Proyecto MP

Clase habitaciones

La clase habitaciones es la clase padre de los tres tipos de habitaciones que tenemos:

- Sala de estar
- Dormitorio
- Cocina

```
public class Habitaciones {

protected String nombre;

protected DispositivosInteligentes[] dispositivosInteligentes;

protected int numDispositivosInteligentes;



protected Habitaciones() {

}

protected Habitaciones(String nombre, int maxDispositivosInteligentes) {

this.nombre = nombre;

this.dispositivosInteligentes = new
DispositivosInteligentes[maxDispositivosInteligentes];

this.numDispositivosInteligentes = 0;

}



public void addDispositivosInteligentes(DispositivosInteligentes dispositivo) {

if (numDispositivosInteligentes < dispositivosInteligentes.length) {

dispositivosInteligentes[numDispositivosInteligentes] = dispositivo;

numDispositivosInteligentes++;

} else {
```

```
System.out.println("No se pueden agregar más dispositivos inteligentes.");

}
}


public void removeDispositivosInteligentes(DispositivosInteligentes dispositivo) {

for (int i = 0; i < numDispositivosInteligentes; i++) {

if (dispositivosInteligentes[i] == dispositivo) {

for (int j = i; j < numDispositivosInteligentes - 1; j++) {

dispositivosInteligentes[j] = dispositivosInteligentes[j + 1];

}

dispositivosInteligentes[numDispositivosInteligentes - 1] = null;

numDispositivosInteligentes--;

return;

}

}

System.out.println("El dispositivo inteligente no se encontró en la habitación.");

}


public DispositivosInteligentes[] getDispositivosInteligentes() {

return dispositivosInteligentes;

}


public String getNombre() {

System.out.println(nombre);

return nombre;

}
```

```
}
```

Tiene dos constructores que tienen un nivel de visibilidad "protected" por razones que tienen que ver con la herencia. Determinamos poner protected para evitar cualquier problema al momento de heredar.

Los atributos son:

- String nombre: la cual representa el nombre
- Dispositivos inteligentes[]: Es un array de elementos Dispositivosinte. Tiene el tamaño que se asigna por el usuario al crear la habitación
- numDispositivosInte: Representa la cantidad de dispositivos inteligentes que tiene la clase como tal, se aumenta y disminuye en los métodos add y remove.

Los dos métodos más difíciles de comprender a mi parecer son los siguientes:

removeDispositivosInteligentes

```
public void removeDispositivosInteligentes(DispositivosInteligentes dispositivo) {

for (int i = 0; i < numDispositivosInteligentes; i++) {

if (dispositivosInteligentes[i] == dispositivo) {

for (int j = i; j < numDispositivosInteligentes - 1; j++) {

dispositivosInteligentes[j] = dispositivosInteligentes[j + 1];

}

dispositivosInteligentes[numDispositivosInteligentes - 1] = null;

numDispositivosInteligentes--;

return;

}

}
```

Este método tiene la función de eliminar un dispositivo inteligente.

Lo hace de la siguiente manera:

1. Creando el bucle que comprende el tamaño de dispositivos inteligentes que tenemos en la casa inteligente.

2. En cada iteración comprobamos si el dispositivo es el mismo que el que queremos eliminar. Si encontramos el dispositivo:
    1. Utilizando un bucle for que inicia en la posición la que se encontro el dispositivo que queremos eliminar y se ejecuta hasta el numdispositivosinteligentes - 1. O sea, desde nos quedamos hasta el final de la lista.
    2. Copiamos el elemento de la posición j +1 (un elemento más adelante del que estamos en cada iteración) y lo copiamos al elemento j, es decir, al elemento en el que estamos actualmente.
    3. Al terminar el ciclo establecemos dispositivo con numero numdispositivosinteligentes -1 en null, para indicar que esta vacio
    4. Reducimos la cantidad de dispositivos y acaba el proceso

Todo eso veamoslo así:

Supongamos que tenemos una lista de dispositivos inteligentes en una habitación y queremos eliminar el dispositivo en la posición 3. La lista se ve de la siguiente manera:

1. Dispositivo A
2. Dispositivo B
3. Dispositivo C (el que queremos eliminar)
4. Dispositivo D
5. Dispositivo E

Ahora, si ejecutamos el método `removeDispositivosInteligentes` con el dispositivo C como argumento, aquí está cómo funciona:

1. El bucle `for` exterior recorre la lista de dispositivos desde la posición 0 hasta la posición 4 (5 dispositivos en total).
2. Cuando `i` es igual a 2 (posición 3 en términos de índices), se verifica si `dispositivosInteligentes[2]` (que es el "Dispositivo C") es igual al dispositivo que queremos eliminar (el "Dispositivo C").
3. Se encuentra una coincidencia, y se procede a eliminar el "Dispositivo C". El bucle `for` anidado se activa para mover los elementos posteriores un paso hacia atrás:
    - En la primera iteración del bucle anidado, "Dispositivo D" se mueve a la posición 2 (donde estaba "Dispositivo C").
    - En la segunda iteración del bucle anidado, "Dispositivo E" se mueve a la posición 3 (donde estaba "Dispositivo D").
    - Luego, el último elemento "Dispositivo E" se establece en `null` para eliminar la referencia a él.
    - Finalmente, se decrementa el contador `numDispositivosInteligentes` en 1, lo que lo lleva a 4.

Después de ejecutar este método, la lista de dispositivos inteligentes se verá así:

1. Dispositivo A
2. Dispositivo B

3. Dispositivo D
4. Dispositivo E

El "Dispositivo C" ha sido eliminado, y la lista se ha reorganizado para llenar el espacio vacío.

addDispositivoInteligente

```
public void addDispositivosInteligentes(DispositivosInteligentes dispositivo) {

if (numDispositivosInteligentes < dispositivosInteligentes.length) {

dispositivosInteligentes[numDispositivosInteligentes] = dispositivo;

numDispositivosInteligentes++;

} else {

System.out.println("No se pueden agregar más dispositivos inteligentes.");

}

}
```

La función sirve para agregar un dispositivo a la lista de dispositivos inteligentes.

- Primero verificamos si la longitud de la lista es menor que la cantidad de dispositivos inteligentes, resulta ser cierto. Esto pasa porque a la lista se le da un tamaño especifico desde antes de si quiera tener dispositivos inteligentes en ella
  - Si eso pasa, al elemento numdispositivointeligente, el cual demuestra la cantidad de dispositivos inteligentes que hay, + 1 se le asigna el dispositivos
  - Se agrega uno al contador
- Si da negativo, solo imprimimos en la terminal que no se pueden agregar más dispositivos

Cocina

```
public class Cocina extends Habitaciones {



public Cocina(String nombre, int maxDispositivosInteligentes) {

super(nombre, maxDispositivosInteligentes);

}

}
```

Hereda todos los métodos y atributos de su padre sin modificar nada.

