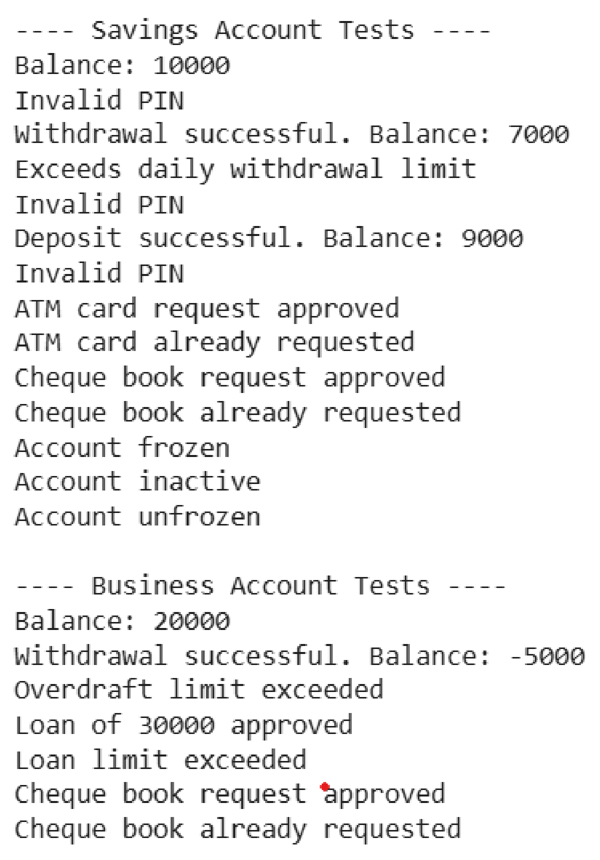
```
# Bank Customer Wise Test – Complete Python Code
# Covers Encapsulation, Inheritance, Method Overriding, super(), and Validations

# --------------------
# Parent Class
# --------------------
class BankAccount:
    def __init__(self, name, balance):
        self._name = name              # protected
        self._balance = balance        # protected
        self._active = True
        self._cheque_book_requested = False

    def check_balance(self):
        if not self._active:
            return "Account inactive"
        return f"Balance: {self._balance}"

    def deposit(self, amount):
        if amount <= 0:
            return "Invalid deposit amount"
        self._balance += amount
        return f"Deposit successful. Balance: {self._balance}"

    def request_cheque_book(self):
        if self._cheque_book_requested:
            return "Cheque book already requested"
        self._cheque_book_requested = True
        return "Cheque book request approved"


# --------------------
# Savings Account
# --------------------
class SavingsAccount(BankAccount):
    DAILY_LIMIT = 5000

    def __init__(self, name, balance, pin):
        super().__init__(name, balance)   # super() verification
        self.__pin = pin                  # private
        self.__atm_requested = False

    def __validate_pin(self, pin):        # private method
        return self.__pin == pin

    def check_balance(self, pin):
        if not self.__validate_pin(pin):
            return "Invalid PIN"
        return super().check_balance()

    def withdraw(self, amount, pin):
        if not self.__validate_pin(pin):
            return "Invalid PIN"
        if not self._active:
            return "Account inactive"
        if amount > self._balance:
            return "Insufficient balance"
        if amount > self.DAILY_LIMIT:
            return "Exceeds daily withdrawal limit"
        self._balance -= amount
        return f"Withdrawal successful. Balance: {self._balance}"

    def deposit(self, amount, pin):
        if not self.__validate_pin(pin):
            return "Invalid PIN"
        return super().deposit(amount)

    def request_atm_card(self):
        if self.__atm_requested:
            return "ATM card already requested"
        self.__atm_requested = True
        return "ATM card request approved"

    def freeze_account(self):
        if not self._active:
            return "Account already frozen"
        self._active = False
        return "Account frozen"

    def unfreeze_account(self):
        if self._active:
            return "Account already active"
        self._active = True
        return "Account unfrozen"



# --------------------
# Business Account
# --------------------
class BusinessAccount(BankAccount):
    OVERDRAFT_LIMIT = 10000
    LOAN_LIMIT = 50000

    def __init__(self, business_name, balance):
        super().__init__(business_name, balance)

    def withdraw(self, amount):
        if not self._active:
            return "Account inactive"
        if amount > (self._balance + self.OVERDRAFT_LIMIT):
            return "Overdraft limit exceeded"
        self._balance -= amount
        return f"Withdrawal successful. Balance: {self._balance}"

    def request_loan(self, amount):
        if amount > self.LOAN_LIMIT:
            return "Loan limit exceeded"
        return f"Loan of {amount} approved"



# --------------------
# COMPLETE TEST FLOW

# --------------------

print("---- Savings Account Tests ----")

sa = SavingsAccount("Nikhitha", 10000, 1234)


print(sa.check_balance(1234))

print(sa.check_balance(1111))

print(sa.withdraw(3000, 1234))

print(sa.withdraw(6000, 1234))

print(sa.withdraw(1000, 1111))

print(sa.deposit(2000, 1234))

print(sa.deposit(2000, 1111))

print(sa.request_atm_card())

print(sa.request_atm_card())

print(sa.request_cheque_book())

print(sa.request_cheque_book())

print(sa.freeze_account())

print(sa.withdraw(500, 1234))

print(sa.unfreeze_account())


print("\n---- Business Account Tests ----")

ba = BusinessAccount("ABC Pvt Ltd", 20000)


print(ba.check_balance())

print(ba.withdraw(25000))

print(ba.withdraw(40000))

print(ba.request_loan(30000))

print(ba.request_loan(70000))
print(ba.request_cheque_book())

print(ba.request_cheque_book())


# --------------------

# Encapsulation Tests (Expected to fail if uncommented)

# print(sa.__pin)          # AttributeError

# print(sa.__validate_pin)  # AttributeError
```



```
---- Savings Account Tests ----
Balance: 10000
Invalid PIN
Withdrawal successful. Balance: 7000
Exceeds daily withdrawal limit
Invalid PIN
Deposit successful. Balance: 9000
Invalid PIN
ATM card request approved
ATM card already requested
Cheque book request approved
Cheque book already requested
Account frozen
Account inactive
Account unfrozen

---- Business Account Tests ----
Balance: 20000
Withdrawal successful. Balance: -5000
Overdraft limit exceeded
Loan of 30000 approved
Loan limit exceeded
Cheque book request approved
Cheque book already requested
```