Programación 4

PARCIAL FINAL EDICIÓN 2022

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su número de parcial junto al parcial
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (25 puntos)

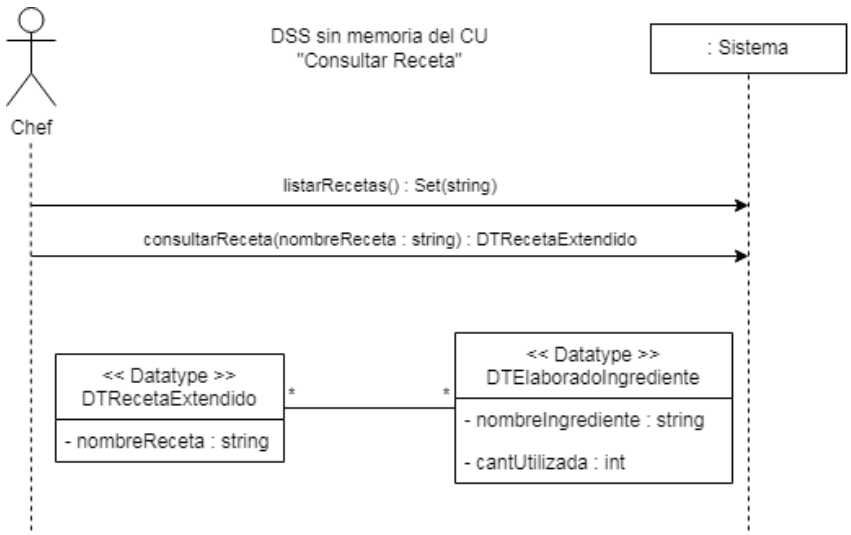
Un restaurante desea desarrollar un sistema para que su Chef gestione su recetario de cocina. Su empresa ha decidido seguir un proceso iterativo e incremental para ir satisfaciendo distintas necesidades del restaurante.

Parte A

Inicialmente requiere que en el sistema puedan existir recetas, de las cuales interesa saber un nombre (que las identifica) y si corresponde a una entrada, un plato principal o un postre. Para la elaboración de las recetas es necesario saber qué ingredientes se utilizarán, así como la cantidad (expresada en gramos o mililitros) de ellos a utilizar. De los ingredientes interesa conocer su nombre (que los identifica), si es líquido y una categoría (lácteo, vegetal o carne).

Además, se definió el siguiente caso de uso que permite consultar una receta y su equipo ya modeló el Diagrama de Secuencia del Sistema (DSS) correspondiente.

Caso de Uso	Consultar Receta
Actor	Chef
Descripción	El caso de uso comienza cuando el/la Chef desea consultar la información existente sobre las recetas. Para ello el sistema lista todas las recetas existentes (mostrando su nombre) y el/la Chef selecciona una de ellas. Luego, se muestra para la receta su nombre así como todos los nombres de los ingredientes necesarios para realizarla y las cantidades requeridas.



Se pide:

a) Realizar el Modelo de Dominio de la realidad planteada **sin incluir** restricciones no estructurales.
b) Escribir las pre y post condiciones de los contratos de **todas** las operaciones del sistema identificadas para el caso de uso "Consultar Receta".

Parte B

En una siguiente fase, se le presenta el requerimiento de modelar la existencia de utensilios de cocina que serán necesarios para la elaboración de las recetas. Estos opcionalmente podrán tener un comentario sobre el uso que se le dará en una receta en particular. A su vez tendrán un identificador creado por el sistema.

Así mismo dispone de la descripción del CU "Agregar Utensilios".

Caso de Uso	Agregar Utensilios
Actor	Chef
Descripción	El caso de uso comienza cuando el/la Chef desea registrar en el sistema algunos de los utensilios que utilizará en sus recetas. Inicialmente existirán en el sistema 2 tipos de utensilios: cubiertos y electrodomésticos. Si se desea agregar un cubierto, el sistema solicita su tipo (cuchara, tenedor o cuchillo) y el material del cual está hecho. En caso de que se desee agregar un electrodoméstico, se solicitará la fecha límite de su garantía. Luego el sistema da la opción de confirmar o cancelar. En caso de confirmar, se da de alta el utensilio con los datos ingresados y, en caso contrario, se cancela el registro. Finalmente, se da la posibilidad de continuar agregando utensilios o finalizar.

Se pide:

a) Realizar el DSS para el caso de uso "Agregar Utensilios", indicando el uso de memoria del Sistema así como los datatypes utilizados.
b) Realizar un nuevo Modelo de Dominio que contemple toda la realidad planteada en ambas fases. Incluir las restricciones en lenguaje natural que correspondan.