Dormitorio

```
import java.util.ArrayList;



public class Dormitorio extends Habitaciones {

public Dormitorio(String nombre, int maxDispositivosInteligentes) {

super(nombre, maxDispositivosInteligentes);

}


}
```

Saladeestar

```
import java.util.ArrayList;



public class Saladeestar extends Habitaciones {

public Saladeestar(String nombre, int maxDispositivosInteligentes) {

super(nombre,maxDispositivosInteligentes);

}



}
```

Dispositivos inteligentes

```
import javax.swing.JOptionPane;

import javax.swing.ImageIcon;

public class DispositivosInteligentes {

//Atributos
```

```
protected int idDispositivo;

protected String nombre;

protected boolean encendido;

//Constructores

ImageIcon encender =new ImageIcon("src/proyecto/encendido.png");

ImageIcon apagar =new ImageIcon("src/proyecto/apagado.jpeg");

int iD,ID;

public DispositivosInteligentes(int id, String nombre, boolean encendido) {

this.idDispositivo = id;

this.nombre = nombre;

this.encendido = false;

}

//Metodos

public void encender() {

encendido = true;

JOptionPane.showMessageDialog(null, nombre+" encendido.", "Jaime, Navarro, Perez y
Soto", JOptionPane.DEFAULT_OPTION, encender);

}

public void apagar() {

encendido = false;

JOptionPane.showMessageDialog(null, nombre+" apagado.", "Jaime, Navarro, Perez y
Soto", JOptionPane.DEFAULT_OPTION, apagar);

}

public boolean obtenerEstado() {

return encendido;

}
```

```
public void establecerEstado(boolean estado) {

encendido = estado;

}

public int getId() {

return idDispositivo;



}



public String getNombre() {

return nombre;

}

}
```

Todos los métodos de esta clase son realmente sencillos y se pueden dividir en métodos de dos tipos:

- Métodos que cambian estados
  - encender: Solo cambia el atributo encendido a true y muestra una ventana que lo indicada
  - apagar: cambia el estado de encendido a false y muestra en la ventana
- Métodos que retornan valores
  - getid: retorta el id del dispositivos
  - getnombre: devuelve el nombre del dispositivo

La particularidad viene cuando tenemos las clases hijas. Las podemos dividir en dos tipos:

- Heredan métodos y no agregan nada relevante más que modificaciones a los de la clase Padre.
  - AireAcondicionado
  - Cafetera
  - Horno
  - LucesInte
  - SeguridadInte

Todos estos solo modifican una pequeña parte que tiene que ver con encendido y apagado.

Termostado es el otro tipo de clase. Explicaremos el código paso a paso:

Termostato

```
import javax.swing.ImageIcon;

import javax.swing.JOptionPane;


public class TermostatoInte extends DispositivosInteligentes {

private int temperatura;

private ImageIcon iconoTemperatura = new ImageIcon("src/proyecto/temperatura.png");


public TermostatoInte(int id, String nombre, boolean estado, int temperatura) {

super(id, nombre, estado);

this.temperatura = temperatura;

}


public int getTemperatura() {

return temperatura;

}


public void setTemperatura(int temperatura) {

this.temperatura = temperatura;

}


@Override

public void encender() {

super.encender();

JOptionPane.showMessageDialog(null, "Temperatura deseada de " + this.getNombre() + "
```

```
establecida a " + this.temperatura + "°C", "Casa Inteligente",
JOptionPane.DEFAULT_OPTION, iconoTemperatura);

}



@Override

public void apagar() {

super.apagar();

JOptionPane.showMessageDialog(null, this.getNombre() + " ha sido apagado", "Casa
Inteligente", JOptionPane.DEFAULT_OPTION, iconoTemperatura);

}

}
```

Heredamos los métodos encender y apagar. Las cuales solo sonn modificadas en el mensaje que dan.

Agregamos los siguientes métodos:

```
public void setTemperatura(int temperatura) {

this.temperatura = temperatura;

}
```

Solo asignamos la temperatura indicada a la temperatura del método

```
public int getTemperatura() {

return temperatura;

}
```

Regresa la temperatura.

