**TITULO DE LABORATORIO**

**PARCIAL (CASILLEROS)**

**HANNA KATHERINE ABRIL GÓNGORA**

**KAROL ASLEY ORJUELA MAPE**

**UNIVERSIDAD MANUELA BELTRÁN**

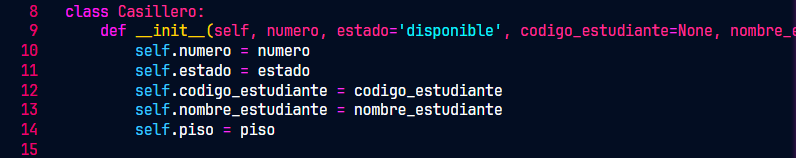
**PROGRAMACIÓN ORIENTADA A OBJECTOS**

**DOCENTE**

**DIANA MARCELA TOQUICA RODRÍGUEZ**

**BOGOTÁ DC VIERNES 22 DE MARZO**

1. **EXPLICACIÓN DEL CÓDIGO EN PYTHON**
   * La primera parte del código define la clase Casillero. Esta clase representa un casillero con atributos como número, estado, codigo\_estudiante, nombre\_estudiante y piso. El método \_\_init\_\_ se utiliza para inicializar un objeto Casillero con los valores proporcionados.



* + El método alquilar se utiliza para alquilar un casillero asignando un código de estudiante y un nombre de estudiante al mismo. Verifica si el casillero está disponible antes de alquilarlo si este no está disponible no se puede alquilar.

Texto

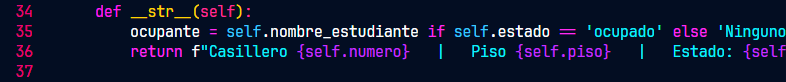
Descripción generada automáticamente

* + El método devolver se utiliza para devolver un casillero si está ocupado. Cambia el estado del casillero a 'disponible' y borra la información del estudiante que se ingresó al momento de pedir el casillero.

Texto

Descripción generada automáticamente

* + El método \_\_str\_\_ devuelve una representación en forma de cadena del casillero, incluyendo su número, piso, estado y el nombre del ocupante si está ocupado.



* + La clase Inventario se encarga de gestionar una lista de casilleros. En el método \_\_init\_\_, se inicializa la lista de casilleros y se llama al método agregar\_casilleros para crear los casilleros y agregarlos al inventario.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* + El método agregar\_casilleros crea los casilleros según el total de casilleros calculado y los añade a la lista de casilleros del inventario, en este caso se tomó la decisión de tener 150 casilleros por piso, siendo 7 pisos, para que simulara la cantidad de casilleros de nuestra institución aproximadamente.

Texto

Descripción generada automáticamente

* + Los métodos buscar\_por\_estado y buscar\_casillero se utilizan para buscar casilleros en el inventario según diferentes criterios como el estado, número, ocupante, estado y piso.

Texto

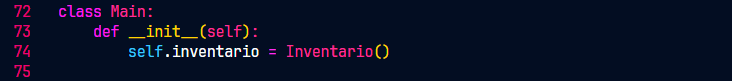
Descripción generada automáticamente

* + El método mostrar\_inventario imprime el inventario de todos los casilleros en forma de lista.

Texto

Descripción generada automáticamente

* + La clase Main se encarga de gestionar la interfaz de usuario y la lógica principal del programa.



* + En el método mostrar\_menu, se muestra un menú de opciones y se ejecuta la lógica correspondiente a la opción seleccionada por el usuario. Las opciones del menú incluyen alquilar casillero, devolver casillero, buscar casillero, mostrar inventario y salir del programa.

