Laporan Project 1

Pemrograman Berbasis Objek

“Bank INI”

Dosen Pengampu :
Erna Kumalasari Nurnawati, S.T., M.T., C.DS

Kelompok :

1. **Jati Kurniawan Yusuf S. (5220411448)**
2. Diki Hendrik Setyawan (5220411435)

1. Overview

A. Pengenalan Sistem Bank INI

Kami membuat sebuah projek system bank yang terdiri dari 2 class induk dan 3 sub-class yaitu User, Query, Customer, Teller dan Admin. User dan Query sebagai class induk dan Customer, Teller, dan Admin sebagai sub-class dari class User.User memiliki id, nama, pin, dan password serta dapat melakukan login dan logout. Class query berfungsi untuk mengkoneksi ke database .Customer dapat melakukan buat akun, depo, withdraw, cek akun, dan saving. Teller dapat melakukan konfirmasi penarikan dan cek saldo customer. Lalu admin dapat melakukan konfirmasi pembuatan akun, cek akun, cek history, dan membuat akun teller.

B. Object Oriented Programming(OOP)

Object Oriented Programming adalah paradigma pemrograman yang menggunakan konsep "objek" untuk mendesain dan mengorganisir kode. Paradigma ini memungkinkan pengembang untuk mengorganisir data dan fungsionalitas ke dalam entitas yang disebut objek. Setiap objek dapat memiliki atribut (data) dan metode (fungsi) yang berkaitan dengannya.

C. Pembuatan Class, Object dan Method

1) Class (Kelas):

Kelas adalah blueprint atau cetak biru untuk menciptakan objek. Ini mendefinisikan atribut (data) dan metode (fungsi) yang akan dimiliki oleh objek yang akan dibuat berdasarkan kelas tersebut. Kelas menyediakan struktur dasar atau template untuk membuat objek dengan karakteristik tertentu. Contoh, jika kita memiliki kelas "Mobil", mungkin memiliki atribut seperti "warna" dan "kecepatan" serta metode seperti "start" dan "stop".

2) Object (Objek):

Objek adalah instance atau perwujudan konkret dari suatu kelas. Objek memiliki atribut dan method yang terkait dengan class nya. Misalnya, jika "Mobil" adalah kelas, maka "Mobil A" dan "Mobil B" adalah objek dari kelas tersebut. Masing-masing objek memiliki atribut yang berbeda (mungkin warna yang berbeda) dan dapat memanggil metode yang sama (misalnya, "start" dan "stop").

3) Method (Metode):

Metode adalah fungsi atau perilaku yang terkait dengan suatu objek atau kelas. Metode didefinisikan di dalam kelas dan dapat diakses melalui objek yang dibuat berdasarkan kelas tersebut. Dalam contoh mobil, metode dapat mencakup "start" untuk menghidupkan mesin atau "stop" untuk mematikannya. Metode juga dapat melakukan manipulasi pada atribut objek atau melakukan operasi tertentu tergantung pada kebutuhan aplikasi.

D. Access Modifier

1) Public

Variabel atau atribut yang memiliki hak akses publik bisa diakses dari mana saja baik dari luar kelas mau pun dari dalam kelas

2) Protected

Variabel atau atribut yang memiliki hak akses protected hanya bisa diakses secara terbatas oleh dirinya sendiri (yaitu di dalam internal kelas), dan juga dari kelas turunannya.

Untuk mendefinisikan atribut dengan hak akses protected, kita harus menggunakan prefix underscore _ sebelum nama variabel.

3) Private

Setiap variabel di dalam suatu kelas yang memiliki hak akses private maka ia hanya bisa diakses di dalam kelas tersebut. Tidak bisa diakses dari luar bahkan dari kelas yang mewarisinya.

Untuk membuat sebuah atribut menjadi private, kita harus menambahkan dua buah underscore sebagai prefix (__) nama atribut.

4) Setter getter

Ia adalah sebuah fungsi yang akan dieksekusi ketika kita mengakses (aksesor) suatu atribut pada suatu kelas, atau fungsi yang dieksekusi ketika hendak mengatur (mutator) suatu atribut pada suatu kelas.

Untuk mendefinisikan accessor (getter), perlu mendefinisikan decorator @property sebelum nama fungsi. Sedangkan untuk mengatur mutator (setter), perlu mendefinisikan descriptor @<nama-atribut>.setter.

