|  |  |
| --- | --- |
| **Examen Ordinaria**  **Programación II** | |
| 2018-2019 | |
|  |  |

En Vitoria a 31 de Mayo de 2019 Profesor: Javier Cerro

1. Contexto

Una inmobiliaria comercializa viviendas, y quiere realizar una gestión de las que tiene en cartera.

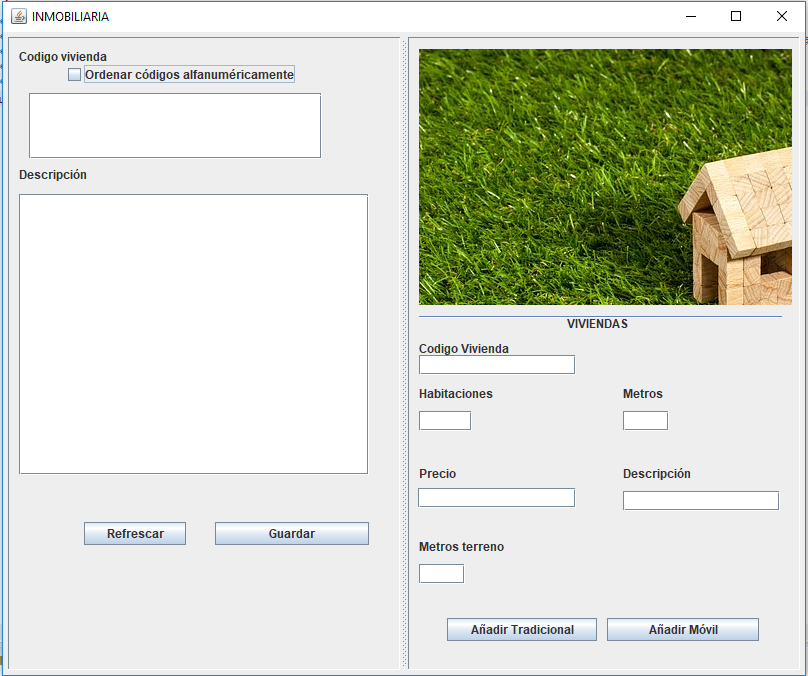


Ilustración Interfaz de usuario de la aplicación

En la parte izquierda se realiza la visualización de los viviendas introducidas, mientras que en la derecha se introducen los datos de **las viviendas.** En la aplicación se contemplan dos tipos de viviendas. Las tradicionales y las móviles.

Los datos requeridos para las viviendas **tradicionales** son los siguientes:

* Código vivienda: código alfanumérico y **único** de la vivienda. En BD Varchar(45).
* Precio: Precio de la vivienda. En BD INT.
* Descripción: descripción de la vivienda. En BD VARCHAR(1000).
* Habitaciones: Número de habitaciones de la vivienda. En BD INT
* Baños: Número de baños de la vivienda. En BD INT.
* Metros: Metros útiles que tiene la vivienda. En BD INT
* Al tura: Piso de la vivienda (0,1,2,3,4…). En BD INT

En el caso de la vivienda **móvil**, además de esas propiedades, también añade los metros de terreno en los que está depositada:

* Metros Terreno: Metros de terreno que la vivienda tiene o necesita para ser instalada.En BD INT.

En la aplicación una vivienda **se distinguirá por código de vivienda. Además el código de vivienda no se puede repetir en ningún objeto de la aplicación. Es decir, no puede haber una vivienda tradicional y una móvil con el mismo código.**

**Ejercicio 1 (2 ptos)**

Diseña la estructura de clases necesaria para contener la información que tiene que gestionar la aplicación, junto con los métodos hashcode e equals necesarios para que los objetos se distingan de forma unívoca en base a los atributos definidos, y para que estos puedan ser guardados en base de datos.

Además distribuye tu aplicación en paquetes en base a tu criterio.

**Ejercicio 2 (2 ptos)**

Programar los botones “Añadir Tradicional” y “Añadir Móvil” de tal forma que se recojan los valores necesarios para dar de alta cada tipo de vivienda. El alta de vivienda supondrá tener el objeto exclusivamente en memoria sin guardarlo en base de datos.

**Ejercicio 3 (1 ptos)**

Programa el botón “Guardar BD” para que se guarde en fichero los objetos que hasta ese momento están en memoria.

**Ejercicio 4 (1 ptos)**

Programa el botón “Refrescar” de tal forma que en la lista superior de la parte izquierda aparezcan todos los códigos de todas las viviendas que hay en el sistema, es decir, tanto los que están en memoria como los que están en BD, sin duplicados. Si al juntar las viviendas de memoria y las de BD existiera algún duplicado, el sistema descartaría uno de los dos, sin importar cuál se borra.

Para rellenar la lista con los códigos de los electrodomésticos, puedes utilizar el siguiente método:

private void CargarListaCodigos(String [] codigos)

**Ejercicio 5(1,5 ptos)**

Al pulsar el botón “Refrescar”, si el checkbox “Ordenar códigos alfanuméricamente” está marcado, debes ordenar los objetos por dicho código alfanumérico, de forma alfabética.

Para saber si el checkbox está activado puedes utilizar la siguiente función:

private boolean getEstadoCheck()

**Ejercicio 6(1,5 ptos)**

En algún punto de la aplicación, añade una excepción en caso de que se dé algún duplicado. Se eliminará uno de los objetos por duplicidad, mostrando además cuál es el código que se ha encontrado duplicado.

**Ejercicio 7(1 ptos)**

Cada vez que el usuario seleccione de la lista de códigos uno de los códigos, en el área de “Descripción” deberá aparecer la descripción completa del electrodoméstico, usando para esto una interfaz de tipo itfProperty.(No se da por bueno usar el método toString). El evento que se lanza al seleccionar un elemento de la lista ya está programado, y es el siguiente método:

public void valueChanged(ListSelectionEvent e)