**LAPORAN APLIKASI**

**PEMPROGRAMAN BERBASIS OBJEK**

**KELAS VIII**

T. A. Semester Ganjil 2024/2025

****

**5230411286 FIRMANTI ALHILMA S.**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS & TEKNOLOGI**

**UNIVERSITAS TEKNOLOGI YOGYAKARTA**

**YOGYAKARTA**

**2024**

1. **Perintah Program**
2. Kelola Debitur

* Tampilkan Debitur Nama(public), KTP (Private), Limit Pinjaman (Protected)
* Cari Debitur berdasarkan nama >> (Nama, KTP, Limit Pinjaman)
* Tambah Debitur berdasar KTP (jika KTP sama = menampilkan validasi)

1. Kelola Pinjaman (Debitur)

* Tambah Pinjaman berdasarkan Nama (jika Nama tidak ada = menampilkan validasi gagal, jika pinjaman > Limit = menampilkan validasi gagal
* Tampilkan pinjaman (Nama, Pinjaman, Bunga, Bulan, Angsuran(/bulan)

Nama angsuran :

Angsuran pokok = pinjaman \* bunga%

Angsuran Bulanan = angsuran pokok/bulan

Total Angsuran = Angsuran pokok = Angsuran bulan

1. **Code Program**

|  |
| --- |
| class Debitur:      def \_\_init\_\_(self, nama, ktp, limit\_pinjaman):          self.nama = nama          self.\_\_ktp = ktp          self.\_limit\_pinjaman = limit\_pinjaman      def get\_info(self):          return (self.nama, self.\_\_ktp, self.\_limit\_pinjaman)  class Pinjaman:      def \_\_init\_\_(self, debitur, pinjaman, bunga, bulan):          self.debitur = debitur          self.pinjaman = pinjaman          self.bunga = bunga          self.bulan = bulan      def hitung\_angsuran(self):          angsuran\_pokok = self.pinjaman \* (self.bunga / 100)          angsuran\_bulanan = (angsuran\_pokok + self.pinjaman) / self.bulan          total\_angsuran = angsuran\_bulanan \* self.bulan          return angsuran\_pokok, angsuran\_bulanan, total\_angsuran  class ManajemenDebitur:      def \_\_init\_\_(self):          self.debitur\_list = []      def tambah\_debitur(self, nama, ktp, limit\_pinjaman):          if any(deb.\_Debitur\_\_ktp == ktp for deb in self.debitur\_list):              print("KTP sudah ada")              return False            new\_debitur = Debitur(nama, ktp, limit\_pinjaman)          self.debitur\_list.append(new\_debitur)          print("Debitur berhasil ditambahkan.")          return True      def cari\_debitur(self, nama):          for deb in self.debitur\_list:              if deb.nama == nama:                  return deb.get\_info()          print("Debitur tidak ditemukan")          return None      def tampilkan\_semua\_debitur(self):          if not self.debitur\_list:              print("Tidak ada debitur yang terdaftar")              return            print("\n======= Daftar Debitur =======")          for deb in self.debitur\_list:              info = deb.get\_info()              print(f"Nama: {info[0]}, KTP: {info[1]}, Limit Pinjaman: {info[2]}")  class ManajemenPinjaman:      def \_\_init\_\_(self):          self.pinjaman\_list = []      def tambah\_pinjaman(self, debitur\_nama, pinjaman, bunga, bulan, manajemen\_debitur):          debitur\_info = manajemen\_debitur.cari\_debitur(debitur\_nama)            if debitur\_info is None:              print("Nama debitur tidak ditemukan")              return False            if pinjaman > debitur\_info[2]:              print("Pinjaman melebihi limit")              return False            new\_pinjaman = Pinjaman(debitur\_info[0], pinjaman, bunga, bulan)          self.pinjaman\_list.append(new\_pinjaman)            print("Pinjaman berhasil ditambahkan.")          return True      def tampilkan\_pinjaman(self):          if not self.pinjaman\_list:              print("Tidak ada pinjaman yang terdaftar.")              return            print("\n=== Daftar Pinjaman ===")          for pinj in self.pinjaman\_list:              angsuran\_pokok, angsuran\_bulanan, total\_angsuran = pinj.hitung\_angsuran()              print(f"Nama: {pinj.debitur}, Pinjaman: {pinj.pinjaman}, Bunga: {pinj.bunga}, Bulan: {pinj.bulan}, "                    f"Angsuran Pokok: {angsuran\_pokok:.2f}, Angsuran Bulanan: {angsuran\_bulanan:.2f}, "                    f"Total Angsuran: {total\_angsuran:.2f}")  manajemen\_debitur = ManajemenDebitur()  manajemen\_debitur.tambah\_debitur("Alhilma", "123", 1230000)  manajemen\_debitur.tambah\_debitur("Firman", "987", 8880000)  manajemen\_debitur.tambah\_debitur("Afra", "456", 990000)  manajemen\_debitur.tambah\_debitur("Salsa", "218", 1110000)  manajemen\_debitur.tambah\_debitur("Bila", "151", 1110000)  manajemen\_pinjam = ManajemenPinjaman()  while True:      print("\n====== Aplikasi Admin Pinjol =======")      print("1. Kelola Debitur")      print("2. Kelola Pinjaman")      print("3. Keluar")      pilihan\_menu\_utama = input("Pilih opsi (1-3): ")        if pilihan\_menu\_utama == '1':          while True:              print("\n====== Menu Debitur =======")              print("1. Tampilkan Semua Debitur")              print("2. Cari Debitur")              print("3. Tambah Debitur")              print("4. Kembali")              pilihan\_menu\_debitur = input("Pilih opsi (1-4): ")              if pilihan\_menu\_debitur == '1':                  manajemen\_debitur.tampilkan\_semua\_debitur()              elif pilihan\_menu\_debitur == '2':                  nama = input("Masukkan Nama Debitur yang dicari: ")                  info\_debitur = manajemen\_debitur.cari\_debitur(nama)                  if info\_debitur:                      print(f"Debitur Ditemukan: Nama: {info\_debitur[0]}, KTP: {info\_debitur[1]}, Limit Pinjaman: {info\_debitur[2]}")              elif pilihan\_menu\_debitur == '3':                  nama = input("Masukkan Nama Debitur: ")                  ktp = input("Masukkan KTP Debitur: ")                  limit\_pinjaman = float(input("Masukkan Limit Pinjaman: "))                  manajemen\_debitur.tambah\_debitur(nama, ktp, limit\_pinjaman)              elif pilihan\_menu\_debitur == '4':                  break              else:                  print("Pilihan tidak valid. Silakan coba lagi.")      elif pilihan\_menu\_utama == '2':          while True:              print("\n====== Menu Pinjaman =======")              print("1. Tambah Pinjaman")              print("2. Tampilkan Pinjaman")              print("3. Kembali")              pilihan\_menu\_pinjaman = input("Pilih opsi (1-3): ")              if pilihan\_menu\_pinjaman == '1':                  debitur\_nama = input("Masukkan Nama Debitur untuk Tambah Pinjaman: ")                  pinjaman = float(input("Masukkan Jumlah Pinjaman: "))                  bunga = float(input("Masukkan Bunga (%): "))                  bulan = int(input("Masukkan Lama Angsuran (bulan): "))                    manajemen\_pinjam.tambah\_pinjaman(debitur\_nama, pinjaman, bunga, bulan, manajemen\_debitur)              elif pilihan\_menu\_pinjaman == '2':                  manajemen\_pinjam.tampilkan\_pinjaman()              elif pilihan\_menu\_pinjaman == '3':                  break              else:                  print("Pilihan tidak valid. Silakan coba lagi.")      elif pilihan\_menu\_utama == '3':          print("Terimakasih Telah menggunakan program ini")          break      else:          print("Pilihan tidak valid. Silakan coba lagi.") |

