Programación 4

EXAMEN AGOSTO 2013

Por favor, siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba las hojas de un solo lado.
- 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 examen junto al examen.

Problema 1 (30 puntos)

a. Defina e indique el uso del "Tipo Asociativo". Ejemplifique.

b. Se desea construir un sistema para la reserva de hoteles en todo el mundo. Para ello interesará registrar reservas de hoteles en distintas ciudades, conociendo para cada uno de ellos: su nombre, dirección, teléfono, cantidad de estrellas y ciudad.

Cada hotel tiene habitaciones que se identifican por un número dentro del hotel, y se conoce la cantidad de camas que posee. Las habitaciones se categorizan en estándar, ejecutiva y superior. Para las superiores, se detalla en particular una descripción de las facilidades adicionales que brinda, la cual puede ser distinta para cada habitación de esta categoría.

Las reservas, identificadas por un código único, se realizan sobre las habitaciones registrando la fecha de inicio y fin de la estadía, modo de pago y número de pasaporte del cliente. No se deberán admitir superposiciones de reservas para una misma habitación de un hotel.

De cada ciudad se conoce su nombre, país y múltiples puntos de interés únicos de cada una. De estos últimos, se conoce su nombre, horario de visita y, si hubiese, su costo. Para facilitar la búsqueda a los usuarios, interesa registrar la distancia a la que dista cada hotel a cada uno de los puntos de interés de la ciudad a la que pertenece. También se brinda información sobre si el hotel ofrece algún descuento particular para determinado punto de interés, indicándolo en términos de porcentaje sobre el costo. En el caso que el cliente desee hacer uso de alguno de estos descuentos, interesa que quede asociado a su reserva de manera de realizar las gestiones correspondientes. Cada reserva podrá hacer uso de hasta 3 descuentos.

Se pide:

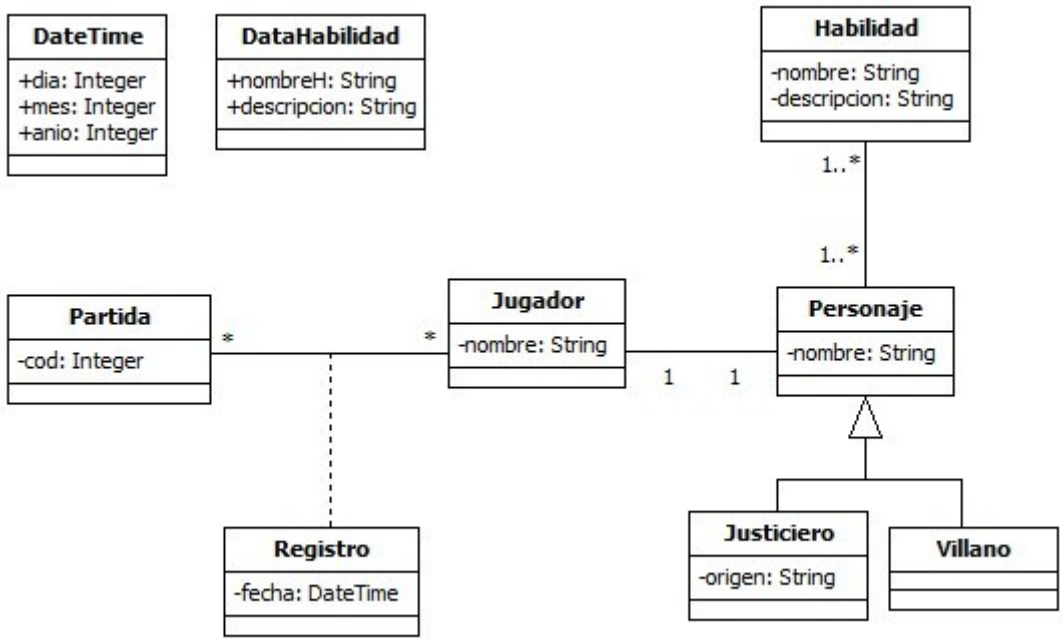
i. Modelar la realidad planteada mediante un Diagrama de Modelo de Dominio UML.

ii. Expresar todas las restricciones del modelo en **lenguaje natural.**

Problema 2 (35 puntos)

A usted y a su grupo de desarrollo de software se le ha encargado la construcción de un pequeño Sistema, que tiene la finalidad de llevar el registro de las distintas partidas de un juego específico. En estas partidas de juego (identificadas por un código) participan jugadores (identificados por su nombre) asociados a su personaje, interesando saber la fecha en que estos jugadores se registran como participantes de esas partidas.

