INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

CARRERA:

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Materia:

Paradigmas de programación

Profesor: García Floriano Andrés

Alumno:

Escobar Rodriguez Alfonso 3CV1

Fecha de entrega: 11 de junio del 2024.

Codigo

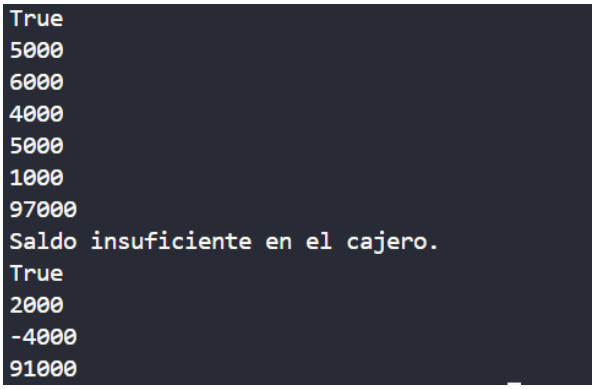
```
from abc import ABC, abstractmethod

class SaldoEfectivoInsuficiente(Exception):
    pass

class SaldoCuentaInsuficiente(Exception):
    pass

class CuentaBase(ABC):
    def __init__(self, numero_cuenta, nombre, saldo):
        self.numero_cuenta = numero_cuenta
        self.nombre = nombre
        self.saldo = saldo

    @abstractmethod
    def depositar(self, monto):
        pass

    @abstractmethod
    def retirar(self, monto):
        pass

    @abstractmethod
    def transferir(self, otra_cuenta, monto):
        pass

class Cuenta(CuentaBase):
    def depositar(self, monto):
        self.saldo += monto

    def retirar(self, monto):
        if self.saldo >= monto:
            self.saldo -= monto
        else:
            raise SaldoCuentaInsuficiente("Saldo insuficiente en la cuenta.")

    def transferir(self, otra_cuenta, monto):
        if self.saldo >= monto:
            self.saldo -= monto
            otra_cuenta.depositar(monto)
        else:
            raise     SaldoCuentaInsuficiente("Saldo     insuficiente     para     la
transferencia.")

class CuentaVIP(CuentaBase):
    def __init__(self, numero_cuenta, nombre, saldo, linea_credito):
        super().__init__(numero_cuenta, nombre, saldo)
        self.linea_credito = linea_credito

    def depositar(self, monto):
        self.saldo += monto

    def retirar(self, monto):
        if self.saldo + self.linea_credito >= monto:
            self.saldo -= monto
```

```
        else:
            raise SaldoCuentaInsuficiente("Saldo insuficiente en la cuenta y línea
de crédito.")

    def transferir(self, otra_cuenta, monto):
        if self.saldo + self.linea_credito >= monto:
            self.saldo -= monto
            otra_cuenta.depositar(monto)
        else:
            raise SaldoCuentaInsuficiente("Saldo insuficiente para la transferencia,
incluso con la línea de crédito.")

class CajeroAutomatico:
    def __init__(self, saldo_efectivo):
        self.saldo_efectivo = saldo_efectivo
        self.cuentas = {}
        self.cuenta_actual = None

    def autenticar(self, numero_cuenta, nombre):
        cuenta = self.cuentas.get(numero_cuenta)
        if cuenta and cuenta.nombre == nombre:
            self.cuenta_actual = cuenta
            return True
        else:
            return False

    def agregar_cuenta(self, cuenta):
        if isinstance(cuenta, CuentaBase):
            self.cuentas[cuenta.numero_cuenta] = cuenta

    def ver_saldo(self):
        if self.cuenta_actual:
            return self.cuenta_actual.saldo
        else:
            return "No hay cuenta autenticada."

    def depositar_a_propia(self, monto):
        if self.cuenta_actual:
            self.cuenta_actual.depositar(monto)
        else:
            return "No hay cuenta autenticada."

    def depositar_a_otra(self, numero_cuenta_destino, monto):
        if self.cuenta_actual:
            cuenta_destino = self.cuentas.get(numero_cuenta_destino)
            if cuenta_destino:
                cuenta_destino.depositar(monto)
            else:
                return "Cuenta destino no encontrada."
        else:
            return "No hay cuenta autenticada."

    def transferir_a_otra(self, numero_cuenta_destino, monto):
