**LAPORAN CHALLENGE**

**PEWARISAN DAN HAK AKSES**

**PEMROGRAMAN BERBASIS OBJEK**

**KELAS VIII**

T. A. Semester Ganjil 2024/2025



**5230411327**

**Muhammad Irfan Baihaqi**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS & TEKNOLOGI**

**UNIVERSITAS TEKNOLOGI YOGYAKARTA**

**YOGYAKARTA**

**2024**

**Kata Pengantar**

Dengan rasa hormat dan penuh rasa syukur yang mendalam, penulis mengucapkan puji dan syukur kepada Tuhan Yang Maha Esa. Berkat limpahan rahmat dan karunia-Nya, penulis telah berhasil mengatasi berbagai rintangan dan kesulitan yang dihadapi selama proses penulisan laporan ini.

Penulis menyadari sepenuhnya bahwa laporan ini masih jauh dari kata sempurna dan memiliki banyak kekurangan. Oleh karena itu, penulis dengan rendah hati menerima segala bentuk kritik dan saran yang bersifat membangun untuk perbaikan dan penyempurnaan laporan ini di masa mendatang. Penulis berharap bahwa laporan ini dapat memberikan manfaat dan kontribusi yang positif bagi semua pihak yang berkepentingan.

Terima kasih penulis haturkan atas perhatian dan kerjasamanya bagi semua pihak yang telah mendukung keberhasilan penulis dalam menyelesaikan tugas ini. Semoga Tuhan Yang Maha Esa senantiasa memberikan petunjuk dan perlindungan-Nya kepada kita semua. Amin.

Yogyakarta, 12 Oktober 2024

Muhammad Irfan Baihaqi

**BAB 1**

**PENDAHULUAN**

**A. Tujuan Praktikum**

Tujuan dari praktikum atau *challange* ini adalah untuk mempelajari sekaligus mempraktekkan salah satu materi Pemrograman Berbasis Objek menggunakan Bahasa Python, yaitu Pewarisan sifat atau *inheritance*, penggunaan *decorator* (*property*, *getter* dan *setter*) *privat* variabel, *protected* *variable* dan *public* *variable*.

**B. Dasar Teori**

Pewarisan Sifat atau *Inheritence* adalah konsep dasar dalam Pemrograman Berorientasi Objek (OOP) di mana sebuah kelas baru dibuat dari kelas yang sudah ada. Kelas baru ini mewarisi atribut dan metode dari kelas yang sudah ada. Ini memungkinkan penggunaan kembali kode dan dapat menyederhanakan sistem yang kompleks dengan memodelkan hierarki dunia nyata. Pada pyhton agar *child class* mewakili atribut dari parentnya , bisa menggunakan super()\_\_init\_\_(parameter) di dalam method init *child class.*

Decorator dalam Python adalah alat yang kuat yang memungkinkan modifikasi fungsi atau metode. Decorator @property digunakan untuk mendefinisikan metode dalam kelas yang berfungsi seperti atribut. Decorator @property ini digunakan bersama dengan @propertyname.setter dan @propertyname.deleter untuk mendefinisikan metode *getter, setter,* dan *deleter*.

Python menggunakan konvensi penamaan untuk menentukan aksesibilitas variable. Variabel Publik dapat diakses dari mana saja. Dinamai secara konvensional. Variabel Protected, ditandai dengan satu garis bawah \_ di depan namanya. Dimaksudkan untuk diakses hanya dalam kelas dan subkelas mereka sendiri. Variabel Privat ditandai dengan dua garis bawah \_\_ di depan namanya. Dimaksudkan untuk diakses hanya dalam kelas mereka sendiri.