Casa inteligente

```
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class CasaInteligente {
```

```
public static void main(String[] args) {

ArrayList<Habitaciones> listaCuartos = new ArrayList<>();

ArrayList<DispositivosInteligentes> listaDispTot = new ArrayList<>();

int n;



do {

n = mostrarMenuPrincipal();

switch (n) {

case 1:

ingresarCuartos(listaCuartos);

break;

case 2:

ingresarDispositivos(listaCuartos, listaDispTot);

break;

case 3:

encenderDispositivo(listaDispTot);

break;

case 4:

apagarDispositivo(listaDispTot);

break;

case 5:

mostrarDispositivosEnHabitaciones(listaCuartos);

break;

case 6:

eliminarDispositivoDeHabitacion(listaCuartos, listaDispTot);
```

```
break;

case 7:

JOptionPane.showMessageDialog(null, "Saliendo del programa. Hasta luego.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

break;

default:

JOptionPane.showMessageDialog(null, "Opción no válida. Por favor, seleccione una
opción válida.", "Casa Inteligente", JOptionPane.WARNING_MESSAGE);

}



} while (n !=7);

}



public static int mostrarMenuPrincipal() {

String[] MenuPpal = {"1. Ingresar cuarto", "2. Ingresar dispositivos a conectar", "3.
Encender dispositivos", "4. Apagar dispositivos", "5. Mostrar dispositivos en
habitaciones","6. Eliminar dispositivo", "7. Salir"};



return JOptionPane.showOptionDialog(null, "Seleccione lo que guste realizar: ", "Casa
Inteligente", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null,
MenuPpal, MenuPpal[0]) + 1;

}



public static void mostrarDispositivosEnHabitaciones(ArrayList<Habitaciones>
listaCuartos) {

if (listaCuartos.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay cuartos registrados para mostrar
dispositivos.", "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;
```

```
}

StringBuilder mensaje = new StringBuilder("Dispositivos en cada habitación:\n");

for (Habitaciones cuarto : listaCuartos) {

mensaje.append(cuarto.getNombre()).append(":\n");

DispositivosInteligentes[] dispositivos = cuarto.getDispositivosInteligentes();

for (DispositivosInteligentes dispositivo : dispositivos) {

if (dispositivo != null) {

mensaje.append("Nombre: ").append(dispositivo.getNombre()).append(", Estado: ");

mensaje.append(dispositivo.obtenerEstado() ? "Encendido" : "Apagado").append(", ");

if (dispositivo instanceof TermostatoInte) {

mensaje.append("Temperatura: ").append(((TermostatoInte)
dispositivo).getTemperatura()).append("°C, ");

}

mensaje.append("\n");

}

}

mensaje.append("\n");

}
```

```
JOptionPane.showMessageDialog(null, mensaje.toString(), "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

}



public static void ingresarCuartos(ArrayList<Habitaciones> listaCuartos) {

String[] TipoCuarto = {"Sala de estar", "Cocina", "Dormitorio"};



String nombreCuarto = JOptionPane.showInputDialog(null, "Por favor, ingrese el nombre
del cuarto: ", "Casa Inteligente", JOptionPane.QUESTION_MESSAGE);

int tipoCuarto = mostrarMenuTipoCuarto(TipoCuarto);



if (tipoCuarto >= 0) {

int maxDispositivos = verNumero("Ingrese la cantidad máxima de dispositivos
inteligentes para esta habitación: ");

ingresarCuarto(listaCuartos, tipoCuarto, nombreCuarto, maxDispositivos);

}

}



public static int mostrarMenuTipoCuarto(String[] tipos) {

String message = "¿Qué tipo de cuarto es? Seleccione una opción:";

String title = "Casa Inteligente";

return JOptionPane.showOptionDialog(null, message, title, JOptionPane.DEFAULT_OPTION,
JOptionPane.QUESTION_MESSAGE, null, tipos, tipos[0]);

}



public static void ingresarCuarto(ArrayList<Habitaciones> listaCuartos, int
tipoCuarto, String nombreCuarto, int maxDispositivos) {
```

```
Habitaciones cuarto = null;


switch (tipoCuarto) {

case 0:

cuarto = new Saladeestar(nombreCuarto, maxDispositivos);

break;

case 1:

cuarto = new Cocina(nombreCuarto, maxDispositivos);

break;

case 2:

cuarto = new Dormitorio(nombreCuarto, maxDispositivos);

break;

}


if (cuarto != null) {

listaCuartos.add(cuarto);

JOptionPane.showMessageDialog(null, "El cuarto " + cuarto.getNombre() + " se ha
registrado con éxito.", "Casa Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

}


public static void ingresarDispositivos(ArrayList<Habitaciones> listaCuartos,
ArrayList<DispositivosInteligentes> listaDispTot) {

if (listaCuartos.isEmpty()) {

JOptionPane.showMessageDialog(null, "ERROR: Registre primero un cuarto para poder
hacer esta acción.", "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;
```

```
}



String[] subclases = {"Aire Acondicionado", "Cafetera", "Horno", "Luces
Inteligentes", "Termostato Inteligente", "Seguridad Inteligente"};



int seleccionSubclase = JOptionPane.showOptionDialog(

null, "Seleccione la subclase de dispositivo inteligente:", "Casa Inteligente",
JOptionPane.DEFAULT_OPTION,

JOptionPane.QUESTION_MESSAGE, null, subclases, subclases[0]);



if (seleccionSubclase >= 0) {

String subclaseSeleccionada = subclases[seleccionSubclase];



DispositivosInteligentes dispositivo = null;

int idDispositivo = verNumero("Ingrese el ID del dispositivo o presione 0 para
asignar automáticamente: ");

String nombreDispositivo = JOptionPane.showInputDialog(null, "Por favor, ingrese el
nombre del dispositivo: ", "Casa Inteligente", JOptionPane.QUESTION_MESSAGE);



String[] habitacionesDisponibles = new String[listaCuartos.size()];

for (int i = 0; i < listaCuartos.size(); i++) {

habitacionesDisponibles[i] = listaCuartos.get(i).getNombre();

}

String habitacionElegida = (String) JOptionPane.showInputDialog(null, "Seleccione la
habitación a la que desea asignar el dispositivo:", "Casa Inteligente",
JOptionPane.QUESTION_MESSAGE, null, habitacionesDisponibles,
habitacionesDisponibles[0]);
```

```
if (habitacionElegida != null) {

for (Habitaciones cuarto : listaCuartos) {

if (cuarto.getNombre().equals(habitacionElegida)) {

if (subclaseSeleccionada.equals("Termostato Inteligente")) {

int temperatura = verNumero("Ingrese la temperatura deseada: ");

dispositivo = new TermostatoInte(idDispositivo, nombreDispositivo, false,
temperatura);

} else {

dispositivo = new DispositivosInteligentes(idDispositivo, nombreDispositivo, false);

}



if (dispositivo != null) {

cuarto.addDispositivosInteligentes(dispositivo);

listaDispTot.add(dispositivo);

if (dispositivo instanceof TermostatoInte) {

JOptionPane.showMessageDialog(null, "El dispositivo " + dispositivo.getNombre() + "
se ha registrado con éxito en la habitación " + cuarto.getNombre() + " con
temperatura establecida en " + ((TermostatoInte) dispositivo).getTemperatura() +
"°C.", "Casa Inteligente", JOptionPane.INFORMATION_MESSAGE);

} else {

JOptionPane.showMessageDialog(null, "El dispositivo " + dispositivo.getNombre() + "
se ha registrado con éxito en la habitación " + cuarto.getNombre(), "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

}

}

}

}
```

```
}

}



public static void encenderDispositivo(ArrayList<DispositivosInteligentes>
listaDispTot) {

if (listaDispTot.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay dispositivos registrados para encender.",
"Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a encender: ");



for (DispositivosInteligentes dispositivo : listaDispTot) {

if (dispositivo.getId() == idDispositivo) {

dispositivo.encender();

if (dispositivo instanceof TermostatoInte) {

int nuevaTemperatura = verNumero("Ingrese la nueva temperatura deseada: ");

((TermostatoInte) dispositivo).setTemperatura(nuevaTemperatura);

JOptionPane.showMessageDialog(null, "El dispositivo ha sido encendido y la
temperatura se ha establecido en " + nuevaTemperatura + "°C.", "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

} else {

JOptionPane.showMessageDialog(null, "El dispositivo ha sido encendido.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

return;

}
```

```
}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo, "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

}



public static void apagarDispositivo(ArrayList<DispositivosInteligentes>
listaDispTot) {

if (listaDispTot.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay dispositivos registrados para apagar.",
"Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a apagar: ");



for (DispositivosInteligentes dispositivo : listaDispTot) {

if (dispositivo.getId() == idDispositivo) {

dispositivo.apagar();

JOptionPane.showMessageDialog(null, "El dispositivo ha sido apagado.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

return;

}

}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo, "Casa Inteligente", JOptionPane.ERROR_MESSAGE);
```

```
}


public static int verNumero(String mensaje) {

int numero = 0;

boolean esNumeroValido = false;



do {

String input = JOptionPane.showInputDialog(null, mensaje, "Casa Inteligente",
JOptionPane.QUESTION_MESSAGE);



try {

numero = Integer.parseInt(input);

esNumeroValido = true;

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(null, "Por favor, ingrese un número válido.", "Casa
Inteligente", JOptionPane.WARNING_MESSAGE);

}



} while (!esNumeroValido);



return numero;

}



public static void eliminarDispositivoDeHabitacion(ArrayList<Habitaciones>
listaCuartos, ArrayList<DispositivosInteligentes> listaDispTot) {

if (listaCuartos.isEmpty() || listaDispTot.isEmpty()) {
```

```
JOptionPane.showMessageDialog(null, "No hay cuartos registrados o dispositivos
registrados para realizar esta acción.", "Casa Inteligente",
JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a eliminar de la
habitación: ");



for (Habitaciones cuarto : listaCuartos) {

for (DispositivosInteligentes dispositivo : cuarto.getDispositivosInteligentes()) {

if (dispositivo != null && dispositivo.getId() == idDispositivo) {

cuarto.removeDispositivosInteligentes(dispositivo);

listaDispTot.remove(dispositivo);

JOptionPane.showMessageDialog(null, "El dispositivo ha sido eliminado de la
habitación " + cuarto.getNombre(), "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

return;

}

}

}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo + " en ninguna habitación.", "Casa Inteligente",
JOptionPane.ERROR_MESSAGE);

}

}
```

