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.

```
 8  class Casillero:
 9      def __init__(self, numero, estado='disponible', codigo_estudiante=None, nombre_
10          self.numero = numero
11          self.estado = estado
12          self.codigo_estudiante = codigo_estudiante
13          self.nombre_estudiante = nombre_estudiante
14          self.piso = piso
15
```

- 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.

```
16      def alquilar(self, codigo_estudiante, nombre_estudiante):
17          if self.estado == 'disponible':
18              self.estado = 'ocupado'
19              self.codigo_estudiante = codigo_estudiante
20              self.nombre_estudiante = nombre_estudiante
21              print(f"Casillero {self.numero} alquilado a {self.nombre_estudiante}")
22          else:
23              print(f"Casillero {self.numero} ya está ocupado")
24
```

- 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.

```
25      def devolver(self):
26          if self.estado == 'ocupado':
27              self.estado = 'disponible'
28              self.codigo_estudiante = None
29              self.nombre_estudiante = None
30              print(f"Casillero {self.numero} devuelto")
31          else:
32              print(f"Casillero {self.numero} ya está disponible")
33
```

- 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.

```
34      def __str__(self):
35          ocupante = self.nombre_estudiante if self.estado == 'ocupado' else 'Ninguno
36          return f"Casillero {self.numero}   |   Piso {self.piso}   |   Estado: {self
37
```

- 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.

```
39  class Inventario:
40      def __init__(self):
41          self.casilleros = []
42          self.agregar_casilleros()
43
```

- 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.

```
44      def agregar_casilleros(self):
45          total_casilleros = 150 * 7  # 150 casilleros por piso, 7 pisos
46          for numero_casillero in range(1, total_casilleros + 1):
47              piso = (numero_casillero - 1) // 150 + 1
48              casillero = Casillero(numero_casillero, piso=piso)
49              self.casilleros.append(casillero)
50
```

- 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.

```
51      def buscar_por_estado(self, estado):
52          return [casillero for casillero in self.casilleros if casillero.estado == es
53
54      def buscar_casillero(self, criterio, valor):
55          criterio = criterio.lower()
56          if criterio == '1':
57              return next((casillero for casillero in self.casilleros if casillero.nu
58          elif criterio == '2':
59              return next((casillero for casillero in self.casilleros if casillero.est
60          elif criterio == '3':
61              return self.buscar_por_estado(valor)
62          elif criterio == '4':
63              return [casillero for casillero in self.casilleros if casillero.piso ==
64          return None
65
```

- El método mostrar_inventario imprime el inventario de todos los casilleros en forma de lista.

```
66      def mostrar_inventario(self):
67          print("Inventario de casilleros:")
68          for casillero in self.casilleros:
69              print(casillero)
70
```

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

```
72  class Main:
73      def __init__(self):
74          self.inventario = Inventario()
75
```

