

TITULO DE LABORATORIO
PARCIAL (CASILLEROS)

HANNA KATHERINE ABRIL GÓNGORA

UNIVERSIDAD MANUELA BELTRÁN

PROGRAMACIÓN ORIENTADA A OBJETOS

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_e
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.estado} | Ocupante: {ocupante}"
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 == estado]
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.numero == valor), None)
58         elif criterio == '2':
59             return next((casillero for casillero in self.casilleros if casillero.estado == valor), None)
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 == valor]
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.

```

76     def mostrar_menu(self):
77         while True:
78             print("----- Menú -----")
79             print("")
80             print("1. Alquilar casillero")
81             print("2. Devolver casillero")
82             print("3. Buscar casillero")
83             print("4. Mostrar inventario")
84             print("5. Salir")
85             print("-----")
86             print("")
87             opcion = input("Seleccione una opción: ")
88             match opcion:
89                 case '1':
90                     codigo_estudiante = input("Ingrese el código del estudiante: ")
91                     nombre_estudiante = input("Ingrese el nombre del estudiante: ")
92                     numero_casillero = int(input("Ingrese el número del casillero a alquilar: "))
93                     casillero = self.inventario.buscar_casillero('1', numero_casillero)
94                     if casillero is not None:
95                         if casillero.alquilar(codigo_estudiante, nombre_estudiante):
96                             print("Alquiler realizado con éxito.")
97                         else:
98                             print("No se pudo realizar el alquiler.")
99                     else:
100                         print("No se encontró ningún casillero con ese número.")
101                 case '2':
102                     numero_casillero = int(input("Ingrese el número del casillero a devolver: "))
103                     casillero = self.inventario.buscar_casillero('1', numero_casillero)
104                     if casillero is not None:
105                         if casillero.devolver():
106                             print("Devolución realizada con éxito.")
107                         else:
108                             print("No se pudo realizar la devolución.")
109                     else:
110                         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_búsqueda = input("Seleccione una opción de búsqueda: ")
121             match opcion_búsqueda:
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_casilleros_estado("ocupado")
133                         case '2':
134                             casilleros_encontrados = self.inventario.buscar_casilleros_estado("desocupado")
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 estado")
144                 case _:
145                     valor_búsqueda = input("Ingrese el valor a buscar: ")
146                     casilleros_encontrados = self.inventario.buscar_casilleros(valor_búsqueda)
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 criterios")

```

```

157         case '4':
158             self.inventario.mostrar_inventario()
159         case '5':
160             print("Saliendo del programa...")
161             break
162         case _:
163             print("Opción no válida. Intente de nuevo.")
164
165
166 main = Main()
167 main.mostrar_menu()
168

```

2. EXPLICACION PARTE AGREGADA DEL PARCIAL.

Ejercicio para implementar en el parcial: Asignar priorización de casilleros, por ejemplo, por edades, cursos específicos o prioridades particulares. Generar función con alertas en los que indiquen alertas en el alquiler de los casilleros o cualquier alerta considerada necesaria.

```

83 class Main:
84     def __init__(self):
85         self.inventario = Inventario()
86
87     def mostrar_menu(self):
88         while True:
89             print("----- Menú -----")
90             print("")
91             print("1. Alquilar casillero")
92             print("2. Devolver casillero")
93             print("3. Buscar casillero")
94             print("4. Mostrar inventario")
95             print("5. Salir")
96             print("-----")
97             print("")
98             opcion = input("Seleccione una opción: ")
99             match opcion:
100                 case '1':
101                     print('Seleccione el tipo de estudiante:
102
103                     1. Con discapacidad (Visual, auditiva, silla de ruedas, entre otras)
104                     2. Estudiante de ingeniería
105                     3. Estudiante de medicina
106                     3. Otros ')
107                     print("")
108                     prioridad= input('Ingrese el numero correspondiente al tipo de estudiante: ')
109
110                     match prioridad:
111