Texto

Descripción generada automáticamente

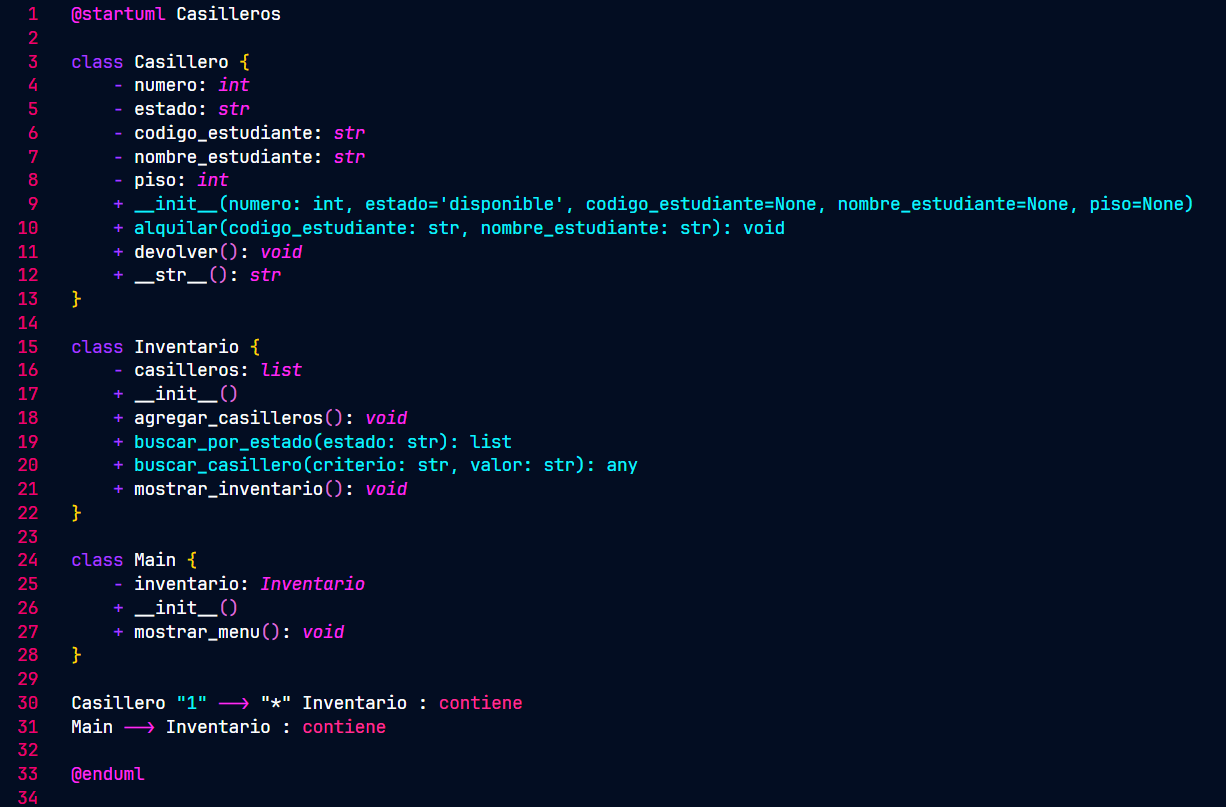
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. **EXPLICACION DEL CODIGO UML**

****

* **Línea 1-5:** Se define la clase Casillero con atributos como número, estado, código del estudiante, nombre del estudiante y piso.
* **Línea 6-10:** El método \_\_init\_\_ inicializa un objeto de la clase Casillero con valores predeterminados o los proporcionados durante la creación del objeto.
* **Línea 11-12:** El método alquilar se utiliza para alquilar un casillero a un estudiante específico.
* **Línea 13-14:** El método devolver cambia el estado del casillero a ‘disponible’.
* **Línea 15-16:** El método especial \_\_str\_\_ se utiliza para devolver una representación legible en forma de cadena del objeto.
* **Línea 18-22:** Se define la clase ‘Inventario’ que tiene una lista de casilleros y métodos para agregar casilleros y buscar por estado.
* **Línea 23-24:** El método ‘agregar\_casilleros’ añade nuevos casilleros al inventario.
* **Línea 25-26:** El método ‘buscar\_por\_estado’ devuelve una lista de todos los casilleros con un estado específico.
* **Línea 27-28:** El método ‘mostrar\_inventario’ puede ser utilizado para mostrar todos los casilleros en el inventario (aunque no se muestra su implementación).
* **Línea 30-32:** La clase ‘Main’ parece ser una clase principal que contiene un objeto de la clase ‘Inventario’ y tiene métodos para interactuar con él, aunque no se muestra su implementación completa.
* **Línea 34–35:** Estas líneas parecen estar relacionadas con UML o algún tipo de notación para describir las relaciones entre las clases, pero no son parte del código Python ejecutable estándar.

1. **DIAGRAMA DEL CODIGO EN UML**

**Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente**

1. **EXPLICACION Y USO**

El código implementa un sistema de gestión de casilleros en un establecimiento, como podría ser una escuela o una institución similar. Comienza definiendo la clase `Casillero`, que representa cada casillero individualmente con atributos como número, estado (disponible u ocupado), código y nombre del estudiante que lo alquila, y el piso en el que se encuentra. Los métodos de la clase permiten alquilar un casillero, devolverlo, entre otros.