E. Inheritance

Konsep pewarisan adalah konsep di mana sebuah kelas atau objek mewariskan sifat dan perilaku kepada kelas lainnya. Kelas yang menjadi “pemberi waris” dinamakan kelas induk atau kelas basis Sedangkan kelas yang menjadi “ahli waris” dinamakan sebagai kelas

turunan.
Kelas turunan akan selalu memiliki sifat dan perilaku yang sama dengan kelas induknya: mulai dari atribut sampai fungsi-fungsinya. Sebaliknya, belum tentu kelas induk memiliki semua atribut dan sifat dari kelas-kelas turunannya

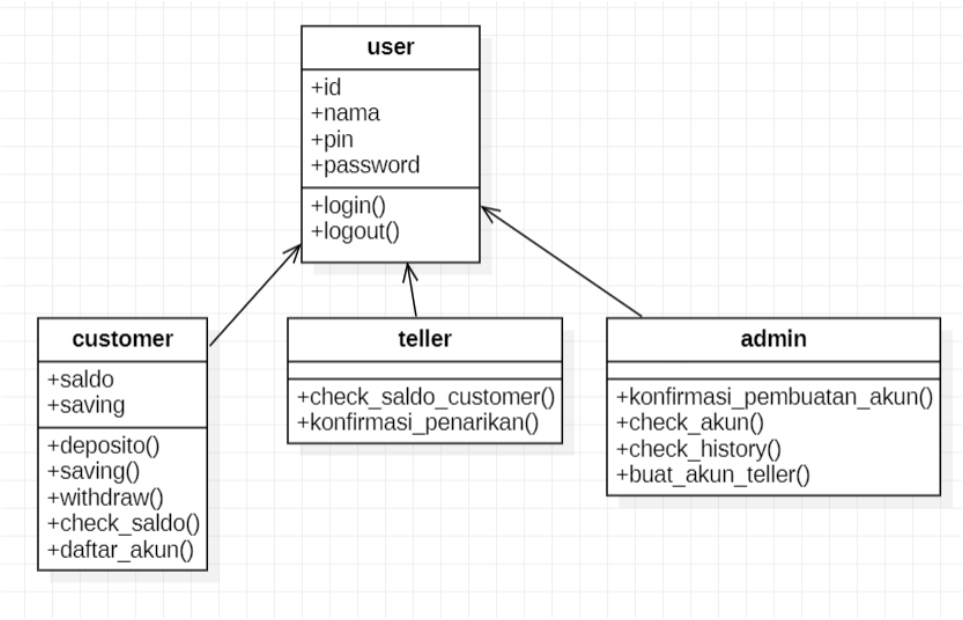
F. Overloading & Overriding
   1. Overloading
      Overloading terjadi ketika sebuah kelas memiliki dua atau lebih metode dengan nama yang sama, tetapi berbeda dalam jumlah atau tipe parameter. Dengan overloading, satu kelas dapat memiliki beberapa metode dengan nama yang sama, tetapi berbeda dalam cara mereka dipanggil.
   2. Overriding
      Overriding terjadi ketika sebuah kelas anak (subclass) menyediakan implementasi ulang dari metode yang sudah didefinisikan di kelas induk (superclass). Metode yang di-overriding di kelas anak harus memiliki nama, tipe pengembalian, dan parameter yang sama dengan metode di kelas induk. Overriding memungkinkan kelas anak untuk memberikan implementasi yang sesuai untuk metode yang diwarisi dari kelas induk.

2. Kebutuhan Penyusunan

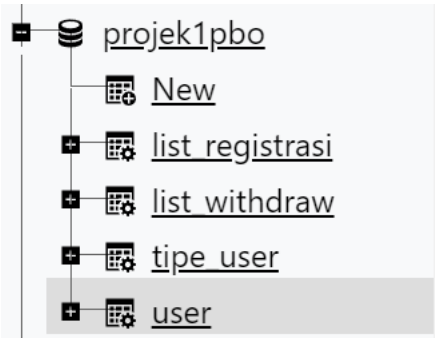
A. Library
   Kami menggunakan 3 library yaitu pyfiglet, random. dan mysql.connector
   a. Pyfiglet
      Pyfiglet adalah modul Python yang memungkinkan Anda membuat teks berformat dengan gaya "figlet" atau seni teks besar menggunakan karakter karakter. Figlet adalah program yang menghasilkan teks dalam bentuk grafis yang dibuat dari karakter ASCII.
   b. Random
      Modul random adalah modul bawaan Python yang menyediakan fungsi-fungsi untuk menghasilkan nilai acak. Dengan menggunakan modul ini, Anda dapat melakukan berbagai operasi yang melibatkan unsur acak dalam skrip atau program Python Anda.
   c. Mysql.connector
      Mysql.connector adalah library Python yang digunakan untuk berinteraksi dengan database MySQL. Library ini menyediakan API

yang mudah digunakan untuk melakukan koneksi, eksekusi query SQL, dan berbagai operasi lainnya terhadap database MySQL