Esta es la clase principal y la encargada de correr el programa. Tiene muchos métodos que en si solo representan menú, el cual fue dividido en métodos para mejorar su organización y manejo.

Main

```
public static void main(String[] args) {

ArrayList<Habitaciones> listaCuartos = new ArrayList<>();

ArrayList<DispositivosInteligentes> listaDispTot = new ArrayList<>();

int n;



do {

n = mostrarMenuPrincipal();

switch (n) {

case 1:

ingresarCuartos(listaCuartos);

break;

case 2:

ingresarDispositivos(listaCuartos, listaDispTot);

break;

case 3:

encenderDispositivo(listaDispTot);

break;

case 4:

apagarDispositivo(listaDispTot);

break;

case 5:

mostrarDispositivosEnHabitaciones(listaCuartos);

break;

case 6:
```

```
eliminarDispositivoDeHabitacion(listaCuartos, listaDispTot);

break;

case 7:

JOptionPane.showMessageDialog(null, "Saliendo del programa. Hasta luego.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

break;

default:

JOptionPane.showMessageDialog(null, "Opción no válida. Por favor, seleccione una
opción válida.", "Casa Inteligente", JOptionPane.WARNING_MESSAGE);

}



} while (n !=7);

}
```

Este es el menú, el cual funciona mediante un do while que contiene un switch el cual muestras las operaciones que podemos efectuar en el proyecto.

Lo más rescatable es que creamos dos arrarys. Uno que contiene los dispositivos inteligentes y otro de las habitaciones. El primero se usa principalmente para efectuar las operaciones encender y apagar de manera fácil.

mostrarMenuPrincipal

```
public static int mostrarMenuPrincipal() {

String[] MenuPpal = {"1. Ingresar cuarto", "2. Ingresar dispositivos a conectar", "3.
Encender dispositivos", "4. Apagar dispositivos", "5. Mostrar dispositivos en
habitaciones","6. Eliminar dispositivo", "7. Salir"};



return JOptionPane.showOptionDialog(null, "Seleccione lo que guste realizar: ", "Casa
Inteligente", JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null,
MenuPpal, MenuPpal[0]) + 1;

}
```

Muestra una pantalla la ventana que te da a elegir y retorna el resultado númerico. Lo retorna a el main y su resultado se almacena en n.

MostrarDispositivosEnHabitaciones

```
public static void mostrarDispositivosEnHabitaciones(ArrayList<Habitaciones>
listaCuartos) {

if (listaCuartos.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay cuartos registrados para mostrar
dispositivos.", "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



StringBuilder mensaje = new StringBuilder("Dispositivos en cada habitación:\n");



for (Habitaciones cuarto : listaCuartos) {

mensaje.append(cuarto.getNombre()).append(":\n");



DispositivosInteligentes[] dispositivos = cuarto.getDispositivosInteligentes();



for (DispositivosInteligentes dispositivo : dispositivos) {

if (dispositivo != null) {

mensaje.append("Nombre: ").append(dispositivo.getNombre()).append(", Estado: ");

mensaje.append(dispositivo.obtenerEstado() ? "Encendido" : "Apagado").append(", ");



if (dispositivo instanceof TermostatoInte) {

mensaje.append("Temperatura: ").append(((TermostatoInte)
dispositivo).getTemperatura()).append("°C, ");

}
```

```
mensaje.append("\n");

}

}


mensaje.append("\n");

}


JOptionPane.showMessageDialog(null, mensaje.toString(), "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

}
```

1. Se verifica si la lista de cuartos (`listaCuartos`) está vacía. Si no se ha registrado ninguna habitación, se muestra un mensaje de error en un cuadro de diálogo y se sale del método.
2. Se crea un objeto `StringBuilder` llamado `mensaje` para construir el mensaje que se mostrará en el cuadro de diálogo. Inicialmente, se agrega la línea "Dispositivos en cada habitación:" al mensaje.
3. Se inicia un bucle `for-each` para recorrer cada objeto `Habitaciones` en la lista `listaCuartos`. Esto permitirá acceder a cada habitación registrada.
4. Dentro del bucle, se agrega el nombre de la habitación al mensaje. Para obtener el nombre de la habitación, se utiliza el método `getNombre()` del objeto `Habitaciones`. El nombre de la habitación se agrega al mensaje seguido de ":\n".
5. Se obtiene un array de `DispositivosInteligentes` llamado `dispositivos` de la habitación actual. Esto se hace utilizando el método `cuarto.getDispositivosInteligentes()`.
6. Se inicia otro bucle `for-each` para recorrer los dispositivos presentes en la habitación. Para cada dispositivo, se realiza lo siguiente:
    - Se verifica si el dispositivo no es nulo (es decir, si realmente hay un dispositivo presente). Esto se hace para evitar mostrar dispositivos nulos.
    - Se agrega al mensaje el nombre del dispositivo (obtenido mediante `dispositivo.getNombre()`).
    - Se agrega el estado del dispositivo. Si el dispositivo está encendido, se agrega "Encendido"; de lo contrario, se agrega "Apagado". Esto se hace usando el método `obtenerEstado()` del objeto `dispositivo`.
    - Si el dispositivo es una instancia de `TermostatoInte`, se agrega también la temperatura del termostato al mensaje, que se obtiene utilizando `((TermostatoInte) dispositivo).getTemperatura()`.