- 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.

```python
    def mostrar_menu(self):
        while True:
            print("---------- Menú ----------")
            print("")
            print("1. Alquilar casillero")
            print("2. Devolver casillero")
            print("3. Buscar casillero")
            print("4. Mostrar inventario")
            print("5. Salir")
            print("--------------------------")
            print("")
            opcion = input("Seleccione una opción: ")
            match opcion:
                case '1':
                    codigo_estudiante = input("Ingrese el código del estudiante: ")
                    nombre_estudiante = input("Ingrese el nombre del estudiante: ")
                    numero_casillero = int(input("Ingrese el número del casillero a
                    casillero = self.inventario.buscar_casillero('1', numero_casille
                    if casillero is not None:
                        if casillero.alquilar(codigo_estudiante, nombre_estudiante):
                            print("Alquiler realizado con éxito.")
                        else:
                            print("No se pudo realizar el alquiler.")
                    else:
                        print("No se encontró ningún casillero con ese número.")
                case '2':
                    numero_casillero = int(input("Ingrese el número del casillero a
                    casillero = self.inventario.buscar_casillero('1', numero_casille
                    if casillero is not None:
                        if casillero.devolver():
                            print("Devolución realizada con éxito.")
                        else:
                            print("No se pudo realizar la devolución.")
                    else:
                        print("No se encontró ningún casillero con ese número.")
```

```
111                 case '3':
112                     print(" ---------- Opciones de búsqueda ----------")
113                     print("")
114                     print("1. Por número de casillero")
115                     print("2. Por nombre de ocupante (si está ocupado)")
116                     print("3. Por estado (ocupado o desocupado)")
117                     print("4. Por piso")
118                     print("-------------------------------------------")
119                     print("")
120                     opcion_busqueda = input("Seleccione una opción de búsqueda: ")
121                     match opcion_busqueda:
122                         case '3':
123                             print(" --- Submenú para búsqueda por estado ---")
124                             print("-----------------------------------------")
125                             print("1. Casilleros ocupados")
126                             print("2. Casilleros desocupados")
127                             print("-----------------------------------------")
128                             print("")
129                             opcion_estado = input("Seleccione una opción de estado:
130                             match opcion_estado:
131                                 case '1':
132                                     casilleros_encontrados = self.inventario.buscar
133                                 case '2':
134                                     casilleros_encontrados = self.inventario.buscar
135                                 case _:
136                                     print("Opción de estado no válida.")
137                                     continue
138                             if casilleros_encontrados:
139                                 print("\nCasilleros encontrados:")
140                                 for casillero in casilleros_encontrados:
141                                     print(casillero)
142                             else:
143                                 print("\nNo se encontraron casilleros con ese estad
144                         case _:
145                             valor_busqueda = input("Ingrese el valor a buscar: ")
146                             casilleros_encontrados = self.inventario.buscar_casille
147                             if casilleros_encontrados is not None:
148                                 if isinstance(casilleros_encontrados, list):
149                                     print("\nCasilleros encontrados:")
150                                     for casillero in casilleros_encontrados:
151                                         print(casillero)
152                                 else:
153                                     print("\nCasillero encontrado:")
154                                     print(casilleros_encontrados)
155                             else:
156                                 print("\nNo se encontró ningún casillero con esos c
```

```
                case '4':
                    self.inventario.mostrar_inventario()
                case '5':
                    print("Saliendo del programa...")
                    break
                case _:
                    print("Opción no válida. Intente de nuevo.")


main = Main()
main.mostrar_menu()

```

2. **EXPLICACION DEL CODIGO UML**

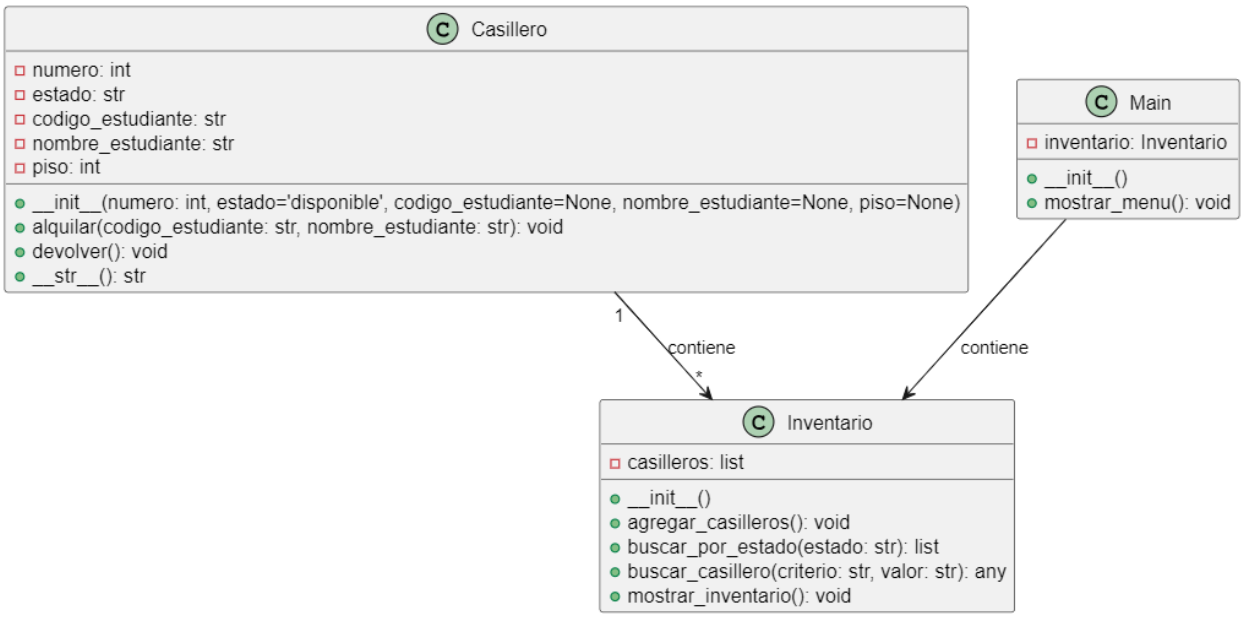
```
@startuml Casilleros

class Casillero {
    - numero: int
    - estado: str
    - codigo_estudiante: str
    - nombre_estudiante: str
    - piso: int
    + __init__(numero: int, estado='disponible', codigo_estudiante=None, nombre_estudiante=None, piso=None)
    + alquilar(codigo_estudiante: str, nombre_estudiante: str): void
    + devolver(): void
    + __str__(): str
}

class Inventario {
    - casilleros: list
    + __init__()
    + agregar_casilleros(): void
    + buscar_por_estado(estado: str): list
    + buscar_casillero(criterio: str, valor: str): any
    + mostrar_inventario(): void
}

class Main {
    - inventario: Inventario
    + __init__()
    + mostrar_menu(): void
}

Casillero "1" ⟶ "*" Inventario : contiene
Main ⟶ Inventario : contiene

@enduml