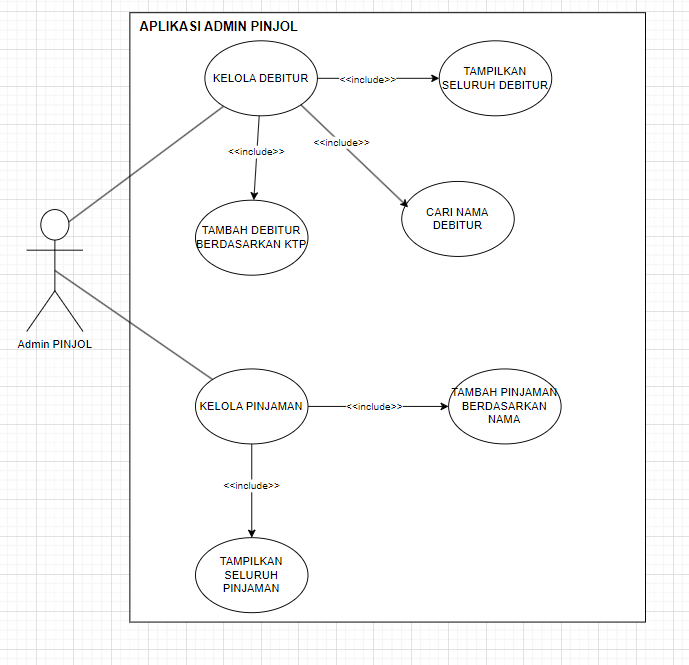
**BAB II**

**UML DAN KODE PROGRAM**

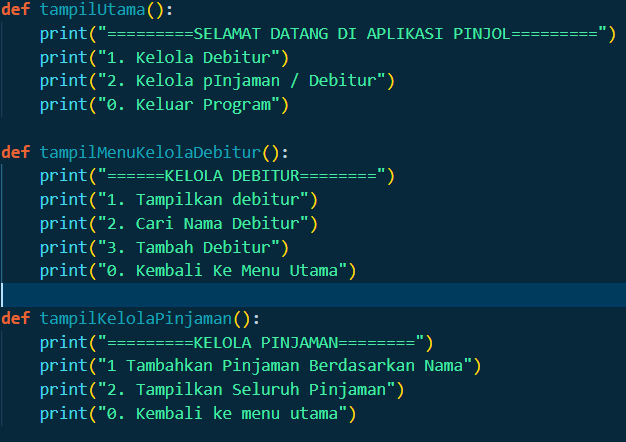
1. **UML**

UML (Unified Modeling Language) adalah alat bantu untuk memvisualisasikan, mendesain, dan mendokumentasikan sistem perangkat lunak. UML menggunakan berbagai jenis diagram untuk menggambarkan struktur, perilaku, dan interaksi dalam sebuah sistem. Dengan UML, pengembang dan tim dapat memahami dan berkomunikasi tentang arsitektur sistem dengan lebih efektif, serta mempermudah proses perancangan dan pemeliharaan sistem yang kompleks. Visualisasinya memberikan cara yang jelas dan terstruktur untuk menggambarkan alur kerja dan hubungan antar komponen.

USE CASE DIAGRAM:



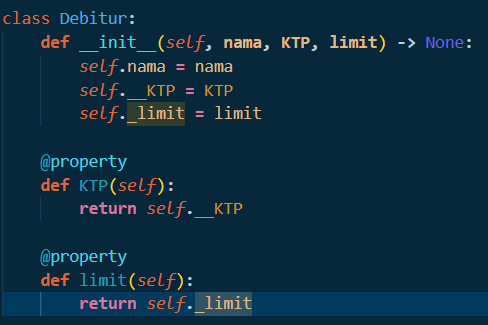
1. **KODE PROGRAM**

****

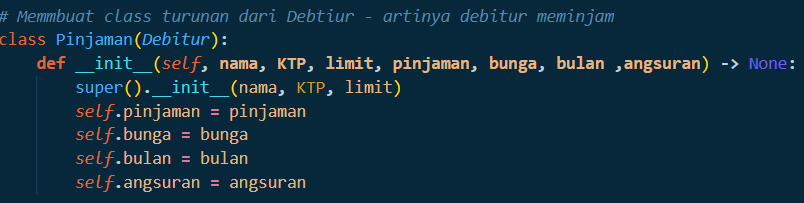
Kode di atas digunakan untuk menampilkan menu secara berturut-turut, yaitu Menu Utama, Menu Kelola debitur, dan kelola pinjaman.



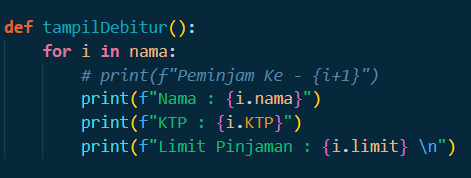
Membuat list kosong dimana list nama digunakan untuk menyimpan objek-objek Debitur dan list pinjaman untuk menyimpan objek pinjaman.



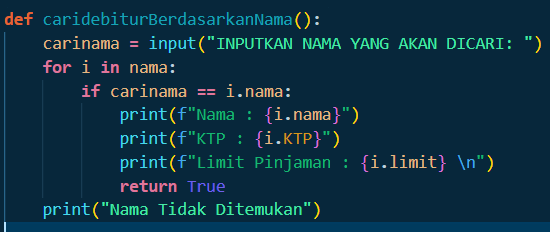
Kode di atas mendefinisikan sebuah kelas dalam Python yang bernama Debitur, yang memiliki tiga atribut: nama, \_\_KTP, dan \_limit. Ketika sebuah objek dari kelas Debitur dibuat, metode inisialisasi \_\_init\_\_ akan dipanggil dan akan menginisialisasi atribut-atribut tersebut dengan nilai-nilai yang diberikan sebagai parameter. Atribut nama adalah atribut publik, sedangkan \_\_KTP adalah atribut privat, yang ditandai dengan dua garis bawah sebelum nama variabel, yang berarti hanya dapat diakses dari dalam kelas tersebut. Atribut \_limit adalah atribut protected, yang ditandai dengan satu garis bawah sebelum nama variabel, yang biasanya hanya diakses dari dalam kelas dan subkelasnya. Penggunaan dekorator @property pada KTP dan limit memungkinkan kedua atribut tersebut diakses seperti atribut biasa, tetapi dengan kontrol tambahan yang ditawarkan oleh properti, seperti pengambilan nilai dengan getter (tetapi tidak ada setter di sini, sehingga properti ini hanya baca). Dengan ini, pengguna kelas dapat mengambil nilai KTP dan limit tanpa langsung mengakses atribut privat dan protected tersebut, menjaga enkapsulasi dan keamanan data. Jadi, kelas ini mengimplementasikan prinsip OOP dengan baik melalui penggunaan inisialisasi atribut, aksesibilitas variabel, dan dekorator properti.