- Finalmente, se agrega una nueva línea para separar este dispositivo de los demás dentro de la misma habitación.

7. Después de completar el bucle que recorre los dispositivos en una habitación, se agrega una línea en blanco para separar la información de la siguiente habitación.
8. Una vez que se han recorrido todas las habitaciones y se ha construido el mensaje completo, se muestra el mensaje en un cuadro de diálogo utilizando `JOptionPane.showMessageDialog`. Este cuadro de diálogo muestra la información de los dispositivos en cada habitación de la casa inteligente.

> **StringBuilder**
>
> StringBuilder pertenece a el paquete java.lang y se usa para crear y manipular cadenas sin problema. Use este paquete porque permite modificar la cadena de manera dinámica.

ingresarCuartos

```
public static void ingresarCuartos(ArrayList<Habitaciones> listaCuartos) {

String[] TipoCuarto = {"Sala de estar", "Cocina", "Dormitorio"};



String nombreCuarto = JOptionPane.showInputDialog(null, "Por favor, ingrese el nombre
del cuarto: ", "Casa Inteligente", JOptionPane.QUESTION_MESSAGE);

int tipoCuarto = mostrarMenuTipoCuarto(TipoCuarto);



if (tipoCuarto >= 0) {

int maxDispositivos = verNumero("Ingrese la cantidad máxima de dispositivos
inteligentes para esta habitación: ");

ingresarCuarto(listaCuartos, tipoCuarto, nombreCuarto, maxDispositivos);

}

}
```

Sirve para agregar nuevas habitaciones a la lista de habitaciones que se creo en el main.

1. `String[] TipoCuarto = {"Sala de estar", "Cocina", "Dormitorio"};`: Esta línea crea un arreglo de cadenas de texto que contiene los tipos de habitaciones disponibles en la casa inteligente. Los tipos son "Sala de estar," "Cocina," y "Dormitorio."

2. `String nombreCuarto = JOptionPane.showInputDialog(null, "Por favor, ingrese el nombre del cuarto: ", "Casa Inteligente", JOptionPane.QUESTION_MESSAGE);` : Esta línea muestra un cuadro de diálogo de entrada de texto (usando JOptionPane) que solicita al usuario que ingrese el nombre de la nueva habitación. El valor ingresado se almacena en la variable `nombreCuarto`.
3. `int tipoCuarto = mostrarMenuTipoCuarto(TipoCuarto);` : Esta línea llama a un método llamado `mostrarMenuTipoCuarto`, que probablemente muestra un cuadro de diálogo o menú para que el usuario seleccione el tipo de habitación de entre las opciones disponibles en `TipoCuarto`. La selección del usuario se almacena en la variable `tipoCuarto`.
4. `if (tipoCuarto >= 0) { ... }` : Esta condición verifica si el usuario ha seleccionado un tipo de habitación válido (es decir, si `tipoCuarto` es mayor o igual a cero). Si el usuario no selecciona una opción válida, no se agrega la habitación.
5. `int maxDispositivos = verNumero("Ingrese la cantidad máxima de dispositivos inteligentes para esta habitación: ");` : Esta línea muestra un cuadro de diálogo de entrada de número (usando el método `verNumero`) que solicita al usuario que ingrese la cantidad máxima de dispositivos inteligentes que se pueden agregar a la habitación. El valor ingresado se almacena en la variable `maxDispositivos`.
6. `ingresarCuarto(listaCuartos, tipoCuarto, nombreCuarto, maxDispositivos);` : Finalmente, si se ha ingresado correctamente el nombre de la habitación y se ha especificado la cantidad máxima de dispositivos, se llama a otro método llamado `ingresarCuarto`. Este método probablemente crea una instancia de la habitación con el nombre, tipo y límite de dispositivos especificados y la agrega a la lista de habitaciones (`listaCuartos`).

MostrarMenuTipoCuarto

```
public static int mostrarMenuTipoCuarto(String[] tipos) {

String message = "¿Qué tipo de cuarto es? Seleccione una opción:";

String title = "Casa Inteligente";

return JOptionPane.showOptionDialog(null, message, title, JOptionPane.DEFAULT_OPTION,
JOptionPane.QUESTION_MESSAGE, null, tipos, tipos[0]);

}
```

Simplemente muestra la ventana que nos permite elegir el tipo de habitación que queremos entre los tres tipos que hay.

1. `public static int mostrarMenuTipoCuarto(String[] tipos) {` : Este método es estático, lo que significa que se puede llamar sin crear una instancia de la clase que lo contiene. Toma un arreglo de cadenas llamado `tipos` como argumento, que contiene las opciones de tipos de habitaciones disponibles.
2. `String message = "¿Qué tipo de cuarto es? Seleccione una opción:";` : Aquí se define un mensaje que se mostrará en el cuadro de diálogo. Este mensaje informa al usuario que debe

seleccionar una opción para el tipo de la habitación.