Los personajes poseen un conjunto de habilidades que irán adquiriendo en el desarrollo del juego. Estos pueden ser justicieros o villanos, interesando en los justicieros su origen. Los Personajes y las Habilidades se identifican también por su nombre.



Su equipo de análisis ha construido el siguiente modelo de dominio:

Parte A

Se consideran los casos de uso "Crear Jugador" y "Obtener información de un Jugador", cada uno de los cuales es modelado con una sola operación del sistema, cuyos contratos se especifican a continuación.

crearNuevoJugadorEnPartida(codP**: Integer,** nomJ, nomP, tipo, origen**: String**, listdh**: set(DataHabilidad),** fActual**: DateTime)**	
Descripción	Ingresa un nuevo jugador con su personaje a una partida dada.
Parámetros	- codP: código de la Partida. - nomJ: nombre del nuevo Jugador. - nomP: nombre del Personaje. - tipo: puede tomar los valores "villano" o "justiciero". - origen: aplica si tipo="justiciero". - ldh: conjunto de habilidades iniciales del Personaje. - fActual: fecha actual del Sistema.
Precondiciones	- No existe en el sistema una instancia de Jugador con nombre igual a nomJ.

	- No existe en sistema una instancia de Personaje con nombre igual a nomP. - No existen en el sistema instancias de Habilidad con nombre igual a los atributos nombreH que vienen como parámetros en listdh. - Existe en el Sistema una instancia de PartidaDelJuego con código a cod.
Postcondiciones	- Existe en el Sistema una nueva instancia de Jugador cuyo atributo nombre es nomJ. - Se crea una instancia de tipo asociativo de Registro con fecha igual a fActual entre la Partida y el nuevo Jugador. - Existe en el Sistema un Personaje Villano o Justiciero. dependiendo de tipo, con los atributos nomP y origen si aplica. - Por cada data value dh en listdh, existe en el sistema una nueva instancia de Habilidad cuyos atributos se definen con los valores que tiene dh y se agrega a la colección de habilidades del Personaje. - Se crea un link entre el nuevo Jugador y su Personaje.

obtenerInfoJugador(nomJ **: String) : DataJugador**	
Descripción	Devuelve información de un jugador
Parámetros	- nomJ: nombre del jugador
Precondiciones	- Existe en el sistema una instancia de Jugador identificada por nomJ.
Postcondiciones	Se retorna un datavalue con la siguiente información: - Nombre del Jugador cuyo nombre es igual a nomJ. - Todos los códigos de las Partidas en las que participó incluyendo su fecha de registro en cada Partida. - El nombre de su Personaje asociado, incluyendo su origen si es Justiciero. (*) Obs: No es necesario mostrar las habilidades del Personaje

Se pide:

i. Realizar los Diagramas de Comunicación de las dos operaciones descritas, indicando el tipo de visibilidad en todos los mensajes y definiendo los nuevos DataTypes que puedan haber surgido, incluyendo DataJugador.

ii. Realizar el Diagrama de Clases de Diseño resultante.

Parte B

Se ha considerado la posibilidad de que las habilidades de los personajes puedan estar compuestas por varias de ellas, formando de esta manera una jerarquía. Por ejemplo la habilidad "Desplazamiento" está compuesta por la habilidad "Correr" y la habilidad "Caminar" y así sucesivamente.