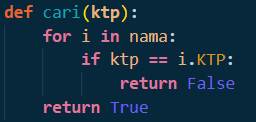
Kode tersebut mendefinisikan sebuah kelas bernama Pinjaman yang merupakan turunan dari kelas Debitur. Dalam konteks ini, kelas Pinjaman adalah sebuah subclass, yang berarti ia mewarisi semua atribut dan metode dari kelas Debitur. Metode \_\_init\_\_ atau konstruktor dari kelas Pinjaman digunakan untuk menginisialisasi objek-objek dari kelas ini. Ketika objek Pinjaman dibuat, metode \_\_init\_\_ memanggil konstruktor dari kelas induk (Debitur) menggunakan super().\_\_init\_\_(nama, KTP, limit) untuk menginisialisasi atribut nama, KTP, dan limit. Setelah itu, atribut spesifik untuk kelas Pinjaman diinisialisasi, yaitu pinjaman, bunga, bulan, dan angsuran. Ini berarti setiap objek Pinjaman akan memiliki atribut-atribut dari kelas Debitur serta atribut tambahan yang terkait dengan pinjaman seperti jumlah pinjaman, bunga, durasi dalam bulan, dan angsuran bulanan.



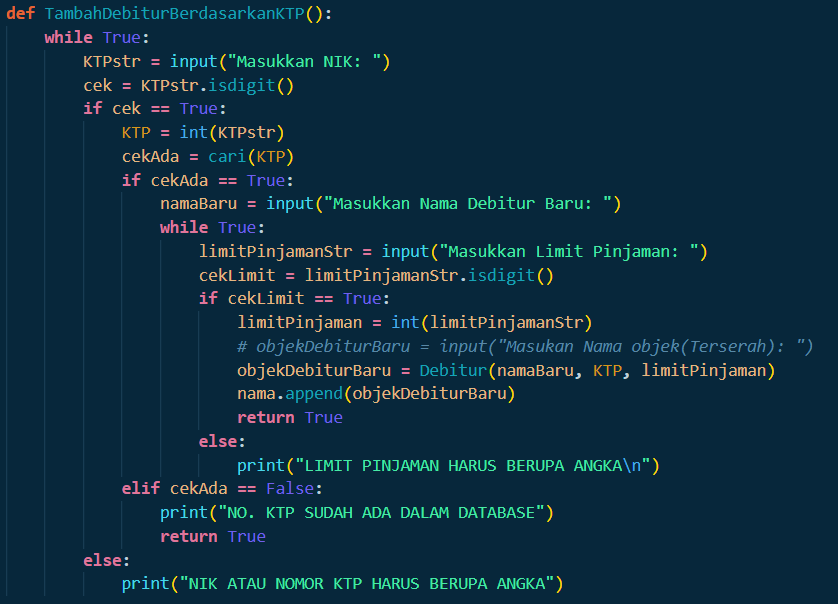
Fungsi tampilDebitur() bertujuan untuk menampilkan informasi mengenai semua debitur yang tersimpan dalam list nama. Fungsi ini akan melakukan iterasi melalui setiap objek dalam list nama, yang diharapkan berisi objek-objek dari kelas Debitur. Untuk setiap objek debitur yang ditemui, fungsi ini akan mencetak nama, KTP, dan limit pinjaman debitur tersebut ke layar. Format cetakannya dirancang agar mudah dibaca, dengan setiap informasi dipisahkan oleh baris baru.



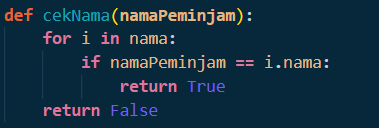
Fungsi caridebiturBerdasarkanNama() digunakan untuk mencari debitur berdasarkan nama yang diinput oleh pengguna. Fungsi ini meminta input nama melalui input("INPUTKAN NAMA YANG AKAN DICARI: ") dan menyimpannya dalam variabel carinama. Kemudian, akan melakukan iterasi pada setiap objek dalam list nama, yang berisi objek-objek dari kelas Debitur. Jika menemukan objek dengan atribut nama yang sesuai dengan carinama, fungsi tersebut akan mencetak informasi nama, KTP, dan limit pinjaman debitur tersebut. Setelah mencetak informasi, fungsi mengembalikan nilai True dan selesai eksekusi. Jika iterasi selesai tanpa menemukan nama yang sesuai, fungsi akan mencetak "Nama Tidak Ditemukan".



