PROGRAMACIÓN ORIENTADA A OBJETOS
Tercera Evaluación – I Término 2011-2012

NOMBRE: PARALELO:

Considere el siguiente problema:

Una empresa gestiona un conjunto de inmuebles, que administra en calidad de propietaria. Cada inmueble puede ser bien un local comercial o un departamento; o bien un edificio que a su vez tiene departamentos y locales. Como el número de inmuebles que la empresa gestiona no es un número fijo, la empresa propietaria exige que la aplicación permita tanto introducir nuevos inmuebles, con sus datos correspondientes (dirección, número, código postal y ciudad), así como eliminarlos del sistema, modificarlos y consultarlos.

Así mismo, que una empresa administre un edificio determinado no implica que gestione todos sus departamentos y locales, por lo que la aplicación también deberá permitir introducir nuevos departamentos o locales con sus datos correspondientes (numero de piso, numero de departamento, número de cuartos, edificio a que pertenece), eliminarlos, modificarlos y hacer consultas sobre ellos. Considere que para introducir un nuevo local o departamento al sistema, debe existir el edificio en donde está ubicado.

Cualquier persona que tenga un aval bancario, un contrato de trabajo o venga avalado por otra persona registrada en el sistema puede alquilar el edificio completo o alguno de los departamentos o locales que no estén ya alquilados. Por ello deberán poder registrarse, si son nuevos inquilinos, con sus datos correspondientes (Cédula, Nombres, Apellidos, Edad, Género, Usuario y Clave), poder modificarlos, darlos de baja, consultar, entre otros. Considerando que para la realización de cualquiera de estas operaciones es necesaria la identificación por parte del inquilino. Por otra parte, cada mes el secretario de la empresa pedirá la generación de un recibo para cada uno de los departamentos y de los locales, el cual lleva asociado un número de recibo que es único para cada departamento y para cada local y que no variará a lo largo del tiempo, indicando el departamento o local a que pertenece, la fecha de emisión, la renta, el IVA correspondiente y el total a pagar. Además, para cada recibo se desea saber si está o no cobrado.

Con vistas a facilitar la emisión de recibos cada mes, la aplicación deberá permitir la generación de recibos idénticos a los del mes anterior, a excepción de la fecha. La aplicación también deberá presentar los recibos en formato impreso. De igual forma, el secretario debe poder gestionar los movimientos bancarios que se producen asociados a cada edificio, departamento o local. Un movimiento bancario siempre estará asociado a un banco y a una cuenta determinada de ese banco. En esa cuenta existirá un saldo, acreedor o deudor, que aumentará o disminuirá con cada. Para cada movimiento se desea saber también la fecha en que se ha realizado. Un movimiento bancario puede ser de dos tipos: un gasto o un ingreso.

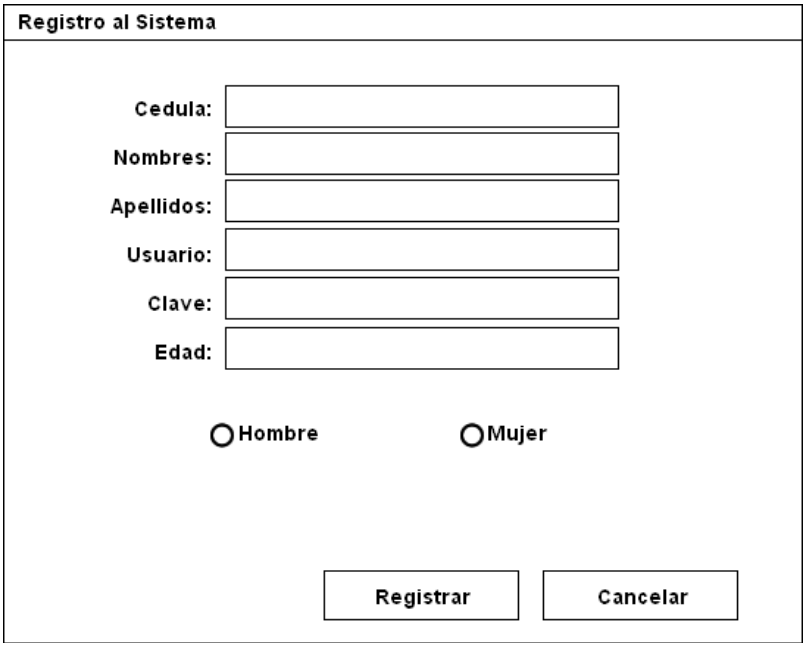
Si el movimiento bancario es un gasto, entonces estará asociado a un inmueble determinado, y se indicará el tipo de gasto al que pertenece entre los que se tienen estipulados. Ejemplos de gastos son el coste de la reparación de un ascensor del inmueble que pertenece a gastos de reparación, el sueldo del conserje, entre otros. Sí el movimiento bancario es un ingreso entonces estará asociado a un piso de un inmueble determinado o a un local en este caso los recibos que se cobran cada mes a los inquilinos.