        if self.cuenta_actual:
            cuenta_destino = self.cuentas.get(numero_cuenta_destino)
            if cuenta_destino:
                self.cuenta_actual.transferir(cuenta_destino, monto)
            else:
                return "Cuenta destino no encontrada."
```

```
        else:
            return "No hay cuenta autenticada."

    def retirar_efectivo(self, monto):
        if self.cuenta_actual:
            if self.saldo_efectivo >= monto:
                try:
                    self.cuenta_actual.retirar(monto)
                    self.saldo_efectivo -= monto
                except SaldoCuentaInsuficiente as e:
                    return str(e)
            else:
                raise SaldoEfectivoInsuficiente("Saldo insuficiente en el cajero.")
        else:
            return "No hay cuenta autenticada."

# Ejemplo de uso del cajero automático
cajero = CajeroAutomatico(saldo_efectivo=100000)

# Crear y agregar cuentas
cuenta1 = Cuenta(numero_cuenta="123", nombre="Alice", saldo=5000)
cuenta2 = Cuenta(numero_cuenta="456", nombre="Bob", saldo=3000)
cuenta_vip    =    CuentaVIP(numero_cuenta="789",    nombre="Charlie",    saldo=2000,
linea_credito=5000)
cajero.agregar_cuenta(cuenta1)
cajero.agregar_cuenta(cuenta2)
cajero.agregar_cuenta(cuenta_vip)

# Autenticar usuario
print(cajero.autenticar("123", "Alice"))  # True
print(cajero.ver_saldo())  # 5000

# Depósito a cuenta propia
cajero.depositar_a_propia(1000)
print(cajero.ver_saldo())  # 6000

# Transferencia a otra cuenta
cajero.transferir_a_otra("456", 2000)
print(cajero.ver_saldo())  # 4000
print(cuenta2.saldo)  # 5000

# Retiro de efectivo
try:
    cajero.retirar_efectivo(3000)
    print(cajero.ver_saldo())  # 1000
    print(cajero.saldo_efectivo)  # 97000
except SaldoEfectivoInsuficiente as e:
    print(e)
except SaldoCuentaInsuficiente as e:
    print(e)

# Intento de retiro con saldo insuficiente en el cajero
try:
    cajero.retirar_efectivo(100000)
except SaldoEfectivoInsuficiente as e:
    print(e)
except SaldoCuentaInsuficiente as e:
    print(e)
```

```
# Autenticar usuario VIP
print(cajero.autenticar("789", "Charlie"))  # True
print(cajero.ver_saldo())  # 2000

# Retiro con cuenta VIP y línea de crédito
try:
    cajero.retirar_efectivo(6000)
    print(cajero.ver_saldo())  # -4000
    print(cajero.saldo_efectivo)  # 91000
except SaldoEfectivoInsuficiente as e:
    print(e)
except SaldoCuentaInsuficiente as e:
    print(e)
```

Prueba de Validez



- True: Indica que la autenticación de Alice (cuenta "123") fue exitosa.
- 5000: Muestra el saldo de la cuenta de Alice antes de cualquier operación.
- 4000: Muestra el saldo de la cuenta de Alice después de depositar 1000 unidades.
- 5000: Muestra el saldo de la cuenta de Bob (cuenta "456") después de recibir una transferencia de 2000 unidades desde la cuenta de Alice.
- 1000: Muestra el saldo de la cuenta de Alice después de retirar 3000 unidades.
- 97000: Muestra el saldo de efectivo del cajero después de que Alice retiró 3000 unidades.
- Saldo insuficiente en el cajero.: Excepción lanzada cuando se intenta retirar 100000 unidades del cajero, pero no hay suficiente efectivo.
- True: Indica que la autenticación de Charlie (cuenta VIP "789") fue exitosa.
- 2000: Muestra el saldo de la cuenta VIP de Charlie antes de cualquier operación.
- -4000: Muestra el saldo de la cuenta VIP de Charlie después de retirar 6000 unidades, utilizando la línea de crédito.
- 91000: Muestra el saldo de efectivo del cajero después de que Charlie retiró 6000 unidades