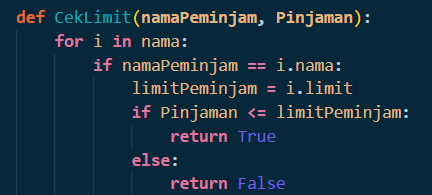
Fungsi cari(ktp) digunakan untuk memeriksa apakah KTP yang diberikan sudah ada dalam list nama. List nama ini berisi objek-objek dari kelas Debitur. Fungsi melakukan iterasi melalui setiap objek dalam daftar tersebut dan memeriksa apakah nilai KTP dari objek tersebut sama dengan nilai KTP yang diinputkan sebagai parameter ktp. Jika ditemukan kecocokan, fungsi akan mengembalikan False, menunjukkan bahwa KTP tersebut sudah ada dalam database. Jika iterasi selesai tanpa menemukan KTP yang cocok, fungsi akan mengembalikan True, menandakan bahwa KTP tersebut belum ada dalam database dan dapat digunakan untuk menambahkan debitur baru. Fungsi ini berguna untuk memastikan bahwa tidak ada duplikasi KTP dalam sistem, sehingga setiap debitur memiliki identitas unik



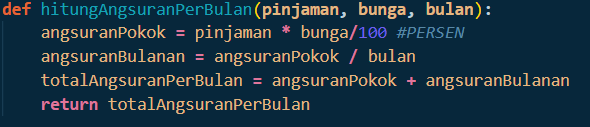
Fungsi TambahDebiturBerdasarkanKTP() dirancang untuk menambahkan debitur baru berdasarkan nomor KTP. Dalam loop while True, fungsi ini pertama kali meminta input nomor KTP (NIK) dari pengguna dan memastikan input tersebut hanya terdiri dari digit dengan menggunakan metode isdigit(). Jika input valid, nomor KTP tersebut dikonversi menjadi integer. Fungsi kemudian memanggil fungsi cari(KTP) untuk memeriksa apakah nomor KTP sudah ada dalam daftar debitur. Jika KTP belum ada (mengembalikan True), fungsi meminta pengguna untuk memasukkan nama debitur baru dan limit pinjaman. Input limit pinjaman juga diverifikasi agar hanya berupa angka. Setelah input valid, objek debitur baru dibuat menggunakan kelas Debitur dan ditambahkan ke dalam list nama. Jika semua langkah selesai dengan sukses, fungsi mengembalikan True dan keluar dari loop. Namun, jika KTP sudah ada dalam database (False), atau input NIK tidak valid, fungsi akan memberikan pesan kesalahan yang sesuai dan mengulangi proses input.



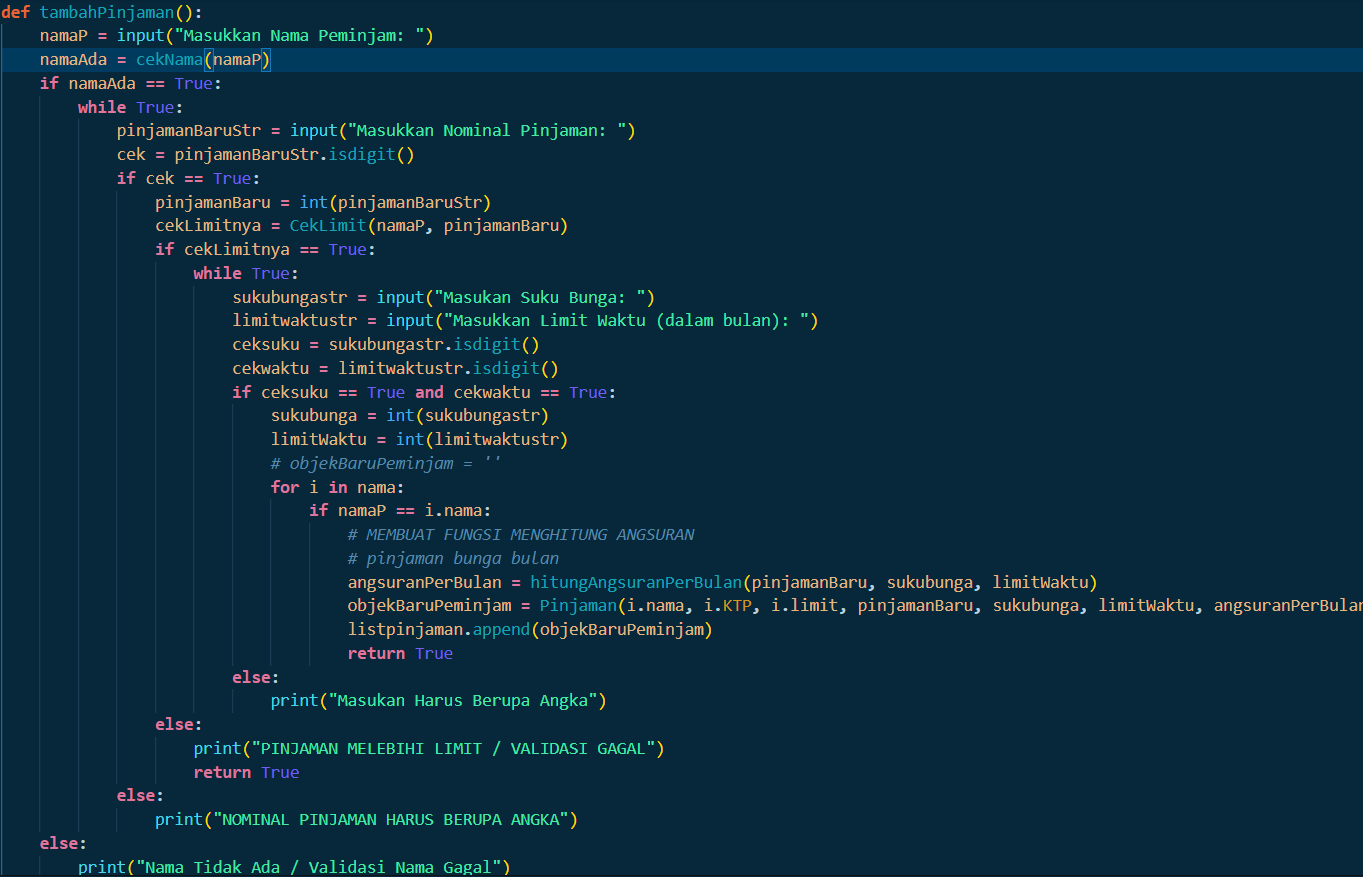
Fungsi cekNama(namaPeminjam) berfungsi untuk memeriksa apakah nama peminjam tertentu sudah ada dalam list nama. List nama ini berisi objek-objek dari kelas Debitur. Fungsi ini melakukan iterasi melalui setiap objek dalam list tersebut, dan pada setiap iterasi, ia memeriksa apakah atribut nama dari objek tersebut sama dengan nilai namaPeminjam yang diberikan sebagai parameter. Jika ditemukan kesamaan, fungsi akan mengembalikan nilai True, menunjukkan bahwa nama tersebut sudah ada dalam daftar. Jika fungsi selesai melakukan iterasi tanpa menemukan kesamaan, fungsi akan mengembalikan nilai False, yang menandakan nama tersebut belum ada dalam daftar dan bisa digunakan untuk menambahkan debitur baru.



