Programação Orientada Objeto

https://dontpad.com/catolicaoop2025

hoje dia 16 de setembro dediquei meu tempo na aula de Programação Orientada para polir e consertar algumas lógicas que estava faltando no meu código, bom consegui resolver a parte de mudança de senha, antes não poderia simplesmente mudar a senha sem ter que confirma ela antes de mudar, mudei também a parte do pix, umas das regras pedidas para esse desenvolvimento foi de que caso mandasse pix para si mesmo gera-se mais saldo sem gastar da própria conta.

bom mudei também meu menu onde estava estático com as informações depois vários espaços para formar um quadrado. fiz de uma forma com que ler as informações depois colocava os espaço automaticamente caso o valor altera se e não perdia a forma Sendo mais dinâmico, e também aparecesse toda vez que o menu de operações fosse aparece a conta contendo seu nome,conta,agência é saldo atual.

```python
import time
class Account:
    def __init__(self, password: int, agency: int, number: int,
balance: float = 0.0, keyPix: str = None, limit: float = 1000.0, name:
str = None):
        self.__password = password
        self.__agency = agency
        self.__number = number
        self.__balance = balance
        self.__keyPix = keyPix
        self.__limit = limit
        self.__transactions = []
        self.__name = name
    @property
    def balance(self):
        return self.__balance
    @property
    def agency(self):
        return self.__agency
    @property
    def number(self):
        return self.__number
    @property
    def keyPix(self):
        return self.__keyPix
    def setKeyPix(self, new_key: str):
        self.__keyPix = new_key
        print(f">> PIX key set to: {new_key}")
        jump
    def removeKeyPix(self):
        if self.__keyPix is None:
            print(">> No PIX key registered to remove.")
        else:
            print(f">> PIX key {self.__keyPix} removed.")
            self.__keyPix = None
        jump
    @property
    def limit(self):
        return self.__limit
    @limit.setter
    def limit(self, new_limit: float):
        self.__limit = new_limit
        self.__transactions.append(f"Limit updated: ${new_limit:.2f}")
        print(f">> Limit successfully updated to ${new_limit:.2f}")
        jump
    @property
    def name(self):
        return self.__name
    def login(self, agency: int, number: int, password: int):
        return self.__agency == agency and self.__number == number and
self.__password == password
    def changePassword(self, current_password: int, new_password: int)
-> bool:
        if current_password != self.__password:
            print(">> Current password is incorrect. Password not
changed.")
            jump
            return False
        self.__password = new_password
        print(">> Password successfully updated!")
        jump
        return True
    def deposit(self, amount: float):
        self.__balance += amount
        self.__transactions.append(f"Deposit: +${amount:.2f}")
        print(f">> Deposit of ${amount:.2f} \nsuccessful. \nBalance:
${self.__balance:.2f}")
```

```
        jump
    def withdraw(self, amount: float):
        if self.__balance - amount < -self.__limit:
            print(">> Withdrawal denied. Limit exceeded!")
            jump
        else:
            self.__balance -= amount
            self.__transactions.append(f"Withdraw: -${amount:.2f}")
            print(f">> Withdrawal of ${amount:.2f} successful.
\nBalance: ${self.__balance:.2f}")
            jump
    def sendPix(self, amount: float, destination: str):
        if self.__keyPix is None:
            print(">> You don't have a PIX key registered. \nPlease
register one first.")
            jump
            return
        if destination == self.__keyPix:
            self.__balance += amount
            self.__transactions.append(f"BUG PIX exploit:
+${amount:.2f} (to yourself)")
            print(f">> PIX exploit detected! \nYou sent to yourself
\nbalance increased by ${amount:.2f}.")
            print(f"New balance: ${self.__balance:.2f}")
            jump
            return
        if self.__balance - amount < -self.__limit:
            print(">> PIX transfer denied. Limit exceeded!")
            jump
        else:
            self.__balance -= amount
            self.__transactions.append(f"PIX sent: -${amount:.2f} to
{destination}")
            print(f">> PIX of ${amount:.2f} sent to {destination}.
\nBalance: ${self.__balance:.2f}")
            jump
    def statement(self):
        print("=== Account Statement ===")
        if not self.__transactions:
            print("No financial transactions yet.")
        else:
            for t in self.__transactions:
                print("-", t)
```

```python
