las tarjetas cuentas con los mismos pasos de validación
    - Los pasos de expiración, caracteres válidos y el algoritmo de suma de chequeo del dígito son el mismo para todas las franquicias.
    - Estos pasos son implementados en la clase abstracta CreditCard.
    - Los demás pasos son sobrescritos en las subclases.
    - El método plantilla es isValid, tiene la palabra reservada Final para que no pueda ser sobrescrito en las subclases.
25. **Basándose en los modelos funcionales, estructurales y dinámicos explique el ejercicio de recuperación de diseño del ejemplo asociado al patrón de diseño "ESTRATEGIA".**
    Ejercicio: Se cifra un mensaje según el método que se quiera, generando por cada método un archivo txt donde se guarda el mensaje.
    - Cuando se desean configurar algoritmos en tiempo de ejecución por parte de quien usa la aplicación o por parte del programador quien implementa condiciones internamente que determinan cual algoritmo usar.
    - Se cuenta con un contexto: Clase que ejemplifica objetos a los que se le configurarán los diferentes tipos de algoritmo.
    - Jerarquía de estrategias o algoritmos: Abstracciones, el contexto tiene una referencia a la jerarquía para que en tiempo de ejecución desde las clases concretas se puedan ejemplificar los diferentes algoritmos que se le pueden configurar a la clase que ejemplifica
    - Es útil cuando existe énfasis en el uso diverso de algoritmos y escenarios problémicos en tiempo de ejecución.
    - EncryptLogger es el contexto, tiene referencia a las estrategias.
    - Cifrador de mensajes en formato de texto plano.
    - Clase contexto que es un EncryptLogger que tiene una referencia a la estrategia actual, un conjunto de estrategias que son algoritmos que varían en tiempo de ejecución dependiendo el tipo de cifrado en cuestión.
    - Jerarquía de loggers.
26. **Responda de manera sucinta: ¿Qué se puede entender por un modelo independiente de la computación? Presente un ejemplo.**
    Estos modelos no están amarrados a un paradigma de programación, son funcionales y declarativos, permiten hacer una idea general de qué hace el programa y ejemplo claro de esto son los diagramas de casos de uso.
27. **Responda de manera sucinta: ¿Qué se puede entender por un modelo independiente de la plataforma? Presente un ejemplo.**

Permiten establecer la estructura que tendrá un programa, sin tener en cuenta en qué lenguaje de programación se utilizará, permite mostrar los elementos que se utilizarán y las distintas relaciones entre ellos. El ejemplo son los Diagrama de Clases

28. **Responda de manera sucinta: ¿Qué se puede entender por un modelo específico de la plataforma? Presente un ejemplo.**
    Permite establecer un modelo teniendo en cuenta en qué lenguaje de programación se utilizará, se ciñe a todas las restricciones que este pueda presentar, ejemplo de esto es un diagrama de secuencia.
29. **Enuncie los patrones arquitectónicos más conocidos. Explique de manera sucinta uno de ellos.**
30. **Describa de manera sucinta la *Chemical Abstract Machine*.**
    La Chemical Abstract Machine (CHAM) es un modelo formal basado en una metáfora química para describir sistemas computacionales. En este modelo, los estados del sistema se representan como soluciones químicas compuestas por moléculas que interactúan según reglas de reacción explícitas.

    Características principales:
    - Molecules y soluciones:
        - Las moléculas son términos algebraicos que representan componentes del sistema (procesos, datos, etc.).
        - Las soluciones son multiconjuntos de moléculas, análogas a mezclas químicas.
    - Reglas de transformación:
        - Reacciones: Reglas que modifican las moléculas (ejemplo: A + B → C).
        - Leyes generales: Incluyen la Ley de Reacción (sustitución), Ley Química (evolución en subsoluciones) y Ley de Membrana (aislamiento de subsistemas).
    - Membranas y airlocks:
        - Las membranas encapsulan soluciones para modularidad.
        - Los air locks permiten extraer/reabsorber moléculas de membranas.
    - Paralelismo y no determinismo:
        - Las reacciones pueden ocurrir en paralelo si no hay conflictos.
        - Elección no determinista cuando múltiples reglas son aplicables.

    Aplicación:
    Se usa para especificar y analizar arquitecturas de software, como compiladores secuenciales o concurrentes, destacando conexiones y comportamientos mediante operaciones químicas abstractas.

    Ejemplo: En un compilador secuencial, las fases (lexer, parser) son moléculas que reaccionan en un orden fijo, mientras que, en uno concurrente, interactúan mediante un "repositorio compartido" (otra molécula).
31. **¿Para qué sirve el operador *Airlock* en la CHAM?**

    Operador airlock es poder sacar una molécula de la membrana se representa por Δ apuntando a la izquierda, como se puede extraer una molécula se hace semi impermeable la membrana

PROCEDIMIENTO

El examen final se efectuará en el espacio de encuentro presencial usado durante el semestre para desarrollar las sesiones de clase. Cada estudiante debe entregar lo correspondiente a las **Partes I y II** grabado en un medio de almacenamiento digital debidamente rotulado y organizado[3]. La sustentación será individual.

FECHA DE REALIZACIÓN: *Julio 11 de 2025 a partir de las 8:00 a.m., en la sala donde convencionalmente se ha realizado la clase durante este semestre académico.*

3 Usar un archivo ***Readme.txt*** indicando autoría del entregable.