Programación 4

EXAMEN JULIO 2018

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 examen junto al examen.
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial.

Problema 1 (35 puntos)

Una empresa especializada en la gestión de estacionamientos desea informatizar su sistema de reservas. Los estacionamientos se diferencian en dos tipos principales: propietario, que corresponden a estacionamientos pertenecientes a la empresa y por otro lado los estacionamientos cedidos, los cuales son propiedad de otras empresas (por ejemplo, hoteles) quienes tercerizan la gestión a la empresa especializada en gestión de estacionamientos. Todos los estacionamientos se identifican por un nombre único, además se conoce su dirección y capacidad. De los estacionamientos propietario se conoce si son techados o no. Por otro lado, de los estacionamientos cedidos se conoce la empresa que lo cede y la comisión acordada (%), y éstos a su vez se clasifican en dos tipos: edificios y playas.

De las playas de estacionamiento se conoce su área total, la cual puede ser utilizada como estacionamiento para vehículos durante eventos. De los eventos interesa el nombre y la fecha (día y hora) de comienzo, par que identifica a un evento ya que algunos eventos pueden repetir funciones en distintos días. De cada evento además se conoce el costo y su fecha (día y hora) de fin. Un evento puede utilizar varias playas de estacionamiento en simultáneo.

Los vehículos para los cuales se realizan reservas son identificados por su matrícula, además se conoce su tipo, que puede ser moto, auto, camioneta u otros que podrán agregarse luego. Para cada uno de estos tipos existe un precio por hora, diferente en cada uno de los estacionamientos.

Las reservas son para una unidad (no importando identificarla) dentro del estacionamiento y para un único vehículo. Las reservas se identifican por un código. Es de interés registrar el monto y fecha de pago de una reserva al momento de realizarla. Las reservas pueden ser por horas (horarias) o para un evento. Las horarias son para el uso del estacionamiento durante un rango de tiempo del cual interesa registrar día y hora habilitados para el ingreso y egreso. Las reservas para eventos no registran rango de uso como las diarias, ya que queda determinado por el comienzo y fin especificados por el evento.

Se relevó además el siguiente caso de uso:

Caso de Uso:	Alta de Reserva a Evento
Actor:	Usuario
Descripción:	El caso de uso comienza cuando el Usuario elige crear una nueva reserva para un evento ingresando el nombre del evento al que desea asistir. El sistema muestra un listado de todos los eventos con ese nombre y sus fechas (de comienzo) para que el usuario indique la fecha del evento al que desea asistir.

	Una vez identificado el evento, el sistema lista las playas de estacionamiento con capacidad disponibles y el rango de precios (min-max sin considerar el tipo de vehículo) para cada una. El usuario selecciona una playa, con lo cual el sistema lista el detalle de todos los precios disponibles por tipo de vehículo. Luego, el usuario puede: 1. elegir un tipo de vehículo para proceder con la reserva, ó 2. volver a seleccionar otra playa para repetir el proceso hasta que encuentre el precio que desea, ó 3. cancelar el caso de uso. En caso de proceder con la reserva, finalmente el usuario confirma su reserva indicando su matrícula y se termina el caso de uso.

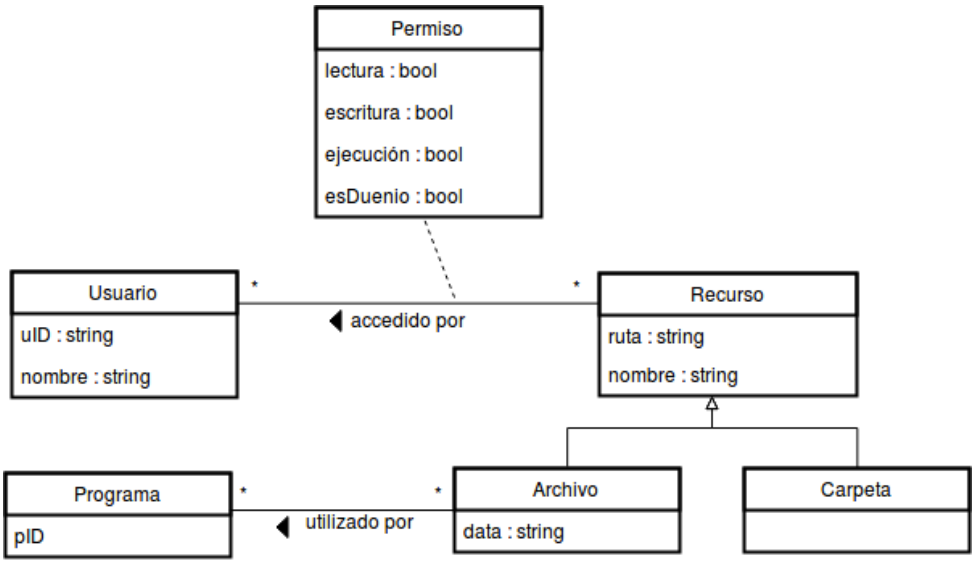
Se pide:

a) Realizar el Modelo de Dominio de la realidad planteada en la descripción y el caso de uso, con restricciones en lenguaje natural.

b) Realizar el DSS para el Caso de Uso Alta Reserva de Evento, indicando el uso de Memoria del Sistema así como de Datatypes.

<u>Problema 2</u> (35 puntos)

Una empresa de software desea crear un nuevo Sistema Operativo: Finglux. El sistema tendrá usuarios, recursos que pueden ser archivos o carpetas, y programas para los cuales se desea saber qué archivos utilizan o podrían utilizar durante su ejecución. Además, dado que será un sistema multiusuario, se desea saber a qué recursos pueden acceder los diferentes usuarios y con qué permisos se da dicho acceso. Además interesa registrar de qué recursos son dueños los usuarios, y qué recursos son utilizados por los diferentes programas de Finglux. A continuación se muestra el modelo de dominio del sistema a crear.



<u>Nota</u>: Un recurso puede ser identificado por la concatenación de su ruta y su nombre.

Se pide:

a) Teniendo en cuenta que el modelo de la figura es parcial y que interesa saber qué recursos contiene cada carpeta, completar el diagrama. ¿Es posible aplicar un patrón de diseño en su solución? En caso afirmativo, indicar qué problema general resuelve el patrón, así como los roles que cumple cada clase dentro de dicho patrón, para la realidad específica de este problema.

b) Tomando en cuenta lo agregado en la parte anterior, realizar el diagrama de comunicación de la siguiente operación, *incluyendo visibilidades*.

`eliminarArchivo(rutaArch: string, nombreArch: string, uID: string): bool`	
Descripción	El usuario de id ***uID*** elimina el archivo que se encuentra en la ruta ***rutaArch***, y tiene nombre ***nombreArch***.
Parámetros	***rutaArch***: La ruta en la que se encuentra el archivo en el sistema. ***nombreArch***: El nombre del archivo. ***uID***: El ID en el sistema del usuario.
Precondiciones	Existe en el sistema una instancia de Archivo con ruta ***rutaArch*** y nombre ***nombreArch***. Existe en el sistema un usuario con id ***uID***.
Postcondiciones	Si el usuario es dueño o tiene permiso de ejecución sobre el archivo: • Se destruyen los links entre los usuarios que acceden al archivo y el archivo. • Se destruyen los links entre los programas que utilizan el archivo y el archivo. • Se destruye el link entre el archivo y la carpeta en la que está. • Se destruye la instancia del archivo. • Se retorna **true**. De lo contrario se retorna **false**.

c) Realizar el Diagrama de Clases de Diseño (DCD) correspondiente a todas las partes anteriores.

