```python
class Usuario:
    def __init__(self, nombre, direccion):
        self.nombre = nombre  # Encapsulamiento
        self.direccion = direccion

class Catalogo:
    def __init__(self):
        self.libros = []  # Lista para almacenar libros en el catálogo

    def agregar(self, libro):
        self.libros.append(libro)

    def eliminar(self, libro):
        if libro in self.libros:
            self.libros.remove(libro)

    def buscar(self, titulo):
        return [libro for libro in self.libros if titulo.lower() in libro.lower()]

class LibrosDisponibles:
    def __init__(self, extension, disponibilidad):
        self.extension = extension  # Encapsulamiento
        self.disponibilidad = disponibilidad

class Prestamo:
    def __init__(self, nombre, direccion, correo, numero_cuenta):
        self.nombre = nombre
        self.direccion = direccion
        self.correo = correo
        self.numero_cuenta = numero_cuenta

    def verificar_penalizaciones(self):
        """Simula la verificación de penalizaciones por retrasos."""
        try:
            print(f"Verificando penalizaciones para {self.nombre}...")
        except Exception as e:
            print(f"Error al verificar penalizaciones: {e}")

# Implementación según los casos de uso
if __name__ == "__main__":
    # Encargado de biblioteca: Check Out, Check Available Books, Check Penalties
    encargado = Usuario("Encargado Biblioteca", "Calle Biblioteca 456")
    catalogo = Catalogo()
    catalogo.agregar("El Principito")
    catalogo.agregar("Cien Años de Soledad")

    print("Libros disponibles:", catalogo.libros)

    resultados = catalogo.buscar("principito")
    print("Resultados de búsqueda:", resultados)
```

```
prestamo1 = Prestamo("Carlos Perez", "Av. Siempre Viva 123", "carlos@example.com", 12345)
prestamo1.verificar_penalizaciones()

# Administrador: Actualizar Catálogo
print("Actualizando el catálogo...")
catalogo.eliminar("Cien Años de Soledad")
print("Catálogo actualizado:", catalogo.libros)
```