B. Class Diagram



C. Database



a) User

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None			Change Drop More
2	nama	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	pin	int(11)			No	None			Change Drop More
4	password	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	saldo	int(11)			No	None			Change Drop More
6	saving	int(11)			No	None			Change Drop More
7	id_tipe	int(11)			No	None			Change Drop More

b) Tipe_user

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id**	int(11)			No	*None*			Change Drop More
2	**tipe_user**	varchar(50)	*utf8mb4_general_ci*		No	*None*			Change Drop More

c) List_withdraw

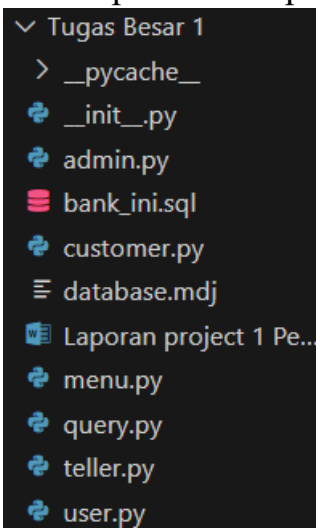
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id**	int(11)			No	*None*			Change Drop More
2	**saldo**	int(11)			No	*None*			Change Drop More
3	**amount**	int(11)			No	*None*			Change Drop More
4	**status**	tinyint(1)			No	*None*			Change Drop More

d) List_registrasi

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**id**	int(11)			No	*None*			Change Drop More
2	**nama**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
3	**pin**	int(11)			No	*None*			Change Drop More
4	**password**	varchar(255)	*utf8mb4_general_ci*		No	*None*			Change Drop More
5	**status**	tinyint(1)			No	*None*			Change Drop More

D. Struktur file pembuatan “Bank INI”

Dalam pembuatan projek ini kita membuatnya manjadi modular



3. Implementasi materi

E. Materi ke -1 OOP

```
class User:
    def __init__(self):
        self.id = None
        self.nama = None
        self._pin = None
        self.__password = None

    def generateId(self):
        listId = []
        listchoice = [0, 1, 2, 3, 4, 5, 6,7 ,8 ,9]
        for i in range(5):
            choice = random.choice(listchoice)
            listId.append(str(choice))
        return int(''.join(listId))

    def inputData(self):
        self.id = self.generateId()
        self.nama = str(input('Masukkan Nama Anda : '))
        self.pin = int(input('Masukkan Pin yang anda inginkan
: '))
        self.password = str(input('Masukkan Password Anda :
'))

        return self.id, self.nama, self.pin, self.password
```

F. Materi ke-2 class, atribut, method

Class & Atribut

```
class Query():
    def __init__(self):
        self.host = None
        self.user = None
        self.password = None
        self.database = None
        self.connect = None
        self.cursor = None
```

Method

```
def login(self, id, password):
        self.cursor.execute(f'SELECT * FROM user WHERE
id="{id}" and password="{password}"')
        return self.cursor.fetchone()
```

G. Materi ke-3 Access Modifier

Public , Privaet, & Protected

```
class User:
    def __init__(self):
        self.id = None
        self.nama = None
        self._pin = None
        self.__password = None
```

Setter Getter

```
def passChanger(self, new):
      self.changePassword = new
      database.changePassword(new, self.id)

  @property
  def changePassword(self):
      return self.__password
  @changePassword.setter
  def changePassword(self, newPass):
     self.__password = newPass
```

H. Materi ke-4 Inheritance

```
class Customer(User):
    def __init__(self):
        super().__init__()
        self.saldo = None
        self.saving = None
```