3. `String title = "Casa Inteligente";` : Se define un título para el cuadro de diálogo que se mostrará en la barra de título. En este caso, el título es "Casa Inteligente".
4. `return JOptionPane.showOptionDialog(null, message, title, JOptionPane.DEFAULT_OPTION, JOptionPane.QUESTION_MESSAGE, null, tipos, tipos[0]);` : Esta línea es donde se crea y muestra el cuadro de diálogo. Explicaré los argumentos de este método:
   - `null` : El primer argumento es el componente padre del cuadro de diálogo. En este caso, se pasa `null` , lo que significa que el cuadro de diálogo no tiene un componente padre específico y se mostrará en el centro de la pantalla.
   - `message` : El segundo argumento es el mensaje que se mostrará al usuario, que en este caso es "¿Qué tipo de cuarto es? Seleccione una opción."
   - `title` : El tercer argumento es el título del cuadro de diálogo, que en este caso es "Casa Inteligente."
   - `JOptionPane.DEFAULT_OPTION` : El cuarto argumento especifica el tipo de opciones disponibles en el cuadro de diálogo. `JOptionPane.DEFAULT_OPTION` se utiliza para tener un cuadro de diálogo con una opción preseleccionada.
   - `JOptionPane.QUESTION_MESSAGE` : El quinto argumento especifica el ícono que se mostrará en el cuadro de diálogo. En este caso, se utiliza un ícono de pregunta.
   - `null` : El sexto argumento es un objeto de ítems. Aquí, se pasa `null` porque se utilizarán los valores del arreglo `tipos` como opciones.
   - `tipos` : El séptimo argumento es el arreglo de opciones que se mostrarán al usuario. Estas opciones son los tipos de habitaciones disponibles.
   - `tipos[0]` : El último argumento es el valor predeterminado seleccionado, que se establece en la primera opción del arreglo `tipos` .

ingresarCuarto

```
public static void ingresarCuarto(ArrayList<Habitaciones> listaCuartos, int
tipoCuarto, String nombreCuarto, int maxDispositivos) {

Habitaciones cuarto = null;



switch (tipoCuarto) {

case 0:

cuarto = new Saladeestar(nombreCuarto, maxDispositivos);

break;

case 1:

cuarto = new Cocina(nombreCuarto, maxDispositivos);
```

```
break;

case 2:

cuarto = new Dormitorio(nombreCuarto, maxDispositivos);

break;

}



if (cuarto != null) {

listaCuartos.add(cuarto);

JOptionPane.showMessageDialog(null, "El cuarto " + cuarto.getNombre() + " se ha
registrado con éxito.", "Casa Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

}
```

1. `public static void ingresarCuarto(ArrayList<Habitaciones> listaCuartos, int tipoCuarto, String nombreCuarto, int maxDispositivos) {` : Este método es público, estático y toma varios argumentos. Recibe una lista de habitaciones llamada `listaCuartos`, un entero que representa el tipo de cuarto (`tipoCuarto`), una cadena que es el nombre de la habitación (`nombreCuarto`), y un entero que representa el número máximo de dispositivos inteligentes que se pueden conectar en la habitación (`maxDispositivos`).
2. `Habitaciones cuarto = null;` : Se declara una variable llamada `cuarto` que se inicializa como nula. Esta variable se utilizará para crear y almacenar la instancia de la habitación.
3. `switch (tipoCuarto) {` : Se utiliza una declaración `switch` para determinar el tipo de habitación que se debe crear en función del valor de `tipoCuarto`, que se pasa como argumento al método.
4. Dentro del bloque `switch`, hay tres casos posibles:
   - `case 0` : Si `tipoCuarto` es igual a 0, se crea una instancia de `Saladeestar` con el nombre de la habitación y el número máximo de dispositivos inteligentes y se asigna a la variable `cuarto`
   - `case 1` : Si `tipoCuarto` es igual a 1, se crea una instancia de `Cocina` con el nombre de la habitación y el número máximo de dispositivos inteligentes y se asigna a la variable `cuarto`.
   - `case 2` : Si `tipoCuarto` es igual a 2, se crea una instancia de `Dormitorio` con el nombre de la habitación y el número máximo de dispositivos inteligentes y se asigna a la variable `cuarto`.
5. Después del `switch`, se verifica si la variable `cuarto` no es nula:
   - Si `cuarto` no es nula, significa que se ha creado una instancia de habitación correctamente. En ese caso, se agrega la habitación a la lista de cuartos (`listaCuartos.add(cuarto)`).

- Luego, se muestra un mensaje al usuario que confirma que la habitación se ha registrado con éxito.
- Si `cuarto` es nula, significa que hubo un problema al crear la instancia de la habitación, y no se agrega a la lista de cuartos.

ingresarDispositivos

```
public static void ingresarDispositivos(ArrayList<Habitaciones> listaCuartos,
ArrayList<DispositivosInteligentes> listaDispTot) {

if (listaCuartos.isEmpty()) {

JOptionPane.showMessageDialog(null, "ERROR: Registre primero un cuarto para poder
hacer esta acción.", "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



String[] subclases = {"Aire Acondicionado", "Cafetera", "Horno", "Luces
Inteligentes", "Termostato Inteligente", "Seguridad Inteligente"};



int seleccionSubclase = JOptionPane.showOptionDialog(

null, "Seleccione la subclase de dispositivo inteligente:", "Casa Inteligente",
JOptionPane.DEFAULT_OPTION,

JOptionPane.QUESTION_MESSAGE, null, subclases, subclases[0]);



if (seleccionSubclase >= 0) {

String subclaseSeleccionada = subclases[seleccionSubclase];



DispositivosInteligentes dispositivo = null;

int idDispositivo = verNumero("Ingrese el ID del dispositivo o presione 0 para
asignar automáticamente: ");

String nombreDispositivo = JOptionPane.showInputDialog(null, "Por favor, ingrese el
nombre del dispositivo: ", "Casa Inteligente", JOptionPane.QUESTION_MESSAGE);
```

```
String[] habitacionesDisponibles = new String[listaCuartos.size()];

for (int i = 0; i < listaCuartos.size(); i++) {

habitacionesDisponibles[i] = listaCuartos.get(i).getNombre();

}

String habitacionElegida = (String) JOptionPane.showInputDialog(null, "Seleccione la
habitación a la que desea asignar el dispositivo:", "Casa Inteligente",
JOptionPane.QUESTION_MESSAGE, null, habitacionesDisponibles,
habitacionesDisponibles[0]);



if (habitacionElegida != null) {

for (Habitaciones cuarto : listaCuartos) {

if (cuarto.getNombre().equals(habitacionElegida)) {

if (subclaseSeleccionada.equals("Termostato Inteligente")) {

int temperatura = verNumero("Ingrese la temperatura deseada: ");

dispositivo = new TermostatoInte(idDispositivo, nombreDispositivo, false,
temperatura);

} else {

dispositivo = new DispositivosInteligentes(idDispositivo, nombreDispositivo, false);

}



if (dispositivo != null) {

cuarto.addDispositivosInteligentes(dispositivo);

listaDispTot.add(dispositivo);

if (dispositivo instanceof TermostatoInte) {

JOptionPane.showMessageDialog(null, "El dispositivo " + dispositivo.getNombre() + "
se ha registrado con éxito en la habitación " + cuarto.getNombre() + " con
temperatura establecida en " + ((TermostatoInte) dispositivo).getTemperatura() +
```

```
"°C.", "Casa Inteligente", JOptionPane.INFORMATION_MESSAGE);

} else {

JOptionPane.showMessageDialog(null, "El dispositivo " + dispositivo.getNombre() + "
se ha registrado con éxito en la habitación " + cuarto.getNombre(), "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

}

}

}

}

}

}
```