Dari program diatas bisa dideskripsikan sebagai berikut:

1. Class Debitur

Digunakan untuk merepresentasikan data debitur (nasabah) yang ingin mememinjam uang. Ini mencakup informasi dasar seperti nama, nomor KTP, dan limit pinjaman yang diizinkan.

1. \_\_init\_\_(self, nama, ktp, limit\_pinjaman) : konstruktor yang menginisialisasi objek dengan nama, KTP, dan limit pinjaman.
2. get\_info(self) : digunakan untuk mengembalikan informasi debitur dalam bentuk tuple yang berisi nama, KTP, dan limit pinjaman. Ini memungkinkan akses data debitur secara terstruktur.
3. Class Pinjaman

Digunakan untuk merepresentasikan informasi tentang pinjaman yang diberikan kepada debitur. Ini mencakup detail seperti jumlah pinjaman, bunga, dan durasi pinjaman.

1. \_\_init\_\_(self, debitur, pinjaman, bunga, bulan) : konstruktor yang menginisialisasi objek dengan debitur yang memimjam, jumlah pinjaman, bunga, dan lama angsuran dalam bulan.
2. Hitung\_angsuran : digunakan untuk menghitung angsuran berdasarkan jumlah pinjaman dan bunga. Ini mengembalikan tiga nilai yaitu angsuran pokok, angsuran bulanan, dan total angsuran.
3. Class ManajemenDebitur

Digunakan untuk mengelola daftar debitur. Ini termasuk menambah debitur baru, mencari debitur berdasarkan nama, dan menampilkan semua debitur yang terdaftar

1. \_\_init\_\_(self) : konstruktor yang menginisialisasi list kosong untuk menyimpan objek debitur (debitur\_list)
2. tambah\_debitur(self, nama, ktp, limit\_pinjaman) : digunakan untuk menambahkan debitur baru kedalam daftar jika KTP belum ada di daftar. Jika KTP sudah ada, akan mencetak pesan kesalahan.
3. cari\_debitur(self, nama) : digunakan mencari debitur berdasarkan nama, jika ditemukan ia akan mengmbalikan informasi debitur, jika tidak ditemukan akan mencetak pesan kesalahan.
4. Tampilan\_semua\_debitur(self) : digunakan untuk menampilkan semua debitur yang terdaftar beserta informasi mereka, jika tidak ada debitur yang terdaftar, akan mencetak pesan bahwa tidk ada debitur
5. Class ManajemenPinjaman

Digunakan untuk mengelola daftar pinjaman yang diberikan kepada debitur. Ini termasuk menambah pinjman baru dan menampikan semua pinjaman yang terdaftar.

1. \_\_init\_\_(self) : konstruktor yang menginisialisasi list kosong untuk menyimpan objek pinjaman (pinjaman\_list)
2. Tambah\_pinjaman(self, debitur\_nama, pinjaman, bunga, bulan manajemen\_debitur) : untuk menambahkan pinjaman baru jika nama debitur ditemukan dan jumlah pinjaman tidak melebihi limit. Jika salah satu kondisi tidak terpenuhi, akan mencetak pesanan kesalahan.
3. Tampilkan\_pinjaman(self) : metode ini menampilkan semua Riwayat transaksi pinjaman beserta detailnya seperti angsuran pokok dan total angsuran.
4. **Hasil Running**