La clase `Inventario` se encarga de gestionar múltiples casilleros, creándolos e integrándolos en una lista. Además, proporciona métodos para buscar casilleros por estado, número, nombre del ocupante o piso. El objeto `Main` actúa como la interfaz de usuario, mostrando un menú para realizar acciones como alquilar, devolver, buscar casilleros y mostrar el inventario.

En resumen, el código organiza y controla el uso de casilleros, permitiendo realizar operaciones básicas como alquiler y devolución, y facilitando la gestión y visualización del inventario.

1. **DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO**

**5.1. Prueba 1**

**Forma

Descripción generada automáticamente con confianza mediaPantalla de computadora con letras

Descripción generada automáticamente con confianza baja**

En esta prueba se muestra el inventario de casilleros con su respectiva información, como el número, piso, estado, y ocupante.

**5.2. Prueba 2**

**Texto

Descripción generada automáticamente**

En esta prueba se demuestra la funcionalidad del método para prestar casilleros, el cual pide el nombre y el código del estudiante además del casillero que se le va a prestar.

**5.3. Prueba 3**

**Texto

Descripción generada automáticamente**

En esta prueba se demuestra la funcionalidad del método para prestar casilleros, el cual pide el nombre y el código del estudiante además del casillero que se le va a prestar. La diferencia es que aquí se intenta pedir prestado un casillero que ya esta en uso, por lo que el sistema no permite prestar ese casillero.

**5.4. Prueba 4**

**Texto

Descripción generada automáticamente**

En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el numero del casillero, el cual al ser encontrado arroja la información correspondiente de este.

**5.5. Prueba 5**

****

En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el nombre del ocupante del casillero, el cual al ser encontrado arroja la información correspondiente de este. Tener en cuenta que se debe colocar el nombre exacto a como se coloco al momento de pedir el casillero. Se piensa mas adelante establecer un método de coincidencias para q sea más fácil la búsqueda.

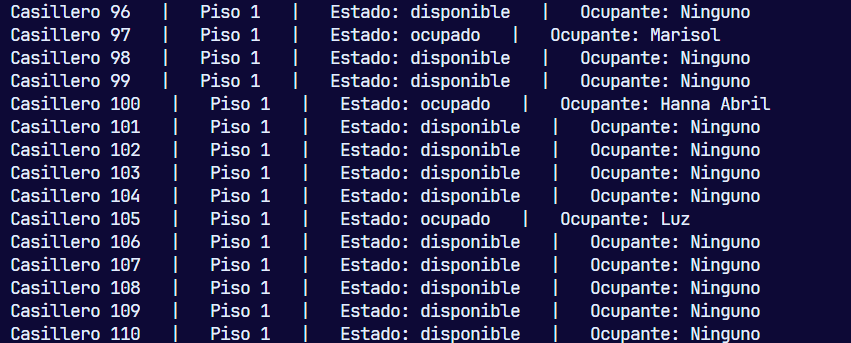
**5.6. Prueba 6**

**Texto

Descripción generada automáticamente**

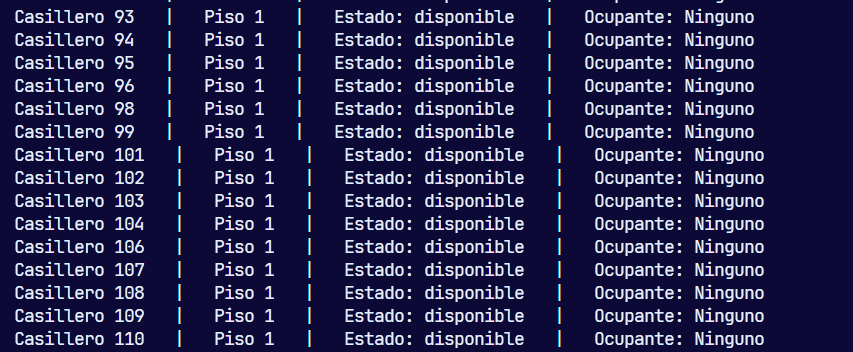
En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el estado del casillero que en este caso se probó con los que están ocupados, el cual al ser encontrado arroja la información correspondiente de este.