```

```

112         case '1':
113             print('''
114             Por motivos de seguridad unicamente se asignaran casilleros del primer o segundo piso
115             y de los niveles bajos.
116
117             Tener en cuenta:
118
119             Primer piso: 1-50
120             Segundo piso: 151-200''')
121             print("")
122             codigo_estudiante = input("Ingrese el código del estudiante: ")
123             nombre_estudiante = input("Ingrese el nombre del estudiante: ")
124             numero_casillero = int(input("Ingrese el número del casillero a alquilar tener en cuenta los parametros anteriores: "))
125             casillero = self.inventario.buscar_casillero('1', numero_casillero)
126             if casillero is not None:
127                 if casillero>=1 and casillero<=50:
128                     if casillero>= 151 and casillero<=200:
129                         if casillero.alquilar(codigo_estudiante, nombre_estudiante):
130                             print("Alquiler realizado con éxito.")
131                         else:
132                             print("No se pudo realizar el alquiler.")
133                     else:
134                         print("No se encontró ningún casillero con ese número.")
135
136         case '2' :
137             print('''
138             Por motivos de comodidad los estudiantes de ingenieria tienen acceso a los casilleros
139             de los pisos 4 - 5, debido a la ubicacion de los laboratorios de ingenieria.
140
141             Tener en cuenta:
142
143             Cuarto piso: 600 - 750
144             Quinto piso: 751 - 900''')
145             print("")
146             codigo_estudiante = input("Ingrese el código del estudiante: ")
147             nombre_estudiante = input("Ingrese el nombre del estudiante: ")
148             numero_casillero = int(input("Ingrese el número del casillero a alquilar tener en cuenta los parametros anteriores: "))
149             casillero = self.inventario.buscar_casillero('1', numero_casillero)
150             if casillero is not None:
151                 if casillero>=600 and casillero<=900:
152                     if casillero.alquilar(codigo_estudiante, nombre_estudiante):
153                         print("Alquiler realizado con éxito.")
154                     else:
155                         print("No se pudo realizar el alquiler.")
156                 else:
157                     print("No se encontró ningún casillero con ese número.")

```

```

159         case '3':
160             print('''
161             Por motivos de comodidad los estudiantes de medicina tienen acceso a los casilleros
162             de los pisos 6 - 7, debido a la ubicacion de los laboratorios y consultorios de medicina.
163
164             Tener en cuenta:
165
166             Sexto piso: 901 - 1050
167             Séptimo piso: 1051 - 1200''')
168             print("")
169             codigo_estudiante = input("Ingrese el código del estudiante: ")
170             nombre_estudiante = input("Ingrese el nombre del estudiante: ")
171             numero_casillero = int(input("Ingrese el número del casillero a alquilar tener en cuenta los parametros anteriores: "))
172             casillero = self.inventario.buscar_casillero('1', numero_casillero)
173             if casillero is not None:
174                 if casillero>=901 and casillero<=1200:
175                     if casillero.alquilar(codigo_estudiante, nombre_estudiante):
176                         print("Alquiler realizado con éxito.")
177                     else:
178                         print("No se pudo realizar el alquiler.")
179                 else:
180                     print("No se encontró ningún casillero con ese número.")
181

```



```

182         case '4':
183             print('')
184             Los estudiantes tienen accesos a todos los casilleros menos los siguientes:
185
186             Tener en cuenta no se pueden pedir los siguientes casilleros:
187
188             Primer piso: 1-50
189             Segundo piso: 151-200
190             Cuarto piso: 600 - 750
191             Quinto piso: 751 -900
192             Sexto piso: 901 - 1050
193             Septimo piso: 1051 - 1200'''
194             print('')
195             codigo_estudiante = input("Ingrese el código del estudiante: ")
196             nombre_estudiante = input("Ingrese el nombre del estudiante: ")
197             numero_casillero = int(input("Ingrese el número del casillero a alquilar tener en cuenta los parametros anteriores: "))
198             casillero = self.inventario.buscar_casillero('1', numero_casillero)
199             if casillero is not None:
200                 if casillero>=51 and casillero<=150:
201                     if casillero>= 200 and casillero<=559:
202                         if casillero.alquilar(codigo_estudiante, nombre_estudiante):
203                             print("Alquiler realizado con éxito.")
204                         else:
205                             print("No se pudo realizar el alquiler.")
206             else:
207                 print("No se encontró ningún casillero con ese número.")
208

