In [ ]:

```
#Banking System (Encapsulation and Private Attributes)
#You are working on a simple banking system. Implement a BankAccount class that
#● Private attributes: __balance (initially set to 0) and __account_number.
#● A method deposit(amount) to deposit money into the account (must ensure amoun
#● A method withdraw(amount) to withdraw money (must ensure balance is sufficien
#● A method get_balance() to check the current balance (via a public method).
#● Demonstrate that the private attributes cannot be accessed directly from outs
#Task:
#1. Create a BankAccount instance with an account number and deposit/withdraw mo
#2. Try to access the private __balance directly and observe the result.
#3. Add a method transfer_money() that allows transferring money between two acc

class BankAccount:
    def __init__(self, account_number):
        self.__account_number = account_number
        self.__balance = 0

    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
            print(f"₹{amount} deposited to Account {self.__account_number}.")
        else:
            print("Deposit amount must be greater than 0.")

    def withdraw(self, amount):
        if amount > self.__balance:
            print("Insufficient balance.")
        elif amount <= 0:
            print("Withdrawal amount must be greater than 0.")
        else:
            self.__balance -= amount
            print(f"₹{amount} withdrawn from Account {self.__account_number}.")

    def get_balance(self):
        return self.__balance

    def transfer_money(self, target_account, amount):
        if not isinstance(target_account, BankAccount):
            print("Target must be a BankAccount instance.")
            return
        if amount <= 0:
            print("Transfer amount must be greater than 0.")
            return
        if amount > self.__balance:
            print("Transfer failed: Insufficient balance.")
            return

        self.withdraw(amount)
        target_account.deposit(amount)
        print(f"₹{amount} transferred from Account {self.__account_number} to Ac

    def get_account_number(self):
        return self.__account_number

account1 = BankAccount("ACC001")
account2 = BankAccount("ACC002")

account1.deposit(1000)
```

```
account1.withdraw(200)
print("Account1 Balance:", account1.get_balance())

try:
    print(account1.__balance)
except AttributeError as e:
    print("Error accessing private attribute __balance:", e)

account1.transfer_money(account2, 500)

print("Account1 Balance after transfer:", account1.get_balance())
print("Account2 Balance after receiving:", account2.get_balance())

print("Accessing private balance with name mangling:", account1._BankAccount__ba
```