Se pide:

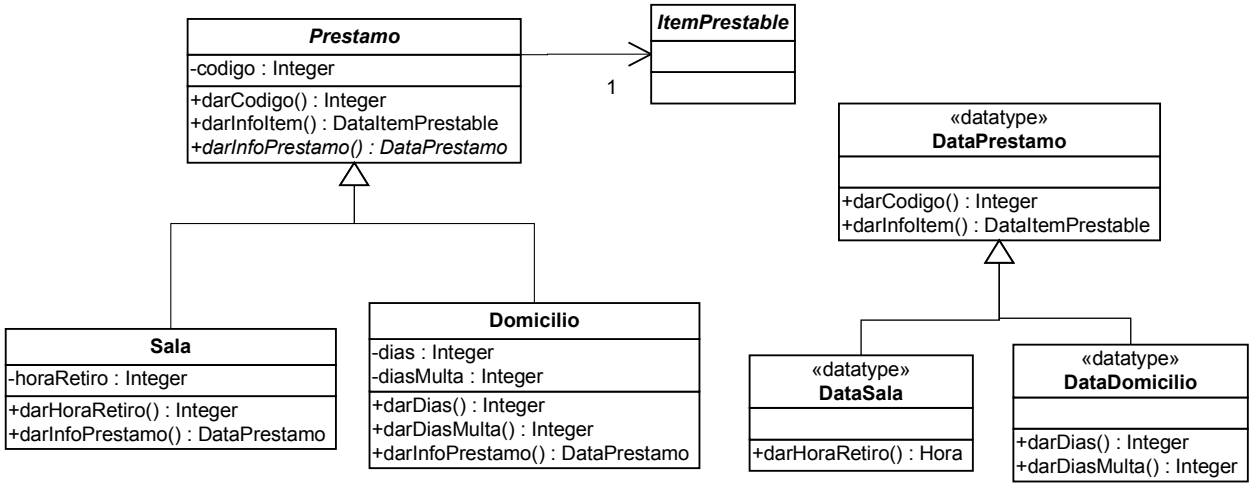
iii. Realizar las modificaciones necesarias al DCD realizadas en la parte ii, para diseñar este mecanismo.

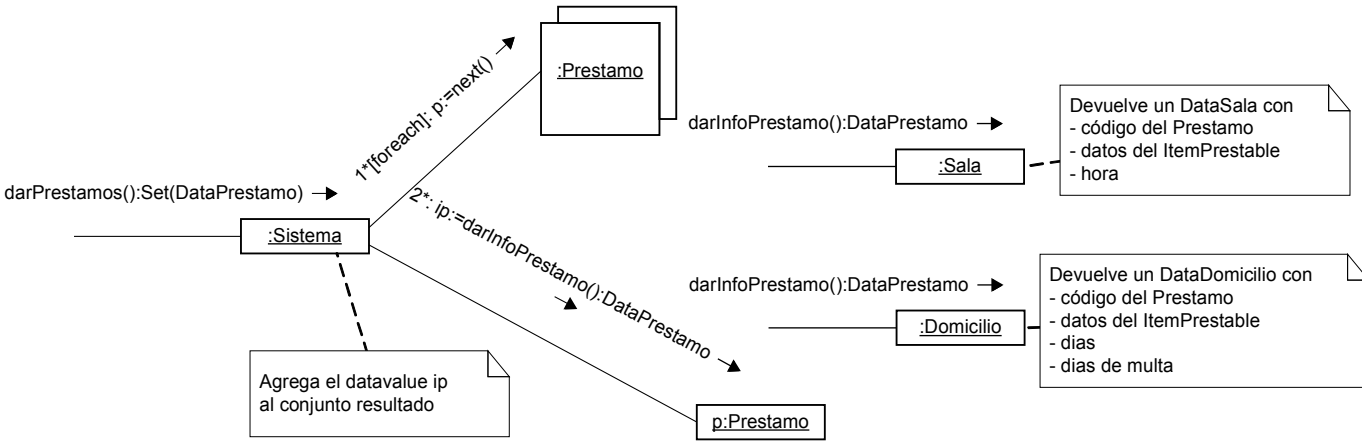
iv. En caso de haber utilizado un Patrón de diseño, identificarlo indicando los roles y las clases participantes.

Problema 3 (35 puntos)

Parte A

La biblioteca de su instituto ya realizó parte del diseño de un nuevo sistema de gestión de préstamos. La figura muestra un Diagrama de Clases de Diseño parcial, el cual contiene las clases relevantes para el subproblema aquí considerado. A su vez, el Diagrama de Comunicación presenta el diseño de la operación `darPrestamos`, que devuelve una lista de todos los préstamos existentes en el sistema.