Basándose en los gastos e ingresos que se deducen de los movimientos bancarios, la aplicación deberá ser capaz de ocuparse de la gestión económica generando los informes que facilitan la realización de la declaración de la renta.

Por último, la aplicación deberá ser capaz de proporcionar el acceso, de forma estructurada, a toda la información almacenada en el sistema, generando para ello los listados necesarios que requiere el secretario. Ejemplos de listado son: el listado de todo los inquilinos ordenado por fechas, el listado de inquilinos que han pagado o no en un determinado intervalo de tiempo, el listado de todos los inmuebles, el listado de todos los pisos y locales de cada edificio, el listado de todos los recibos pendientes de cobro en un determinado intervalo de tiempo, entre otros.

DESARROLLO

1. (10 PUNTOS) Realice el Diagrama de Casos de uso referente al problema relacionado al Actor Inquilino
2. Realice el Diagrama de Clases de la Aplicación completa. En el diagrama deberá reflejar
    a. (10 PUNTOS)Todas las clases que intervienen en el problema
    b. (8 PUNTOS)Todas las relaciones entre las clases
    c. (7 PUNTOS)Todos los atributos necesarios para cada clase
3. Considere la siguiente ventana de registro de un usuario



Considere la clase Usuario:

USUARIO
String cedula String apellidos String nombres String usuario String clave Integer edad GeneroEnum genero
TODOS LOS CONSTRUCTORES TODOS LOS GETTERS TODOS LOS SETTERS

a. (5 PUNTOS) Implemente el Enumerador **GeneroEnum** más adecuado.
b. Considere que toda la información de los usuarios del sistema se encuentra en el archivo usuarios.dat y la información se encuentra en el siguiente formato:

```
CEDULA,NOMBRES,APELLIDOS,USUARIO,CLAVE,EDAD,GENERO
```

   i. (5 PUNTOS)Implemente una excepción personalizada llamada **UserNotFoundException** que será lanzada en un literal posterior cuando el usuario y clave no coinciden con los registros del archivo de usuarios. El mensaje de la Excepción deberá decir "Usuario y/o clave incorrecto"

Implemente las siguientes funciones:

   ii. (5 PUNTOS) **bool cargaUsuarios(ArrayList<Usuario> usuarios)** que lee la información de TODOS los usuarios en el sistema y lo almacena en el arreglo usuarios previamente inicializado.
   iii. (10 PUNTOS) **Usuario verificaUsuarioClave(String usuario, String clave) throws UserNotFoundException** , la cual retorna un objeto de tipo Usuario que corresponde al usuario y clave recibido como parámetro. La función lanza la excepción si el usuario y clave no existen. Considere que toda la información va a ser obtenida del archivo de usuarios y no de una lista.
   iv. (5 PUNTOS)**bool grabaUsuarios(ArrayList<Usuario> usuarios, String nombre_archivo)** que graba la información de los usuarios en el archivo con el nombre que recibe como segundo parámetro. La función deberá retornar true si el proceso de grabado se realizó con éxito y false si no lo fue.

c. Implemente la clase **RegistroView** considerando lo siguiente
   i. (10 PUNTOS) Colocar todos los atributos gráficos y no gráficos. Considere la lista de usuarios como atributos de la clase.
   ii. (10 PUNTOS) Dibujar todos los componentes.
   iii. (10 PUNTOS) Considerar TODOS los listeners necesarios para poder utilizar las acciones de los botones Registrar y Cancelar.
   iv. (5 PUNTOS) Cree la clase interna para el manejo del listener que considera las acciones sobre los botones. Al hacer clic en el botón registrar se deberá agregar el usuario en la lista y enviar a grabar toda la lista en el archivo.