        print(f"Current balance: ${self.__balance:.2f}")
        jump
layout = """
========================================
|                                      |
|          WELCOME TO PYATM            |
|                                      |
========================================
|          [1] Check Balance           |
|          [2] Deposit                 |
|          [3] Withdraw                |
|          [4] Send PIX                |
|          [5] Change Password         |
|          [6] Manage PIX Key          |
|          [7] Adjust Limit            |
|          [8] Exit                    |
========================================
"""
layoutLG = """
========================================
|                                      |
|          WELCOME TO PYATM            |
|                                      |
========================================
"""
jump = "\n========================================\n"
def print_box(account):
    width = 40

    def center_text(text):
        return f"| {text.center(width-4)} |"
    def left_text(label, value):
        text = f"{label}: {value}"
        return f"| {text.ljust(width-4)} |"
    print("=" * width)
    print(center_text(f"Hello, {account.name}"))
    print(left_text("Agency", account.agency))
    print(left_text("Account", account.number))
    print(left_text("Current balance", f"${account.balance:.2f}"))
    print("=" * width)
def homeScreen(account: Account):
    print_box(account)
    time.sleep(1)
    atm_interface(account)
def atm_interface(account: Account):
    while True:
        jump
        print_box(account)
        print(layout)
        option = input("Select an option: ")
        jump
        if option == "1":
            account.statement()
        elif option == "2":
            try:
                value = float(input("Enter deposit amount: "))
                account.deposit(value)
            except ValueError:
                print(">> Invalid amount.")
        elif option == "3":
            try:
                value = float(input("Enter withdrawal amount: "))
                account.withdraw(value)
            except ValueError:
                print(">> Invalid amount.")
        elif option == "4":
            try:
                value = float(input("Enter PIX amount: "))
                dest = input("Enter destination PIX key: ")
                account.sendPix(value, dest)
            except ValueError:
                print(">> Invalid amount.")
        elif option == "5":
            print("=== Change Password ===")
            try:
                current = int(input("Enter current password: "))
                new_pass = int(input("Enter new password: "))
                if account.changePassword(current, new_pass):
                    print(">> Use your new password next login.")
                else:
                    print(">> Password not changed.")
            except ValueError:
                print(">> Password must be numeric.")
        elif option == "6":
            print("=== PIX Key Management ===")
            if account.keyPix is None:
                print("No PIX key registered yet.")
            else:
                print(f"Current PIX key: {account.keyPix}")
            print("[1] Register/Change PIX key")
            print("[2] Remove PIX key")
            print("[3] Cancel")
            choice = input("Choose an option: ")
            if choice == "1":
                new_key = input("Enter new PIX key (email or phone): ")
                account.setKeyPix(new_key)
            elif choice == "2":
                account.removeKeyPix()
            elif choice == "3":
                print(">> PIX key operation canceled.")
            else:
                print(">> Invalid option.")
        elif option == "7":
            try:
                new_limit = float(input("Enter new limit: "))
                account.limit = new_limit
            except ValueError:
                print(">> Invalid limit value.")
        elif option == "8":
            print(">> Thank you for using PYATM. Goodbye!")
            break
        else:
            print(">> Invalid option, try again.")
        time.sleep(1)
if __name__ == "__main__":
    acc = Account(password=123, agency=1, number=101, balance=800.0,
name="Victor Emanuel")
    print(layoutLG)
    while True:
        agency = int(input("Agency: "))
        number = int(input("Account number: "))
        password = int(input("Password: "))
        if acc.login(agency, number, password):
            print(">> Login successful!\n")
            jump
            homeScreen(acc)
            break
        else:
            jump
            print(">> Invalid login. Try again.")
            print(layoutLG)
```