```

En esta primera parte de la implementación del código para definir las prioridades lo que se hizo fue agregar un menú con condicionales, se definieron 4 tipos de estudiantes (con discapacidad, de ingeniería, de medicina, otros), cada estudiante tiene una asignación de casillero respectivos según conveniencia por ejemplo las personas de discapacidad solo tienen casilleros asignados en los dos primeros pisos, los estudiantes de ingeniería y de medicina tienen los casilleros asignados a los pisos en donde están ubicados sus laboratorios respectivamente. Esto se definió mediante intervalos, si la petición del casillero se realizo en el intervalo correspondiente a el tipo de estudiante el préstamo será exitoso, si no, no se podrá realizar el préstamo de estos.

```

210         case '2':
211             numero_casillero = int(input("Ingrese el número del casillero a devolver: "))
212             meses_prestado=input("Ingrese la cantidad de meses que se le presto el casillero: ")
213             casillero = self.inventario.buscar_casillero('1', numero_casillero)
214             if meses_prestado<5:
215                 if casillero is not None:
216                     if casillero.devolver():
217                         print("Devolución realizada con éxito.")
218                     else:
219                         print("No se pudo realizar la devolución.")
220             if meses_prestado>5:
221                 if casillero is not None:
222                     if casillero.devolver():
223                         print("Devolución realizada con éxito.")
224                         print("; Se le aplicara una multa por no devolver el casillero a tiempo !")
225             else:
226                 print("No se encontró ningún casillero con ese número.")
227

```

Respecto a la limitación del préstamo de casilleros, se le pregunta al usuario, cuantos meses tuvo en su poder el casillero y se definió una condición en donde se especificara que si el usuario tubo el casillero por menos de 5 meses la devolución se realizara sin ningún problema ni cargo adicional, pero si el usuario tubo el casillero por mas de 5 meses en ese momento se le muestra un mensaje de que la devolución fue exitosa pero se le aplicara una multa por no entregar el

casillero a tiempo.

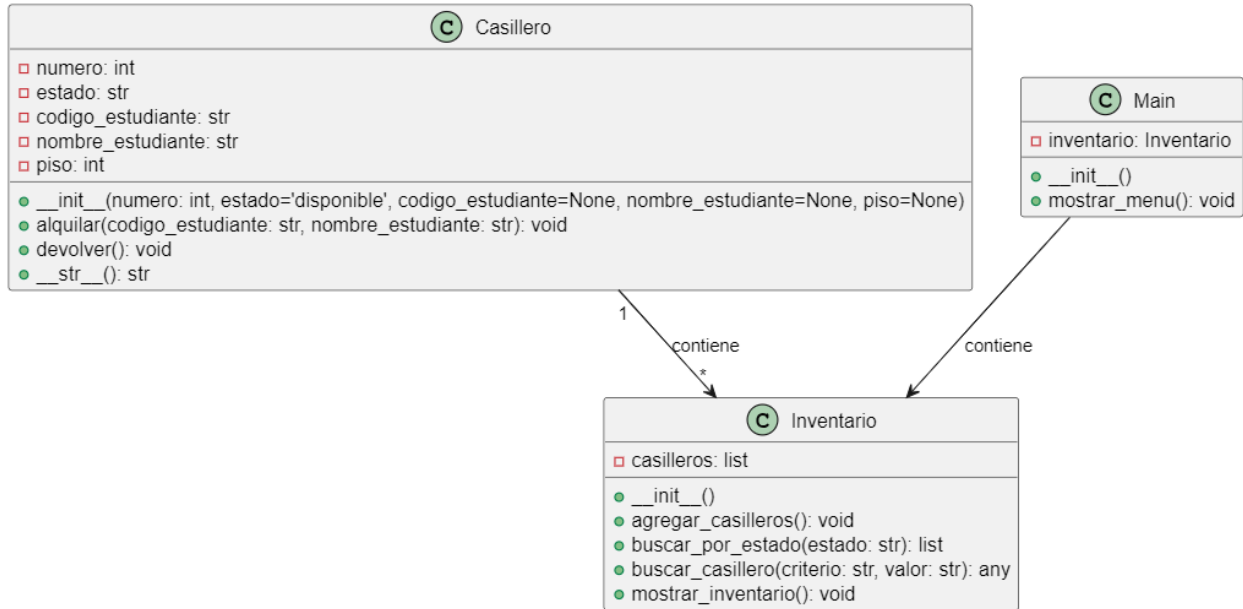
3. EXPLICACION DEL CODIGO UML

```
1  @startuml Casilleros
2
3  class Casillero {
4      - numero: int
5      - estado: str
6      - codigo_estudiante: str
7      - nombre_estudiante: str
8      - piso: int
9      + __init__(numero: int, estado='disponible', codigo_estudiante=None, nombre_estudiante=None, piso=None)
10     + alquilar(codigo_estudiante: str, nombre_estudiante: str): void
11     + devolver(): void
12     + __str__(): str
13 }
14
15 class Inventario {
16     - casilleros: list
17     + __init__()
18     + agregar_casilleros(): void
19     + buscar_por_estado(estado: str): list
20     + buscar_casillero(criterio: str, valor: str): any
21     + mostrar_inventario(): void
22 }
23
24 class Main {
25     - inventario: Inventario
26     + __init__()
27     + mostrar_menu(): void
28 }
29
30 Casillero "1" --> "*" Inventario : contiene
31 Main --> Inventario : contiene
32
33 @enduml
34
```