I. Materi ke-5 Overloading & Overriding

Overriding

- pada fungsi admin

```
def checkCustomer(self, data):
        print('\n====================================='
)
        print(f'ID Customer         : {data[0]}')
        print(f'Nama Customer       : {data[1]}')
        print(f'Saldo Customer      : {data[5]}')

        print(f'\nPin Customer       : {data[2]}')
        print(f'Password Customer   : {data[3]}')
        print('=====================================\n'
)
```

- pada fungsi teller

```
def checkCustomer(self, data):
```

```
    print('\n====================================')
            print(f'ID Customer     : {data[0]}')
            print(f'Nama Customer   : {data[1]}')
            print(f'Saldo Customer  : {data[5]}')

    print('====================================\n')
```

4. Dokumentasi Source Code

Dokumentasi penuh => https://github.com/JatiKurniawan/BankINI

Query.py

```
import mysql.connector

class Query():
    def __init__(self):
        self.host = None
        self.user = None
        self.password = None
        self.database = None
        self.connect = None
        self.cursor = None

    def connection(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connect = mysql.connector.connect(host = host,
user = user, passwd = password, database = database)
        self.cursor = self.connect.cursor()
        return True

    def login(self, id, password):
        self.cursor.execute(f'SELECT * FROM user WHERE
id="{id}" and password="{password}"')
        return self.cursor.fetchone()

    def tambahAkun(self, data):
        self.cursor.execute(f'INSERT INTO list_registrasi (id,
nama, pin, password, status) VALUES ("{data[0]}", "{data[1]}",
"{data[2]}", "{data[3]}", "False")')
        self.connect.commit()

    def register(self, data):
```

```
        self.cursor.execute(f'SELECT * FROM user WHERE
id="{data[0]}"')
        check = self.cursor.fetchone()
        if check:
            return 'Id Sudah Terdaftar, Silahkan Masuk ke Akun
anda'
        self.cursor.execute(f'INSERT INTO user (id, nama, pin,
password, saldo, saving, id_tipe) VALUES ("{data[0]}",
"{data[1]}", "{data[2]}", "{data[3]}", "0", "0", "34521")')
        self.connect.commit()
        return True

    def informasiPenarikan(self):
        self.cursor.execute(f'SELECT * FROM list_withdraw WHERE
status=0')
        return self.cursor.fetchall()

    def konfirmasiPenarikan(self, id, jumlah_penarikan):
        self.cursor.execute(f"DELETE FROM list_withdraw WHERE
id='{id}' ")
        self.connect.commit()
        self.cursor.execute(f"UPDATE user SET saldo = saldo -
%s WHERE id=%s", (jumlah_penarikan, id))
        self.connect.commit()
        return True

    def dataAkun(self):
        self.cursor.execute('SELECT * FROM user WHERE
id_tipe=34521')
        return self.cursor.fetchall()

    def cekSaldo(self, id):
        self.cursor.execute(f"SELECT saldo FROM user WHERE
id='{id}'")
        return self.cursor.fetchone()

    def cariAkunCustomer(self, id):
        self.cursor.execute(f"SELECT * FROM user WHERE
id='{id}' ")
        return self.cursor.fetchone()

    def buatakunTeller(self, data):
        self.cursor.execute(f"INSERT INTO user (id, nama, pin,
password, saldo, saving, id_tipe ) VALUES (%s, %s, %s, %s, 0,
0, 11150)", (data[0], data[1], data[2], data[3]))
        self.connect.commit()

    def deposito(self, id, jumlah):
```

```
        self.cursor.execute(f'UPDATE user SET saldo = saldo +
%s WHERE id = %s', (jumlah, id))
        self.connect.commit()

    def withdraw(self, id, saldo, data, pin):
        if data[1] == pin:
            self.cursor.execute(f'INSERT INTO list_withdraw
(id, saldo, amount, status ) VALUES(%s, %s, %s, %s)', (id,
saldo, data[0], False))
            self.connect.commit()
            print('Penarikan saldo membutuhkan konfirmasi
teller, Harap menunggu Konfirmasi')
        else:
            print('Pin yang anda masukkan Salah')

    def informasiRegistrasi(self):
        self.cursor.execute('SELECT * FROM list_registrasi
WHERE status=0')
        return self.cursor.fetchall()

    def konfirmasiRegistrasi(self, data):
        self.cursor.execute(f'DELETE FROM list_registrasi WHERE
id="{data[0]}"')
        self.connect.commit()
        self.cursor.execute(f'INSERT INTO user (id, nama, pin,
password, saldo, saving, id_tipe) VALUES (%s, %s, %s, %s, 0, 0,
34521)', (data[0], data[1], data[2], data[3]))
        self.connect.commit()
```