```

- **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.

3. DIAGRAMA DEL CODIGO EN UML



4. 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.

5. DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO

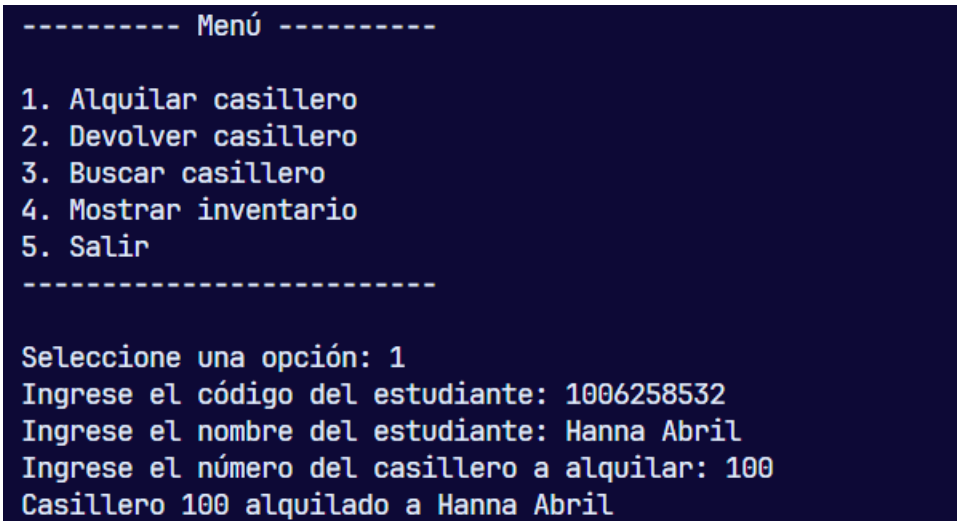
5.1. Prueba 1

```
---------- Menú ----------

1. Alquilar casillero
2. Devolver casillero
3. Buscar casillero
4. Mostrar inventario
5. Salir
--------------------------

Seleccione una opción: 4
```

```
Casillero 893   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 894   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 895   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 896   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 897   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 898   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 899   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 900   |   Piso 6   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 901   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 902   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 903   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 904   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 905   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 906   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 907   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 908   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 909   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 910   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 911   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
Casillero 912   |   Piso 7   |   Estado: disponible   |   Ocupante: Ninguno
```