Fungsi CekLimit(namaPeminjam, Pinjaman) berfungsi untuk memeriksa apakah jumlah pinjaman yang diminta oleh peminjam melebihi batas (limit) yang diperbolehkan. Fungsi ini menerima dua parameter, yaitu namaPeminjam (nama debitur yang ingin meminjam) uang, dan Pinjaman, yaitu jumlah uang yang ingin dipinjam. Fungsi ini akan melakukan iterasi melalui list nama, yang berisi objek-objek dari kelas Debitur. Pada setiap iterasi, fungsi akan memeriksa apakah nama peminjam cocok dengan atribut nama dari objek debitur dalam list. Jika ditemukan kecocokan, fungsi akan mengambil nilai limit pinjaman dari objek tersebut (limitPeminjam = i.limit). Selanjutnya, fungsi memeriksa apakah jumlah pinjaman yang diminta (Pinjaman) tidak melebihi limit pinjaman (limitPeminjam). Jika jumlah pinjaman kurang dari atau sama dengan limit, fungsi akan mengembalikan nilai True, menandakan bahwa pinjaman dapat disetujui. Jika jumlah pinjaman melebihi limit, fungsi akan mengembalikan nilai False, menandakan bahwa pinjaman tidak dapat disetujui.

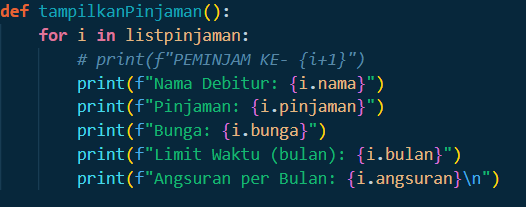


Fungsi ini digunakan untuk menghitung total angsuran per bulan, dengan parameter pinjaman (yaitu jumlah pinjaman debitur, bunga, dan limitasi waktu bayar dalam bulan). Angsuran pokok didapat dari perhitungan pinjaman dikali bunga per 100 (alasan mengapa per 100 adalah karena parameter bunga masih dalam angka basis 10 bukan persen). Kemudian angsuran bulanan didapat dari angsuran pokok dibagi bulan. Lalu total angsuran yang dihasilkan per bulan didapat dari angsuran bulan ditambah dengan angsuran pokok. Fungsi akan mereturn total angsuran per bulan.