User.py

```
import random
from query import Query

database = Query()
database.connection('localhost', 'root', '', 'bank_ini')


class User:
    def __init__(self):
        self.id = None
        self.nama = None
        self._pin = None
        self.__password = None

    def generateId(self):
        listId = []
        listchoice = [0, 1, 2, 3, 4, 5, 6,7 ,8 ,9]
        for i in range(5):
```

```
            choice = random.choice(listchoice)
            listId.append(str(choice))
        return int(''.join(listId))

    def inputData(self):
        self.id = self.generateId()
        self.nama = str(input('Masukkan Nama Anda : '))
        self._pin = int(input('Masukkan Pin yang anda inginkan
: '))
        self.__password = str(input('Masukkan Password Anda :
'))

        return self.id, self.nama, self._pin, self.__password

    def printInfo(self, data):
        print('\nInformasi Akun')
        print('====================================')
        print(f'Id : {data[0]}')
        print(f'Nama : {data[1]}')
        print(f'Password : {data[3]}')
        print('====================================\n')

    def login(self):
        id = int(input('Masukkan ID Anda : '))
        password = str(input('Masukkan Password Anda : '))

        check = database.login(id, password)
        if check:
            self.id = check[0]
            self.nama = check[1]
            self._pin = check[2]
            self.__password = check[3]
            return check
        return False

    def passChanger(self, new):
        self.changePassword = new
        database.changePassword(new, self.id)

    @property
    def changePassword(self):
        return self.__password
    @changePassword.setter
    def changePassword(self, newPass):
        self.__password = newPass

    def logout(self):
        return False
```

Customer.py

```
from user import User

class Customer(User):
    def __init__(self):
        super().__init__()
        self.saldo = None
        self.saving = None

    def inputData(self, data):
        self.saldo = data[4]
        self.saving = data[5]

    def checkSaldo(self, data):
        print('\n====================================')
        print(f'Nama Customer : {data[1]} ({data[0]})')
        print(f'Saldo Customer : {data[2][0]}')
        print('====================================\n')

    def checkAkun(self, data):
        print('\n====================================')
        print(f'Nama Customer : {data[1]} ({data[0]})')
        print(f'Saldo Customer : {data[4]}')

        print(f'\nPin = {data[2]}')
        print(f'Password = {data[3]}')
        print('====================================\n')

    def deposito(self):
        print(f'Saldo Anda Sekarang : Rp {self.saldo}')
        deposito = int(input('Jumlah Deposito : Rp '))
        self.saldo = self.saldo + deposito
        return deposito

    def savingMoney(self):
        print('saving')

    def withdraw(self):
        print(f'Saldo Anda Sekarang : Rp {self.saldo}')
        withdraw = int(input('Jumlah Penarikan : Rp '))
        pin = int(input('Masukkan Pin anda : '))
        return withdraw, pin

    def register(self):
        print('register')
```

Admin.py

```
from user import User
from query import Query

database = Query()
database.connection('localhost', 'root', '', 'bank_ini')

class Admin(User):
    def __init__(self):
        super().__init__()

    def printIdCustomer(self):
        akun = database.dataAkun()
        for i, j in enumerate(akun, start=1):
            print(f"{i}. {j[1]} ({j[0]})")
        id = int(input('Pilih ID Customer : '))
        return id

    def checkCustomer(self, data):
        print('\n====================================')
        print(f'ID Customer         : {data[0]}')
        print(f'Nama Customer       : {data[1]}')
        print(f'Saldo Customer      : {data[5]}')

        print(f'\nPin Customer      : {data[2]}')
        print(f'Password Customer   : {data[3]}')
        print('====================================\n')

    def akunTeller(self):
        id = self.generateId()
        nama = str(input('Masukkan Nama : '))
        pin = int(input('Masukkan 6 digit Pin : '))
        password = str(input('Masukkan Password : '))

        return id, nama, pin, password

    def konfirmasiRegistrasi(self, data):
        print('====================================\n')
        for i in data:
            print(f'Customer : {i[1]} ({i[0]})')
            confirm = str(input('Konfirmasi ? '))
            if confirm.lower() == 'yes':
                database.konfirmasiRegistrasi([i[0], i[1],
i[2], i[3]])
```