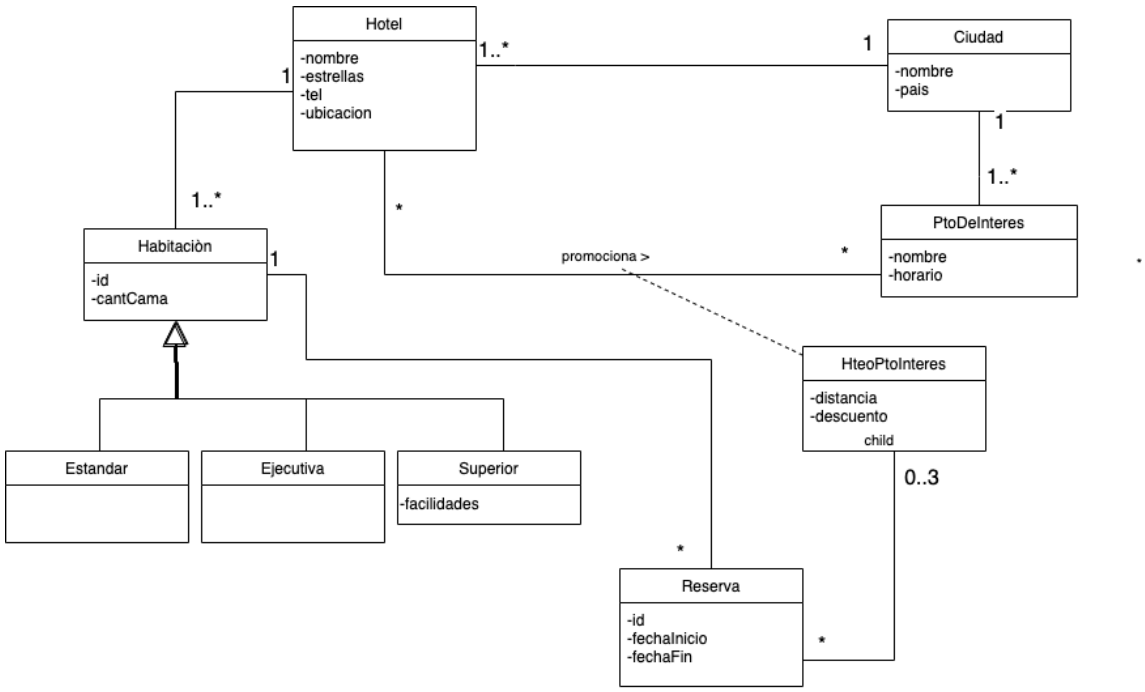
Problema 2 (30 puntos)

Parte A

Explique el problema tipo que resuelve el patrón Strategy e ilustre gráficamente su estructura general y participantes.

Parte B

Considere la siguiente realidad para un sistema de gestión de reservas en hoteles en distintas ciudades. En cada ciudad existen puntos de interés (ej: museos) promocionados por hoteles para quienes se realiza un descuento en la entrada. Realizando una reserva en un hotel es posible acceder a los descuentos ofrecidos para dicho hotel, hasta un máximo de tres puntos de interés por reserva. Se realizó el siguiente Modelo de Dominio de la realidad.



Restricciones:

- El nombre de Hotel es único.
- El nombre de Ciudad es único.
- El nombre de PtodeInteres es único por ciudad.
- El id de Habitación es único por Hotel.
- El id de Reserva es único.
- La fecha de inicio de una reserva debe ser menor a la fechaFin.
- No existen Reservas para una misma Habitación cuyas fechas de estadía se superpongan.
- El HotelPtoInteres de interés asociado a una reserva debe corresponder a una reserva para la habitación del mismo hotel.
- Si existe una instancia de la relación HotelPtoInteres entre un Hotel y un PtoDeInteres, entonces ambos se encuentran en la misma ciudad.

Se definió la siguiente operación del sistema del controlador CtrlCiudades.

CtrlCiudades::listarReservasCiudadConPuntosDeInteresXTipo(String: ciudad, String tipo): Set(String)	
Descripción	Retorna todos los identificadores de reservas de la ciudad que estén asociadas al menos con un punto de interés y cuyo habitación sea del tipo especificado.
Parámetros	String `ciudad`: nombre de la ciudad String `tipo`: tipo de habitación
Precondiciones	Pre1: Existen una ciudad con nombre `ciudad` Pre2: El `tipo` es uno de los siguientes (estándar, ejecutiva, superior)

Se pide:

a) Realizar el Diagrama de Comunicación correspondiente a la operación especificada en el controlador CtrlCiudades, incluyendo las visibilidades. Asuma que CtrlCiudades posee una colección global de ciudades por la cual se debe acceder a la información. Ninguna otra colección global ni controladores adicionales pueden ser definidos.