Se pide:

i. Implementar las **declaraciones** en C++ de las clases `Prestamo` y `Sala`, y las de los datatypes `DataPrestamo` y `DataSala`.

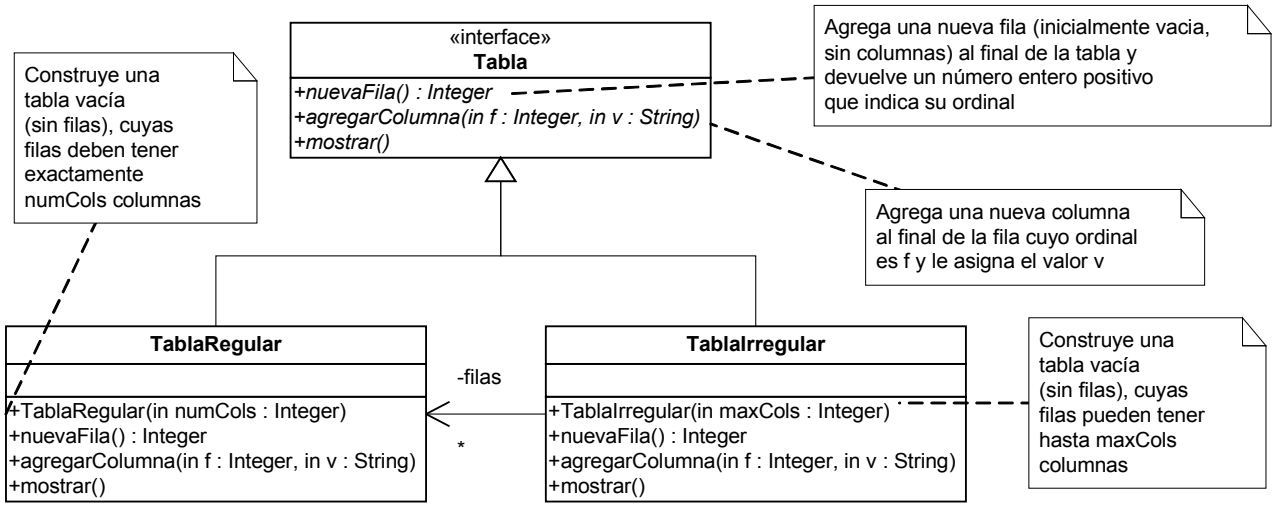
ii. Implementar los constructores y destructores de las clases y datatypes mencionados en la parte i, e implementar la operación `darInfoPrestamo` donde corresponda (excepto en la clase `Domicilio`).

iii. Implementar en C++ la operación `darPrestamos`.

Parte B

Con el objetivo de poder mostrar información en formato de tablas, un equipo de programación ha desarrollado la estructura que se muestra en la figura. La clase `TablaRegular` permite representar tablas rectangulares, cuya cantidad exacta de columnas se conoce al momento de

construir la tabla (todas las filas deben tener la misma cantidad de columnas). La operación `TablaRegular::mostrar()` lanza una excepción si para alguna fila de la tabla no se invocó la operación `agregarColumna` una cantidad de veces exactamente igual a la cantidad de columnas especificadas en el constructor (es decir, si en alguna fila falta completar alguna columna). Para poder tener filas con distintas cantidades de columnas en una misma tabla, el equipo ha diseñado la clase `TablaIrregular`, donde cada fila es una `TablaRegular`. Los usuarios de estas clases siempre lo hacen a través de una referencia a `Tabla`.



Se pide:

iv. Implementar en C++ completamente la clase `TablaIrregular` (excepto la operación `mostrar`), considerando que la misma tiene un diccionario de instancias de clase `TablaRegular` identificadas por el número de fila.

v. Indicar qué patrón de diseño se aplica, dando su nombre, clases participantes y roles de cada una.

Observaciones:

- No incluir directivas al precompilador.
- Puede suponer la existencia de implementaciones de `IDictionary`, `ICollection`, `IIterator` y `KeyInt` según sea necesario.
- Es posible utilizar las clases `set<T>` y `map<K,V>` de la STL.