Teller.py

```
from user import User
from query import Query

database = Query()
```

```
database.connection('localhost', 'root', '', 'bank_ini')

class Teller(User):
    def __init__(self):
        super().__init__()

    def printIdCustomer(self):
        akun = database.dataAkun()
        for i, j in enumerate(akun, start=1):
            print(f"{i}. {j[1]} ({j[0]})")
        id = int(input('Pilih ID Customer : '))
        return id

    def checkCustomer(self, data):
        print('\n====================================')
        print(f'ID Customer     : {data[0]}')
        print(f'Nama Customer   : {data[1]}')
        print(f'Saldo Customer  : {data[5]}')
        print('====================================\n')

    def konfirmasiPenarikan(self):
        penarikan = database.informasiPenarikan()
        for i in penarikan:
            print('====================================')
            print(f'ID Customer         : {i[0]}')
            print(f'Withdraw Amount     : {i[2]}')
            print(f'Saldo Customer      : {i[1]}')
            confirm = str(input('Konfirmasi ? : '))
            if confirm.lower() == 'yes':
                database.konfirmasiPenarikan(i[0], i[2])
```

Menu.py

```
import os
from pyfiglet import Figlet

from query import Query

from user import User
from customer import Customer
from teller import Teller
from admin import Admin

database = Query()
database.connection('localhost', 'root', '', 'bank_ini')

dataUser = User()
```

```
dataCustomer = Customer()
dataTeller = Teller()
dataAdmin = Admin()


listMenuCustomer = ['Setor Tunai', 'Penarikan Tunai',
'Informasi Saldo', 'Informasi Akun']
listMenuTeller = ['Konfirmasi Penarikan', 'Cari Akun Customer']
listMenuAdmin = ['Cari Akun Customer', 'Buat Akun Teller',
'Konfirmasi Registrasi']


def menu(list):
    for i in range(len(list)):
        print(f'{i+1}. {list[i]}')
    print('0. Keluar')

    choice = int(input('Masukkan pilihan anda : '))
    return choice

def mainMenu():
    while True:
        # os.system(('cls'))
        f = Figlet(font='slant')
        print(f.renderText('Bank INI'))
        login = dataUser.login()
        if login :
            if login[6] == 34521:
                dataCustomer.inputData(login)
                while True:
                    choiceCustomer = menu(listMenuCustomer)
                    if choiceCustomer == 1:
                        deposito = dataCustomer.deposito()
                        database.deposito(login[0], deposito)
                    elif choiceCustomer == 2:
                        withdraw = dataCustomer.withdraw()
                        database.withdraw(login[0],
dataCustomer.saldo, withdraw, login[2])
                    elif choiceCustomer == 3:
                        saldo = database.cekSaldo(login[0])
                        dataCustomer.checkSaldo([login[0],
login[1], saldo])
                    elif choiceCustomer == 4:
                        data =
database.cariAkunCustomer(login[0])
                        dataCustomer.checkAkun(data)
                    elif choiceCustomer == 0:
                        return False
```

```
            elif login[6] == 98710:
                while True:
                    choiceTeller = menu(listMenuTeller)
                    if choiceTeller == 1:
                        dataTeller.konfirmasiPenarikan()
                    elif choiceTeller == 2:
                        id = dataTeller.printIdCustomer()
                        data = database.cariAkunCustomer(id)
                        dataTeller.checkCustomer(data)
                    elif choiceTeller == 0:
                        return False

            elif login[6] == 11150:
                while True:
                    choiceAdmin = menu(listMenuAdmin)
                    if choiceAdmin == 1:
                        id = dataAdmin.printIdCustomer()
                        data = database.cariAkunCustomer(id)
                        dataAdmin.checkCustomer(data)
                    elif choiceAdmin == 2:
                        data = dataAdmin.akunTeller()
                        database.buatakunTeller(data)
                    elif choiceAdmin == 3:
                        data = database.informasiRegistrasi()
                        dataAdmin.konfirmasiRegistrasi(data)

        else:
            print('\nPassword atau Username Salah')
            signup = str(input('Buat Akun? <y/n>'))
            if signup.lower() == 'y':
                dataSign = dataUser.inputData()
                database.tambahAkun(dataSign)
                dataUser.printInfo(dataSign)
                print('\nPembuatan akun membutuhkan konfirmasi
Admin Bank. Harap Menunggu Konfirmasi\n')

mainMenu()
```