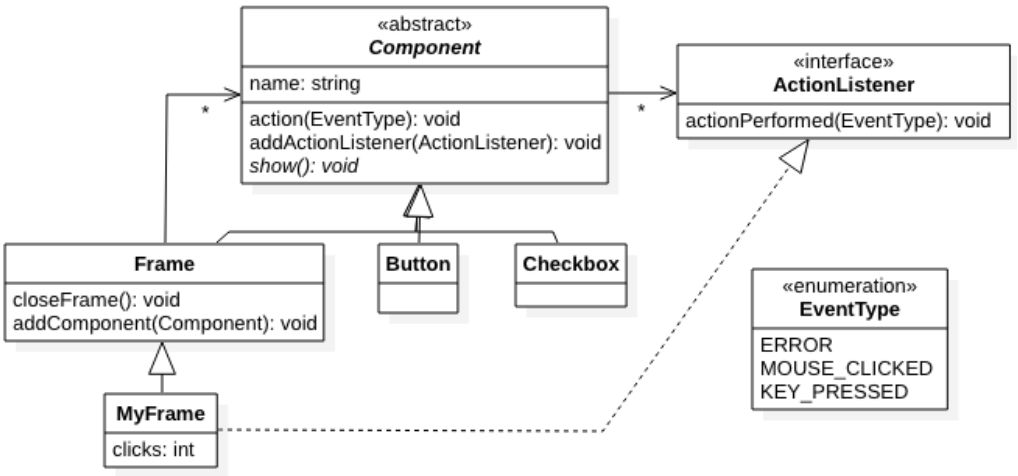
b) Realizar el Diagrama de Clases de Diseño (DCD) resultante del diseño realizado en a).

Problema 3 (30 puntos)

Como parte del diseño de una biblioteca para interfaces gráficas de usuario (GUI), se realizó el diagrama de clases de la figura. Allí se puede observar la definición de varios componentes gráficos (`Component`) como ventanas (`Frame`), botones (`Button`) y casillas de verificación (`Checkbox`). En particular, las ventanas están compuestas por otros componentes gráficos anidados.

La biblioteca también define un mecanismo de manejo de eventos de GUI (`EventType`) como la presión de una tecla (`KEY_PRESSED`) o el click del mouse (`MOUSE_CLICKED`) sobre un componente, así como posibles errores (`ERROR`) que puedan ocurrir en un componente de la GUI. Cualquier otro objeto puede registrarse para ser notificado de los eventos que ocurren en un componente específico. Para ello, las clases deben implementar una interfaz (`ActionListener`) y agregarse como observadores de un componente. Los observadores serán notificados de los eventos que ocurran en el componente (`action`) a través de la operación definida para ello (`actionPerformed`).

Como ejemplo, en el digrama se observa una ventana definida por un usuario (`MyFrame`) que cuenta las veces que se hace click (atributo `clicks`) sobre algún botón de la ventana, por lo que, además de extender las propiedades de ventana, implementa la interfaz para recibir notificaciones.



A continuación, se describen el comportamiento de las operaciones definidas en el diagrama.

`Component::action(EventType): void`
Permite notificar de la ocurrencia de un evento del tipo indicado por parámetro, a todos los observadores de un componente.

`Component::addActionListener(ActionListener*): void`
Permite agregar un nuevo observador (`ActionListener`) a la lista de observadores del componente.

`Component::show(): void`
Permite dibujar un componente gráfico, lo que depende de cada componente en particular y de las propiedades que tenga definido (altura, color, posición, etc.)

`ActionListener::actionPerformed(EventType): void`
Permite que un observador reaccione a un evento ocurrido en un componente que esté observando. En el caso del tipo de ventanas definidas por usuario (`MyFrame`), si el evento fue un click del mouse (`MOUSE_CLICKED`), simplemente incrementa en 1 el contador de clicks, y si el evento fue un error (`ERROR`), debe cerrar la ventana eliminando todos los componentes que hayan sido creados y, luego, lanzar una excepción (`std::runtime_error`) con un mensaje. En otro caso, no hace nada.

`Frame::closeFrame(): void`
Permite cerrar una ventana, lo que consiste en eliminar todos los componentes (objetos) que hayan sido creados e incorporados a dicha ventana, así como eliminar la propia ventana.

`Frame::addComponent(Component*): void`
Permite agregar un nuevo componente a la ventana.

Se pide:

a. Implementar en C++ los .h del enumerado `EventType` y de las clases `ActionListener`, `Component` y `MyFrame`.
b. Implementar en C++ el .cpp de la clase `Component`, incluyendo constructor y destructor.
c. Implementar en C++ los métodos de las operaciones `Frame::closeFrame` y `MyFrame::actionPerformed`.
d. Implementar un main que permita hacer lo siguiente, con manejo de excepciones imprimiendo en pantalla el mensaje de error:
    - Crear un `MyFrame` *mf* que incluya un `Button` *bu*
    - Hacer que *mf* observe los eventos de *bu*
    - Generar un evento `MOUSE_CLICKED` sobre el botón *bu*
    - Generar un evento `ERROR` sobre el botón *bu*

Observaciones:

- Puede utilizar colecciones genéricas (realizaciones de `IDictionary` e `ICollection`) o paramétricas (contenedores STL).
- No incluir directivas al precompilador (#include, etc).
- No es necesario implementar setters y getters adicionales de las clases, salvo que se requieran en algún método.
- La implementación debe hacer un correcto manejo de memoria.