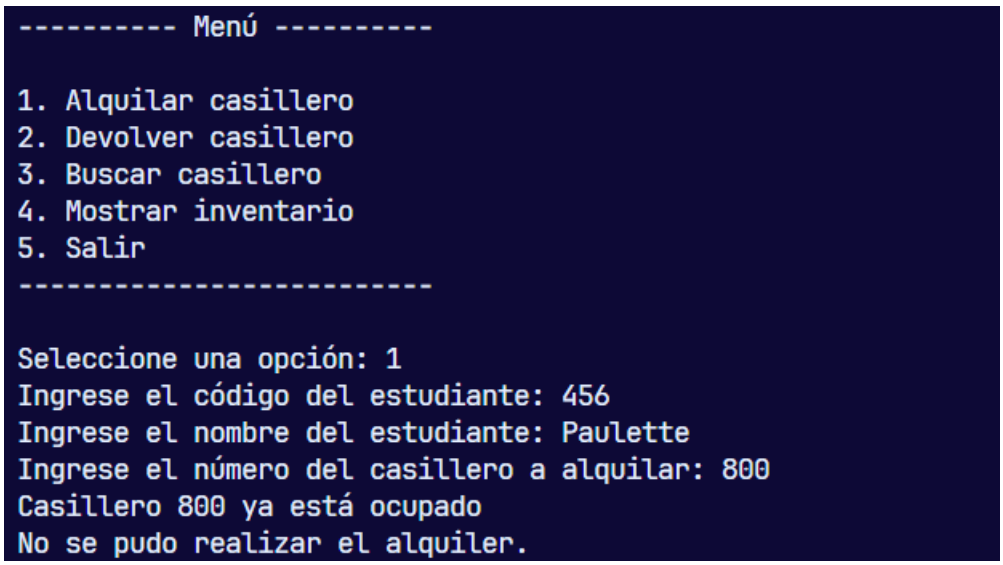
En esta prueba se muestra el inventario de casilleros con su respectiva información, como el número, piso, estado, y ocupante.

5.2. Prueba 2



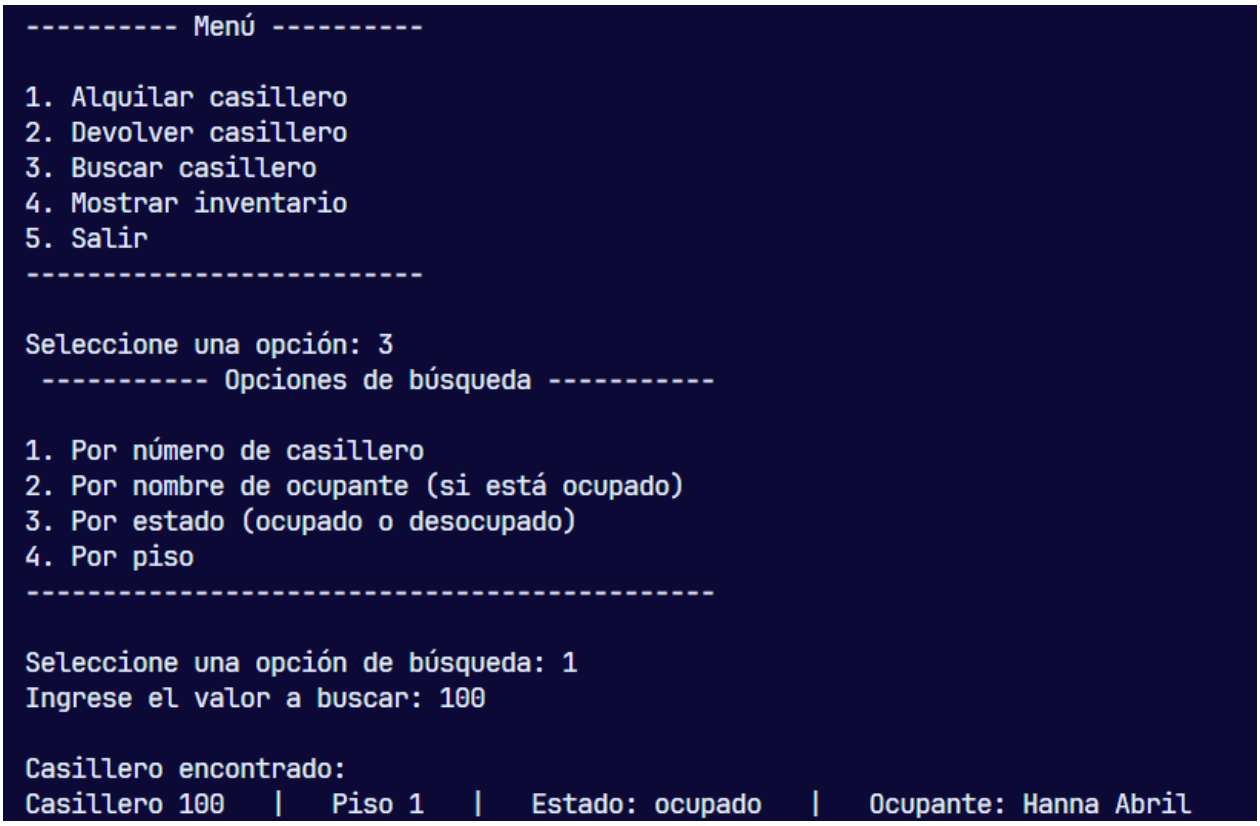
En esta prueba se demuestra la funcionalidad del método para prestar casilleros, el cual pide el nombre y el código del estudiante además del casillero que se le va a prestar.