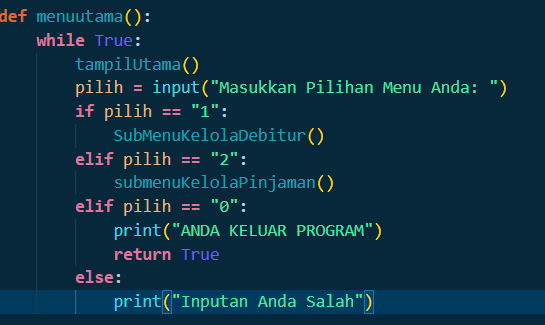




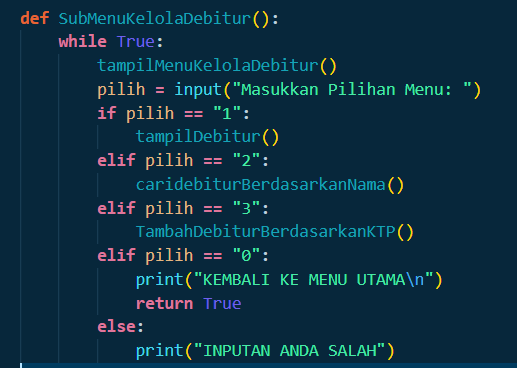
Fungsi tambahPinjaman() bertujuan untuk menambah pinjaman baru untuk debitur yang sudah ada. Pertama, nama peminjam diminta dari pengguna melalui input("Masukkan Nama Peminjam: ") dan kemudian dicek apakah nama tersebut ada di daftar debitur menggunakan fungsi cekNama(namaP). Jika nama ditemukan, fungsi memasuki loop untuk meminta input nominal pinjaman dan memeriksa apakah input tersebut berupa angka. Setelah itu, fungsi CekLimit(namaP, pinjamanBaru) dipanggil untuk memastikan jumlah pinjaman tidak melebihi limit yang diperbolehkan. Jika jumlah pinjaman valid, loop lain dimulai untuk meminta input suku bunga dan jangka waktu dalam bulan, memastikan keduanya berupa angka. Selanjutnya, untuk setiap debitur dalam daftar nama, jika nama cocok, angsuran per bulan dihitung dengan memanggil fungsi hitungAngsuranPerBulan(pinjamanBaru, sukubunga, limitWaktu). Sebuah objek Pinjaman baru kemudian dibuat dan ditambahkan ke daftar listpinjaman. Jika inputan tidak valid atau melebihi limit, pesan kesalahan akan dicetak dan loop akan mengulangi proses input sampai nilai yang valid diberikan.



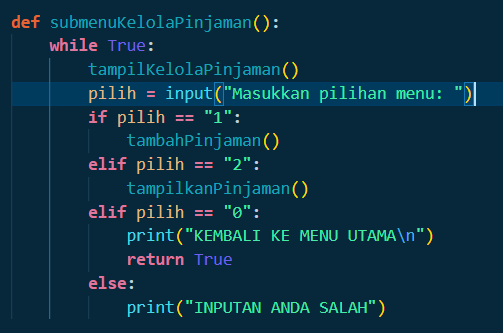
Fungsi tampilkanPinjaman() dirancang untuk menampilkan informasi mengenai semua pinjaman yang ada dalam daftar listpinjaman. Fungsi ini melakukan iterasi melalui setiap objek dalam daftar listpinjaman, yang diharapkan berisi objek-objek dari kelas Pinjaman. Pada setiap iterasi, fungsi mencetak informasi nama debitur, jumlah pinjaman, persentase bunga, durasi pinjaman dalam bulan, dan jumlah angsuran per bulan. Informasi ini diformat agar mudah dibaca, dengan setiap bagian informasi dipisahkan oleh baris baru. Dengan menggunakan fungsi ini, pengguna dapat dengan cepat melihat rincian semua pinjaman yang telah dibuat dan disimpan dalam sistem



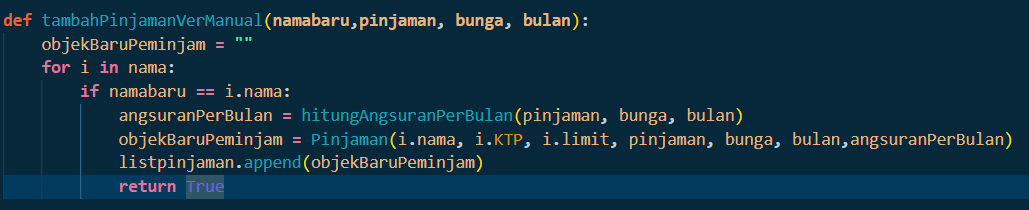
Fungsi menuutama() dirancang untuk mengelola menu utama dari aplikasi pinjaman ini. Fungsi ini menggunakan loop while True untuk terus menampilkan menu utama dan menerima input dari pengguna hingga pengguna memilih untuk keluar. Fungsi tampilUtama() pertama kali dipanggil untuk menampilkan opsi-opsi di menu utama, seperti mengelola debitur, mengelola pinjaman, atau keluar dari program. Pengguna diminta untuk memasukkan pilihan mereka melalui input("Masukkan Pilihan Menu Anda: "). Berdasarkan input ini, fungsi menggunakan kondisi if-elif-else untuk menentukan tindakan berikutnya. Jika pengguna memilih "1", fungsi SubMenuKelolaDebitur() akan dipanggil untuk menampilkan sub-menu terkait pengelolaan debitur. Jika pengguna memilih "2", fungsi submenuKelolaPinjaman() akan dipanggil untuk mengelola pinjaman. Jika pengguna memilih "0", fungsi akan mencetak pesan "ANDA KELUAR PROGRAM" dan mengembalikan nilai True, yang berarti keluar dari loop dan mengakhiri program. Jika input pengguna tidak sesuai dengan pilihan yang diberikan, fungsi akan mencetak "Inputan Anda Salah" dan mengulangi loop untuk meminta input lagi.