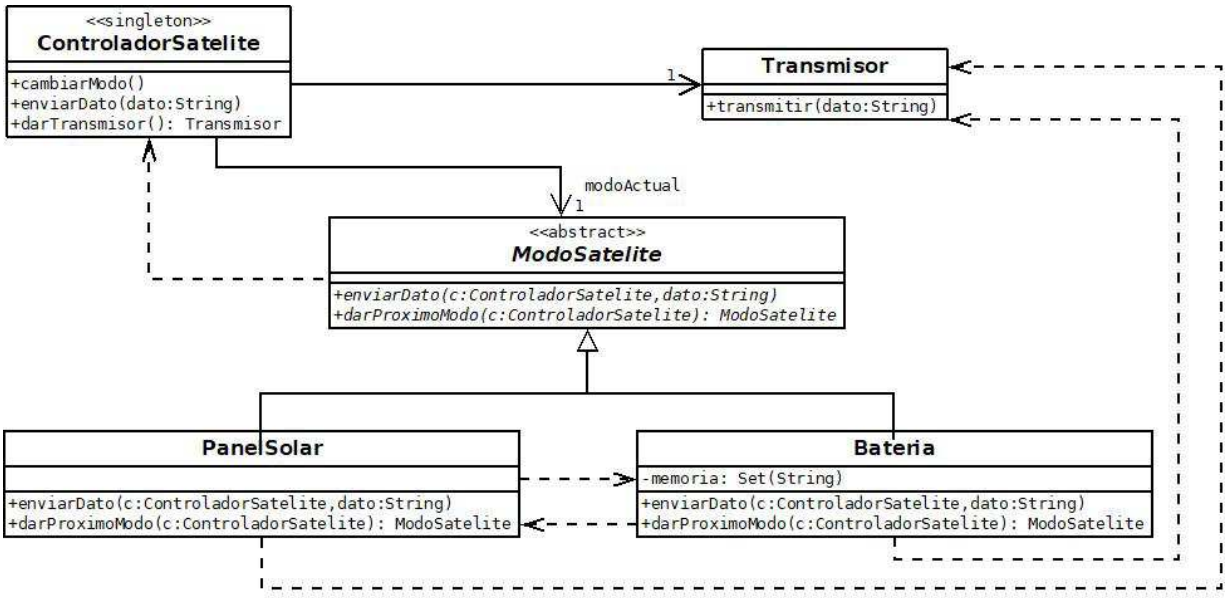
Notas:
- No incorporar setters ni getters al DCD.
- Incorporar en el DCD sólo los constructores o destructores que se utilicen en el diagrama de comunicación.

Problema 3 (30 puntos)

La figura muestra el Diagrama de Clases de Diseño (DCD) parcial del software de un satélite. El satélite toma periódicamente datos del espacio y los envía a tierra lo antes posible mediante el transmisor invocando la operación `enviarDato()` del controlador, la cual delega el envío a la clase `ModoSatelite`.

La mayor parte del tiempo el satélite recibe la energía a través de paneles solares instalados en el mismo, aunque en períodos de sombra la energía es suministrada por una batería auxiliar. Dependiendo de la fuente de energía que esté empleando, el satélite alterna entre dos modos de operación (modo `PanelSolar` y modo `Bateria`) invocando (automáticamente y desde fuera del sistema) la operación `cambiarModo()`, la cual delega también la obtención del próximo modo a la clase `ModoSatelite`. Inicialmente el satélite comienza su funcionamiento en el modo `Bateria`.

En el modo `PanelSolar` los datos recolectados son enviados inmediatamente a tierra, mientras que en modo `Bateria` son almacenados temporalmente en una memoria interna y no serán enviados hasta pasar nuevamente al modo `PanelSolar`.



A continuación se describen las operaciones de `PanelSolar` y `Bateria`:

Clase `PanelSolar`:

- `enviarDato`(): Envía el dato a tierra mediante el transmisor.
- `darProximoModo`(): Retorna una instancia del modo `Bateria`.

Clase `Bateria`:

- `enviarDato`(): Almacena temporalmente el dato agregándolo a la memoria del satélite.
- `darProximoModo`(): Envía todos los datos que estaban almacenados en la memoria y luego retorna una instancia del modo `PanelSolar`.

Por otra parte, la instanciación de la clase `ControladorSatelite` implica crear una instancia del transmisor y establecer el modo inicial en `Bateria`.

Se pide:

a) Implementar en C++ completamente los .h de todas las clases menos la de `Transmisor`.
b) Implementar en C++ completamente el .cpp de todas las clases menos la de `Transmisor`.

Observaciones:

- Puede utilizar colecciones genéricas (realizaciones de IDictionary e ICollection) o paramétricas (contenedores STL).
- Implementar exclusivamente las operaciones del DCD pedidas. Incluir constructores y destructor cuando sea necesario.
- No incluir directivas al precompilador.