5.3. Prueba 3



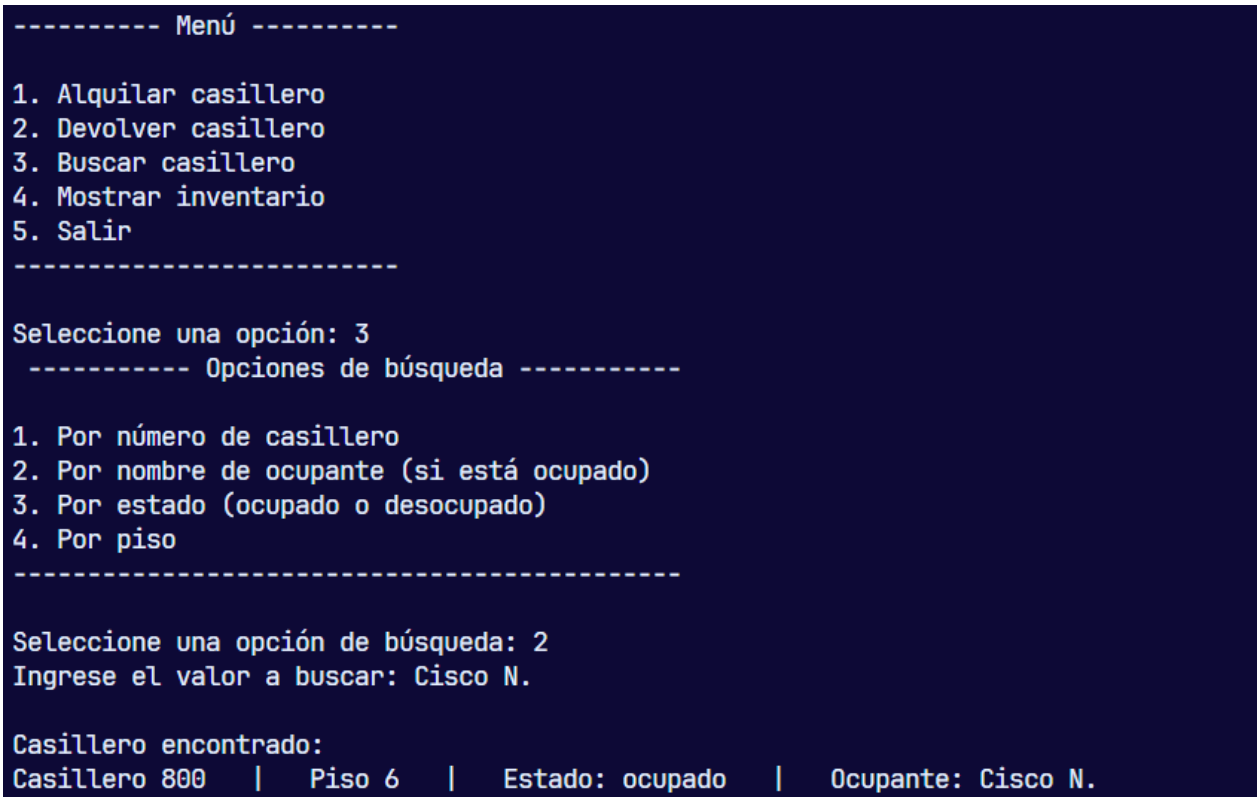
En esta prueba se demuestra la funcionalidad del método para prestar casilleros, el cual pide el nombre y el código del estudiante además del casillero que se le va a prestar. La diferencia es que aquí se intenta pedir prestado un casillero que ya esta en uso, por lo que el sistema no permite prestar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