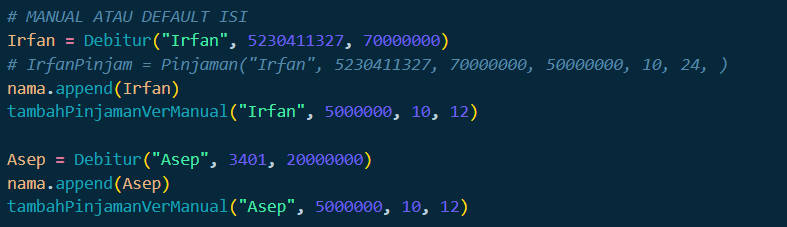
Fungsi SubMenuKelolaDebitur() berfungsi sebagai antarmuka pengguna untuk mengelola debitur. Dalam loop while True, fungsi ini pertama-tama memanggil tampilMenuKelolaDebitur() untuk menampilkan opsi-opsi menu yang terkait dengan pengelolaan debitur. Pengguna kemudian diminta untuk memasukkan pilihan menu melalui input("Masukkan Pilihan Menu: "). Berdasarkan input ini, fungsi menggunakan struktur kontrol if-elif-else untuk mengeksekusi tindakan yang sesuai: jika pengguna memilih "1", fungsi tampilDebitur() dipanggil untuk menampilkan daftar debitur; jika memilih "2", fungsi caridebiturBerdasarkanNama() dipanggil untuk mencari debitur berdasarkan nama; jika memilih "3", fungsi TambahDebiturBerdasarkanKTP() dipanggil untuk menambahkan debitur baru berdasarkan KTP; dan jika memilih "0", fungsi akan mencetak "KEMBALI KE MENU UTAMA" dan mengembalikan nilai True untuk keluar dari loop dan kembali ke menu utama. Jika input pengguna tidak valid, pesan "INPUTAN ANDA SALAH" akan ditampilkan dan loop akan mengulangi proses input.



Fungsi submenuKelolaPinjaman() bertujuan untuk mengelola menu yang terkait dengan pinjaman. Dalam loop while True, fungsi ini pertama-tama memanggil tampilKelolaPinjaman() untuk menampilkan opsi-opsi yang tersedia untuk pengelolaan pinjaman. Pengguna kemudian diminta untuk memasukkan pilihan mereka melalui input("Masukkan pilihan menu: "). Berdasarkan input ini, fungsi menggunakan struktur kontrol if-elif-else untuk mengeksekusi tindakan yang sesuai: jika pengguna memilih "1", fungsi tambahPinjaman() dipanggil untuk menambahkan pinjaman baru; jika memilih "2", fungsi tampilkanPinjaman() dipanggil untuk menampilkan semua pinjaman yang ada. Jika pengguna memilih "0", fungsi akan mencetak "KEMBALI KE MENU UTAMA" dan mengembalikan nilai True, yang berarti keluar dari loop dan kembali ke menu utama. Jika input pengguna tidak valid, pesan "INPUTAN ANDA SALAH" akan ditampilkan dan loop akan mengulangi proses input.



Fungsi tambahPinjamanVerManual(namabaru, pinjaman, bunga, bulan) bertujuan untuk menambahkan pinjaman baru secara manual untuk debitur yang sudah ada. Fungsi ini menerima empat parameter: namabaru (nama debitur), pinjaman (jumlah pinjaman yang ingin diajukan), bunga (persentase bunga pinjaman), dan bulan (jangka waktu pinjaman dalam bulan). Dalam fungsi ini, dilakukan iterasi melalui daftar nama, yang berisi objek-objek dari kelas Debitur. Jika nama yang diberikan (namabaru) cocok dengan nama salah satu objek debitur dalam daftar tersebut, angsuran per bulan dihitung menggunakan fungsi hitungAngsuranPerBulan(pinjaman, bunga, bulan). Selanjutnya, dibuat objek Pinjaman baru dengan informasi debitur tersebut, termasuk pinjaman, bunga, bulan, dan angsuran per bulan. Objek Pinjaman baru ini kemudian ditambahkan ke dalam daftar listpinjaman, dan fungsi mengembalikan nilai True menandakan bahwa pinjaman berhasil ditambahkan. Fungsi ini memudahkan proses penambahan pinjaman secara manual



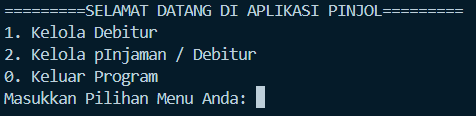
Kelompok baris kode di atas adalah untuk menambahkan objek-objek default yang telah ada dari awal program dibuat.