Este método permite agregar dispositivos a las habitaciones

1. Comienza con una comprobación para asegurarse de que haya al menos una habitación registrada en la lista de cuartos (`listaCuartos`). Si no hay ninguna habitación registrada, muestra un mensaje de error y regresa.
2. Se crea un array de cadenas llamado `subclases` que contiene diferentes tipos de dispositivos inteligentes que se pueden agregar, como "Aire Acondicionado", "Cafetera", "Horno", "Luces Inteligentes", "Termostato Inteligente" y "Seguridad Inteligente".
3. Utiliza un cuadro de diálogo (`JOptionPane.showOptionDialog`) para que el usuario seleccione el tipo de dispositivo inteligente que desea agregar. La selección se almacena en la variable `seleccionSubclase`.
4. Si el usuario selecciona un tipo de dispositivo (`seleccionSubclase >= 0`), se recoge la cadena correspondiente a la subclase seleccionada desde el array `subclases`.
5. Se inicia la creación de un nuevo dispositivo inteligente. El usuario puede ingresar un ID personalizado para el dispositivo o presionar 0 para asignar automáticamente un ID. Se recoge esta información en la variable `idDispositivo`. Luego, se solicita al usuario que ingrese el nombre del dispositivo, que se almacena en `nombreDispositivo`.
6. Se crea un array de cadenas llamado `habitacionesDisponibles` para almacenar los nombres de las habitaciones disponibles donde se pueden agregar dispositivos. Luego, se muestra un cuadro de diálogo para que el usuario seleccione la habitación a la que desea asignar el dispositivo. La selección se almacena en `habitacionElegida`.
7. Si el usuario selecciona una habitación (`habitacionElegida != null`), el código continúa ejecutándose.

8. El bucle `for` itera a través de la lista de cuartos (`listaCuartos`) y busca la habitación que coincide con el nombre seleccionado por el usuario (`habitacionElegida`).
9. Una vez que se encuentra la habitación, se crea una instancia del dispositivo inteligente seleccionado, ya sea un `TermostatoInte` con una temperatura inicial (si la subclase seleccionada es "Termostato Inteligente") o un `DispositivosInteligentes` normal.
10. Si se crea con éxito una instancia del dispositivo (`dispositivo != null`), se agrega a la habitación utilizando el método `addDispositivosInteligentes` y se agrega a la lista de dispositivos totales (`listaDispTot`). Luego, se muestra un mensaje de confirmación al usuario que indica que el dispositivo se ha registrado con éxito en la habitación seleccionada. Si el dispositivo es un termostato, también muestra la temperatura inicial.

Encender dispositivos

```
public static void encenderDispositivo(ArrayList<DispositivosInteligentes>
listaDispTot) {

if (listaDispTot.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay dispositivos registrados para encender.",
"Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a encender: ");



for (DispositivosInteligentes dispositivo : listaDispTot) {

if (dispositivo.getId() == idDispositivo) {

dispositivo.encender();

if (dispositivo instanceof TermostatoInte) {

int nuevaTemperatura = verNumero("Ingrese la nueva temperatura deseada: ");

((TermostatoInte) dispositivo).setTemperatura(nuevaTemperatura);

JOptionPane.showMessageDialog(null, "El dispositivo ha sido encendido y la
temperatura se ha establecido en " + nuevaTemperatura + "°C.", "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

} else {
```

```
JOptionPane.showMessageDialog(null, "El dispositivo ha sido encendido.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

}

return;

}

}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo, "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

}
```

1. Comienza con una comprobación para asegurarse de que haya al menos un dispositivo registrado en la lista de dispositivos totales (`listaDispTot`). Si no hay dispositivos registrados, muestra un mensaje de error y regresa.
2. Luego, solicita al usuario que ingrese el ID del dispositivo que desea encender. Esto se hace llamando a la función `verNumero` para obtener un número entero que representa el ID.
3. El método utiliza un bucle `for-each` para iterar a través de la lista de dispositivos totales (`listaDispTot`) y busca el dispositivo con el ID ingresado por el usuario.
4. Si encuentra un dispositivo con el ID correspondiente, realiza lo siguiente:
   - Llama al método `encender` del dispositivo, que cambia el estado del dispositivo a encendido.
   - Verifica si el dispositivo es una instancia de `TermostatoInte` utilizando `instanceof` . Si es un termostato, le solicita al usuario que ingrese una nueva temperatura deseada y actualiza la temperatura del termostato utilizando el método `setTemperatura` .
   - Muestra un mensaje de confirmación que indica que el dispositivo se ha encendido con éxito. Si es un termostato, también muestra la nueva temperatura deseada.
5. Finaliza el método con una declaración `return` , lo que significa que después de encender el dispositivo y mostrar el mensaje de confirmación, no se procesará ningún otro dispositivo en la lista. El método termina.
6. Si el bucle `for-each` no encuentra un dispositivo con el ID ingresado, muestra un mensaje de error indicando que el dispositivo con ese ID no se encontró en la "Casa Inteligente".

apagarDispositivo

```
public static void apagarDispositivo(ArrayList<DispositivosInteligentes>
listaDispTot) {

if (listaDispTot.isEmpty()) {
```

```
JOptionPane.showMessageDialog(null, "No hay dispositivos registrados para apagar.",
"Casa Inteligente", JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a apagar: ");



for (DispositivosInteligentes dispositivo : listaDispTot) {

if (dispositivo.getId() == idDispositivo) {

dispositivo.apagar();

JOptionPane.showMessageDialog(null, "El dispositivo ha sido apagado.", "Casa
Inteligente", JOptionPane.INFORMATION_MESSAGE);

return;

}

}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo, "Casa Inteligente", JOptionPane.ERROR_MESSAGE);

}
```

1. Comienza con una comprobación para asegurarse de que haya al menos un dispositivo registrado en la lista de dispositivos totales (`listaDispTot`). Si no hay dispositivos registrados, muestra un mensaje de error y regresa.
2. Luego, solicita al usuario que ingrese el ID del dispositivo que desea apagar. Esto se hace llamando a la función `verNumero` para obtener un número entero que representa el ID.
3. El método utiliza un bucle `for-each` para iterar a través de la lista de dispositivos totales (`listaDispTot`) y busca el dispositivo con el ID ingresado por el usuario.
4. Si encuentra un dispositivo con el ID correspondiente, realiza lo siguiente:
    - Llama al método `apagar` del dispositivo, que cambia el estado del dispositivo a apagado.
    - Muestra un mensaje de confirmación que indica que el dispositivo se ha apagado con éxito.