5.4. Prueba 4



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



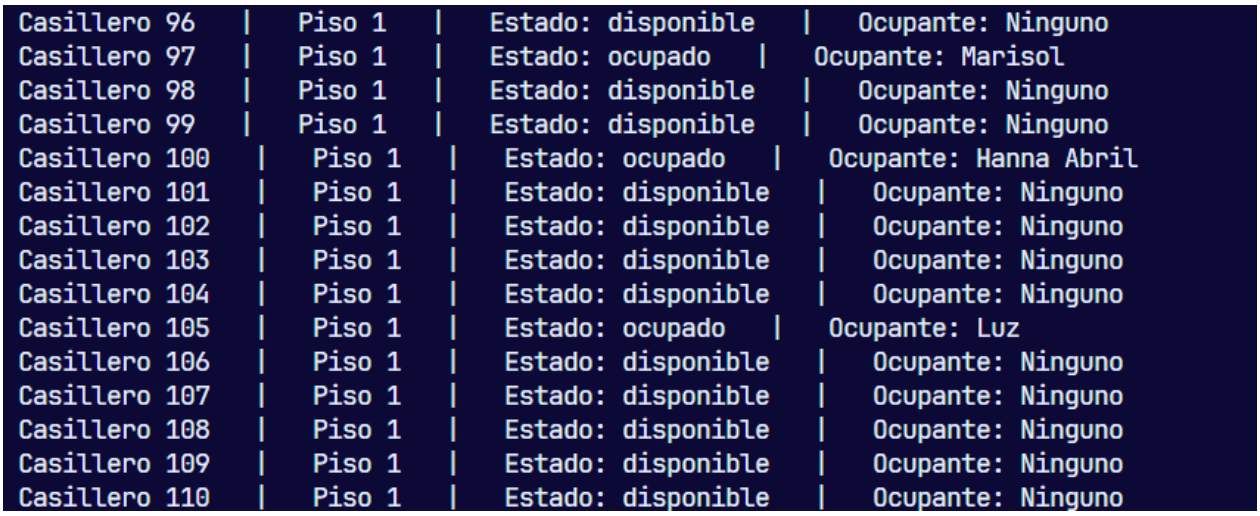
En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el nombre del ocupante del casillero, el cual al ser encontrado arroja la información correspondiente de este. Tener en cuenta que se debe colocar el nombre exacto a como se coloco al momento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

5.6. Prueba 6

```
---------- Menú ----------

1. Alquilar casillero
2. Devolver casillero
3. Buscar casillero
4. Mostrar inventario
5. Salir
--------------------------

Seleccione una opción: 3
 ----------- Opciones de búsqueda -----------

1. Por número de casillero
2. Por nombre de ocupante (si está ocupado)
3. Por estado (ocupado o desocupado)
4. Por piso
-------------------------------------------

Seleccione una opción de búsqueda: 3
 --- Submenú para búsqueda por estado ---
-------------------------------------------
1. Casilleros ocupados
2. Casilleros desocupados
-------------------------------------------

Seleccione una opción de estado: 1

Casilleros encontrados:
Casillero 10   |   Piso 1   |   Estado: ocupado   |   Ocupante: Ricardo
Casillero 20   |   Piso 1   |   Estado: ocupado   |   Ocupante: Laura
Casillero 97   |   Piso 1   |   Estado: ocupado   |   Ocupante: Marisol
Casillero 100   |   Piso 1   |   Estado: ocupado   |   Ocupante: Hanna Abril
Casillero 105   |   Piso 1   |   Estado: ocupado   |   Ocupante: Luz
Casillero 600   |   Piso 4   |   Estado: ocupado   |   Ocupante: Issa
Casillero 800   |   Piso 6   |   Estado: ocupado   |   Ocupante: Cisco N.
```