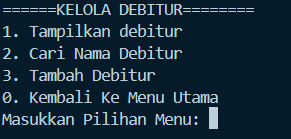
Kode tersebut untuk memanggil menu utama, yang artinya menjalankan seluruh program.

**HASIL RUNNING:**

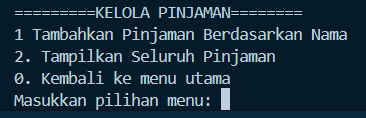
1. **TAMPIL MENU UTAMA**

****

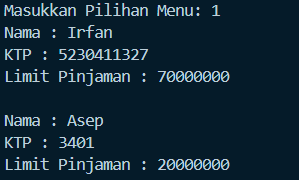
1. **TAMPIL SUBMENU KELOLA DEBITUR**

****

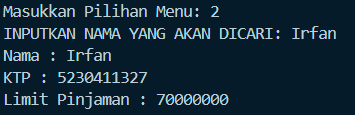
1. **TAMPIL SUBMENU KELOLA PINJAMAN**

****

1. **MENU TAMPILKAN DAFTAR SELURUH DEBITUR**

****

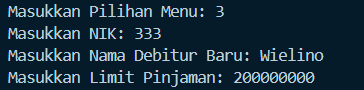
1. **TAMPILAN MENU CARI DEBITUR BERDASARKAN NAMA (Jika nama yang dicari ada)**

****

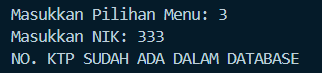
1. **TAMPILAN MENU CARI DEBITUR BERDASARKAN NAMA (Jika nama yang dicari tidak ada)**

****

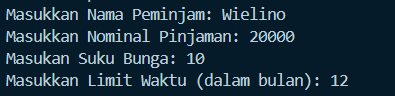
1. **Tampilan menu tambah debitur (jika KK belum ada di database)**

****

1. **Tampilan menu tambah debitur jika KK sudah ada di database**

****

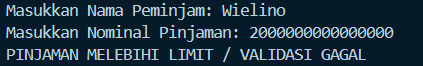
1. **Tampilan menu tambahkan pinjaman baru , jika nama debitur sudah ada dan nominal pinjam tidak melebihi limit.**

****

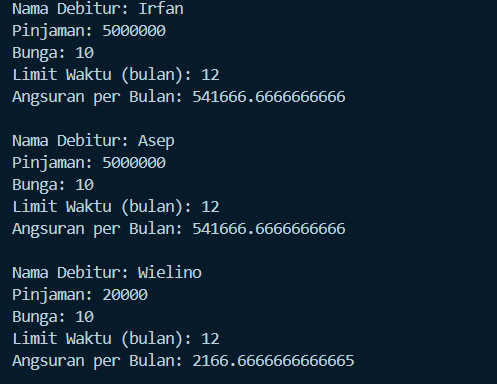
1. **Tampilan menu tambahkan pinjaman baru , jika nama debitur belum ada.**

****

1. **Tampilan menu tambahkan pinjaman baru , jika nama debitur sudah ada , tetapi nominal pinjaman melebihi limit.**

****

1. **Menu menampilkan seluruh pinjaman**

****

**BAB III**

**KESIMPULAN**

Kesimpulan dari laporan ini menyoroti penerapan konsep Pemrograman Berbasis Objek (OOP) dalam aplikasi pinjaman online (pinjol). OOP adalah paradigma pemrograman yang menggunakan "objek" untuk mewakili data dan metode yang beroperasi pada data tersebut. Dalam konteks aplikasi pinjol, konsep *inheritance* digunakan untuk mewarisi sifat dari kelas induk Debitur ke kelas turunan Pinjaman, memungkinkan reuse dan pengaturan yang lebih baik dari kode. Kelas Debitur dan Pinjaman juga mencontohkan penggunaan variabel privat (\_\_KTP), protected (\_limit), dan publik (nama) untuk mengontrol aksesibilitas data, memastikan keamanan dan enkapsulasi. Dekorator seperti @property digunakan untuk memberikan akses terkendali ke variabel-variabel ini.

UML (*Unified Modeling Language*) dalam perancangan sistem berfungsi untuk memvisualisasikan dan mendokumentasikan desain sistem secara terstruktur. U*se Case Diagram* membantu dalam menggambarkan interaksi antara aktor (pengguna) dan sistem, memberikan panduan yang jelas tentang fungsionalitas yang diharapkan dari sistem. Dalam aplikasi pinjol ini, UML digunakan untuk memetakan berbagai use case seperti kelola debitur, kelola pinjaman, dan keluar program, sehingga tim pengembang dapat memahami dan mengkomunikasikan desain sistem dengan lebih efektif.

Secara keseluruhan, laporan ini menekankan pentingnya penerapan prinsip-prinsip OOP, penggunaan UML dalam perancangan sistem, dan teknik inheritance serta kontrol akses variabel dalam pengembangan program. Ini juga dapat memastikan bahwa sistem yang dibangun dapat dikelola dan dikembangkan di kemudian hari dengan mudah.