- **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.

4. DIAGRAMA DEL CODIGO EN UML



5. 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.

6. 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

```
----- Menú -----  
  
1. Alquilar casillero  
2. Devolver casillero  
3. Buscar casillero  
4. Mostrar inventario  
5. Salir  
-----  
  
Seleccione una opción: 1  
Ingrese el código del estudiante: 1006258532  
Ingrese el nombre del estudiante: Hanna Abril  
Ingrese el número del casillero a alquilar: 100  
Casillero 100 alquilado a Hanna Abril
```

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

```
----- Menú -----  
  
1. Alquilar casillero  
2. Devolver casillero  
3. Buscar casillero  
4. Mostrar inventario  
5. Salir  
-----  
  
Seleccione una opción: 1  
Ingrese el código del estudiante: 456  
Ingrese el nombre del estudiante: Paulette  
Ingrese el número del casillero a alquilar: 800  
Casillero 800 ya está ocupado  
No se pudo realizar el alquiler.
```

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 está en uso, por lo que el sistema no permite prestar ese casillero.

5.4. Prueba 4

```
----- 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: ocupado | Ocupante: Hanna Abril
```

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

```
----- 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: 2  
Ingrese el valor a buscar: Cisco N.  
  
Casillero encontrado:  
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 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 colocó al momento de pedir el casillero. Se piensa más adelante establecer un método de coincidencias para que 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

Casillero 96	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 97	Piso 1	Estado: ocupado	Ocupante: Marisol
Casillero 98	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 99	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 100	Piso 1	Estado: ocupado	Ocupante: Hanna Abril
Casillero 101	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 102	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 103	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 104	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 105	Piso 1	Estado: ocupado	Ocupante: Luz
Casillero 106	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 107	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 108	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 109	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 110	Piso 1	Estado: disponible	Ocupante: Ninguno

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

Casillero 93	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 94	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 95	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 96	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 98	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 99	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 101	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 102	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 103	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 104	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 106	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 107	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 108	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 109	Piso 1	Estado: disponible	Ocupante: Ninguno
Casillero 110	Piso 1	Estado: disponible	Ocupante: Ninguno

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

```
----- Menú -----  
1. Alquilar casillero  
2. Devolver casillero  
3. Buscar casillero  
4. Mostrar inventario  
5. Salir  
-----  
  
Seleccione una opción: 5  
Saliendo del programa...
```

Se prueba la salida del sistema.

7. 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 sería más funcional el programa respecto a préstamos y devoluciones. Además de eso para hacer más 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 está ocupado o no, además de eso se puede buscar por el nombre del propietario siempre y cuando este esté 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 más de 1000 casilleros al igual que nuestro código.

8. 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, código_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.

9. CONCLUSIONES

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.