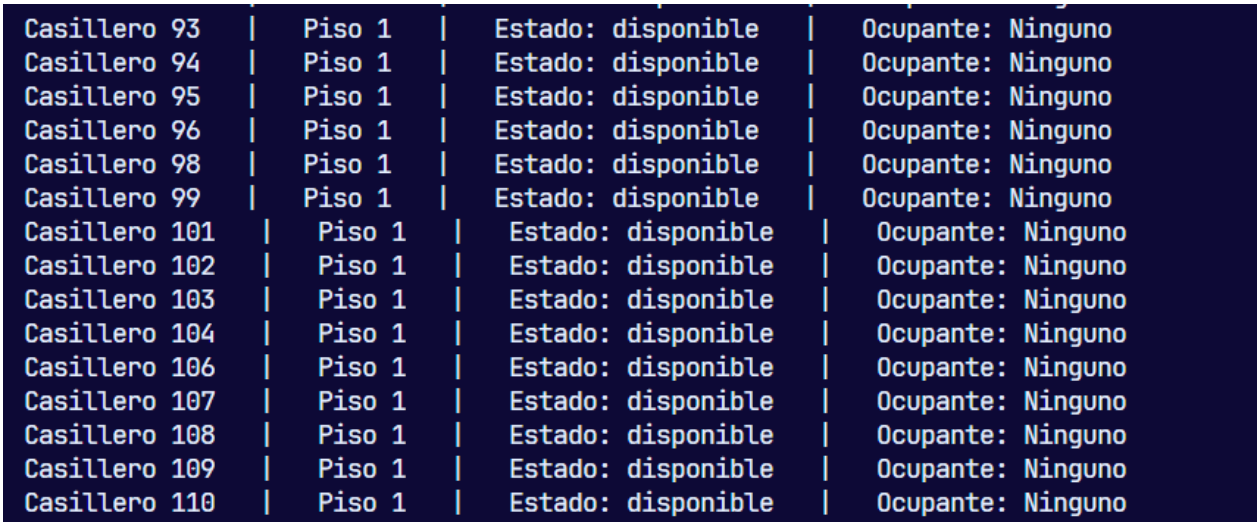
En esta prueba se comprueba el sistema de búsqueda, probando una de sus opciones de búsqueda en este caso por el estado del casillero que en este caso se probó con los que están ocupados, el cual al ser encontrado arroja la información correspondiente de este.

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.

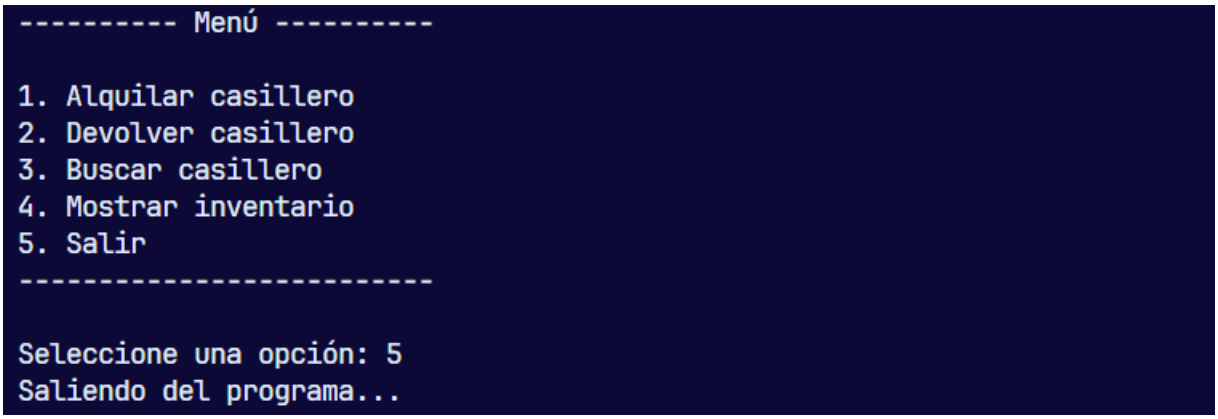
5.9. Prueba 9

```
---------- Menú ----------

1. Alquilar casillero
2. Devolver casillero
3. Buscar casillero
4. Mostrar inventario
5. Salir
--------------------------

Seleccione una opción: 2
Ingrese el número del casillero a devolver: 100
Casillero 100 devuelto
```

```
 ---------- Menú ----------

 1. Alquilar casillero
 2. Devolver casillero
 3. Buscar casillero
 4. Mostrar inventario
 5. Salir
 --------------------------

 Seleccione una opción: 3
  ----------- Opciones de búsqueda -----------

 1. Por número de casillero
 2. Por nombre de ocupante (si está ocupado)
 3. Por estado (ocupado o desocupado)
 4. Por piso
 --------------------------------------------

 Seleccione una opción de búsqueda: 1
 Ingrese el valor a buscar: 100

 Casillero encontrado:
 Casillero 100   |   Piso 1   |   Estado: disponible   |   Ocupante: Ninguno
```

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



Se prueba la salida del sistema.

6. 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.

7. 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.

8. 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.