5. Finaliza el método con una declaración `return`, lo que significa que después de apagar el dispositivo y mostrar el mensaje de confirmación, no se procesará ningún otro dispositivo en la lista. El método termina.
6. Si el bucle `for-each` no encuentra un dispositivo con el ID ingresado, muestra un mensaje de error indicando que el dispositivo con ese ID no se encontró en la "Casa Inteligente".

verNumero

```
public static int verNumero(String mensaje) {

int numero = 0;

boolean esNumeroValido = false;



do {

String input = JOptionPane.showInputDialog(null, mensaje, "Casa Inteligente",
JOptionPane.QUESTION_MESSAGE);



try {

numero = Integer.parseInt(input);

esNumeroValido = true;

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(null, "Por favor, ingrese un número válido.", "Casa
Inteligente", JOptionPane.WARNING_MESSAGE);

}



} while (!esNumeroValido);



return numero;

}
```

Funciona como verificador de un entero valido. Se utiliza en diversos métodos pero generalmente tiene protagonismo cuando el usuario debe de ingresar un número entero para el id o la cantidad de

dispositivos.

1. Se le pasa un mensaje como argumento que se mostrará al usuario para solicitar la entrada de un número. El mensaje se muestra en una ventana de diálogo.
2. Se inicializa una variable `numero` en 0 y un booleano `esNumeroValido` en `false`. El `numero` almacenará el número entero válido ingresado por el usuario, y `esNumeroValido` se utilizará para controlar un bucle que continuará hasta que se ingrese un número válido.
3. Entra en un bucle `do-while`. En el interior del bucle:
   - Muestra una ventana de diálogo que muestra el mensaje especificado y permite al usuario ingresar una cadena de texto.
   - Luego, se intenta convertir la cadena de texto ingresada en un número entero utilizando `Integer.parseInt(input)`. Si el usuario ingresa una cadena de texto que no es un número válido, se generará una excepción `NumberFormatException`.
4. Si la conversión a número entero tiene éxito (sin generar una excepción), significa que el usuario ha ingresado un número válido. En este caso, `numero` se actualiza con el valor numérico ingresado, y `esNumeroValido` se establece en `true`.
5. Si el usuario ingresa una cadena de texto que no es un número válido, se captura la excepción `NumberFormatException`. En este caso, se muestra un mensaje de advertencia que indica que se debe ingresar un número válido.
6. El bucle continúa mientras `esNumeroValido` sea `false`, lo que significa que continuará ejecutándose hasta que el usuario ingrese un número válido.
7. Una vez que se ingresa un número válido, el bucle se detiene y el método devuelve el valor de `numero`, que contiene el número entero válido ingresado por el usuario.

eliminarDispositivoDeHabitacion

```
public static void eliminarDispositivoDeHabitacion(ArrayList<Habitaciones>
listaCuartos, ArrayList<DispositivosInteligentes> listaDispTot) {

if (listaCuartos.isEmpty() || listaDispTot.isEmpty()) {

JOptionPane.showMessageDialog(null, "No hay cuartos registrados o dispositivos
registrados para realizar esta acción.", "Casa Inteligente",
JOptionPane.ERROR_MESSAGE);

return;

}



int idDispositivo = verNumero("Ingrese el ID del dispositivo a eliminar de la
habitación: ");
```

```
for (Habitaciones cuarto : listaCuartos) {

for (DispositivosInteligentes dispositivo : cuarto.getDispositivosInteligentes()) {

if (dispositivo != null && dispositivo.getId() == idDispositivo) {

cuarto.removeDispositivosInteligentes(dispositivo);

listaDispTot.remove(dispositivo);

JOptionPane.showMessageDialog(null, "El dispositivo ha sido eliminado de la
habitación " + cuarto.getNombre(), "Casa Inteligente",
JOptionPane.INFORMATION_MESSAGE);

return;

}

}

}



JOptionPane.showMessageDialog(null, "No se encontró el dispositivo de ID " +
idDispositivo + " en ninguna habitación.", "Casa Inteligente",
JOptionPane.ERROR_MESSAGE);

}
```

Se utiliza para eliminar dispositivos de una habitación.

1. Primero, el método verifica si la lista de cuartos (`listaCuartos`) o la lista total de dispositivos (`listaDispTot`) están vacías. Si alguna de estas listas está vacía, muestra un mensaje de error en una ventana de diálogo y regresa temprano del método. Esto evita realizar la eliminación de dispositivos si no hay cuartos registrados o dispositivos registrados para operar.
2. A continuación, el método solicita al usuario que ingrese el ID del dispositivo que desea eliminar de una habitación. Esto se hace llamando al método `verNumero` , que se explicó anteriormente, y se almacena el ID ingresado en la variable `idDispositivo` .
3. Luego, el método recorre la lista de cuartos (`listaCuartos`) en un bucle `for-each` utilizando la variable `cuarto` para representar cada habitación.
4. Dentro del bucle de habitaciones, se inicia otro bucle `for-each` para recorrer los dispositivos inteligentes en cada habitación. Esto se hace llamando al método `cuarto.getDispositivosInteligentes()` para obtener la lista de dispositivos en la habitación actual.

5. Para cada dispositivo inteligente (`dispositivo`) en la lista de dispositivos de la habitación, se verifica si el dispositivo no es nulo y si el ID del dispositivo coincide con el ID ingresado por el usuario (`idDispositivo`).
6. Si se encuentra un dispositivo con el ID correspondiente, se procede a eliminarlo de la habitación y de la lista total de dispositivos. Esto se hace llamando al método `cuarto.removeDispositivosInteligentes(dispositivo)` para eliminar el dispositivo de la habitación y llamando a `listaDispTot.remove(dispositivo)` para eliminarlo de la lista total de dispositivos.
7. Luego, se muestra un mensaje informativo en una ventana de diálogo indicando que el dispositivo ha sido eliminado de la habitación.
8. Finalmente, el método regresa temprano, completando su ejecución.
9. Si el bucle no encuentra un dispositivo con el ID proporcionado en ninguna de las habitaciones, se muestra un mensaje de error en una ventana de diálogo indicando que no se encontró un dispositivo con el ID ingresado.