**5.7. Prueba 7**

****

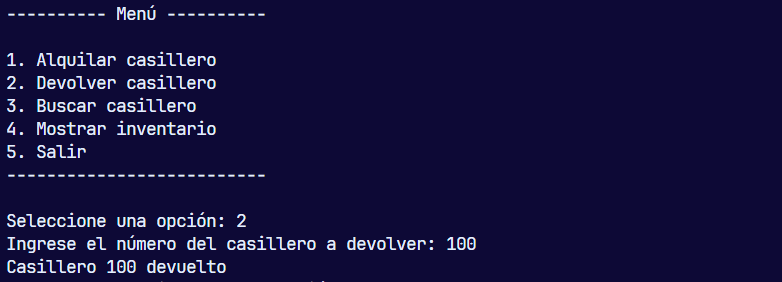
En esta prueba se realiza la prueba de búsqueda por piso, para este ejemplo se hizo con el piso 1 en donde podemos ver la respectiva información de un fragmento de casilleros de este piso.

**5.8. Prueba 8**

****

En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el estado del casillero que en este caso se probó con los que están desocupados, comparando la prueba anterior donde se muestran los ocupados según la búsqueda por piso, en esta podemos ver que dichos casilleros ocupados anteriormente en este caso al listar los desocupados esos casilleros no aparecen en el listado.

**5.9. Prueba 9**

**Texto

Descripción generada automáticamente**

En esta prueba podemos ver que se devolvió un casillero mediante el numero de este, y además de eso se revisó la búsqueda de casilleros por numero y se vio que paso de estado ocupado con un ocupante a estado desocupado.

**5.10. Prueba 10**

**Imagen que contiene Texto

Descripción generada automáticamente**

Se prueba la salida del sistema.

1. **DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE**

Entre las consideraciones importantes en el código fue tener una biblioteca de casilleros divididos por pisos y con su respectivo número, ya que así seria mas funcional el programa respecto a prestamos y devoluciones. Además de eso para hacer mas eficiente el programa se añadió un método de búsqueda de casilleros para localizar un casillero mediante parámetros como su número, el piso, el estado si esta ocupado o no, además de eso se puede buscar por el nombre del propietario siempre y cuando este este ocupando uno, todo esto se hizo con el propósito de mejorar la administración de los casilleros ya que para propósitos de este código y para adherirnos mejor a la vida real tomamos como ejemplo nuestra universidad que cuenta con mas de 1000 casilleros al igual que nuestro código.

1. **PILARES DE POO IMPLEMENTADOS**

**7.1. Encapsulación:** Se utiliza la encapsulación para ocultar los detalles internos de la implementación de las clases. Por ejemplo, los atributos de la clase Casillero como número, estado, codigo\_estudiante, nombre\_estudiante, y piso se definen como variables de instancia y se acceden a través de métodos de la clase.

**7.2. Abstracción:** El código utiliza la abstracción al definir las clases Casillero e Inventario para modelar conceptos del mundo real relacionados con casilleros y un inventario de casilleros, respectivamente. Además, se definen métodos como alquilar, devolver, buscar\_por\_estado, buscar\_casillero, mostrar\_inventario, etc., para proporcionar una interfaz para interactuar con los objetos y ocultar los detalles internos de su implementación.

**7.3. Polimorfismo:** El polimorfismo permite que objetos de diferentes clases respondan al mismo mensaje (método) de manera distinta. En este código, el método buscar\_casillero en la clase Inventario se comporta de manera diferente dependiendo del criterio de búsqueda proporcionado, lo cual es un ejemplo de polimorfismo.

1. **CONCLUCIONES**

Al desarrollar el código, hemos fortalecido nuestra comprensión de la programación orientada a objetos (POO) al implementar clases como Casillero, Inventario y Main, cada una con sus propios métodos y atributos para gestionar casilleros y su uso en un entorno práctico. Aprendimos a diseñar un sistema modular y escalable donde cada clase tiene responsabilidades específicas, lo que facilita la organización y mantenimiento del código. También mejoramos nuestras habilidades en la manipulación de listas y en la creación de interfaces de usuario simples pero efectivas, lo que nos permitió crear un menú interactivo para realizar acciones como alquilar, devolver, buscar y mostrar el inventario de casilleros. En general, este proyecto nos ha brindado una experiencia valiosa en el diseño y desarrollo de aplicaciones basadas en POO, así como en la implementación de funcionalidades prácticas para la gestión de recursos.