**Minggu, 1 September 2024**

Nama : Kadek Wisnu Parijata Putra

NIM : 21120122140036

Prodi : Teknik Komputer / 2022

Mata Kuliah : RPLBK / B

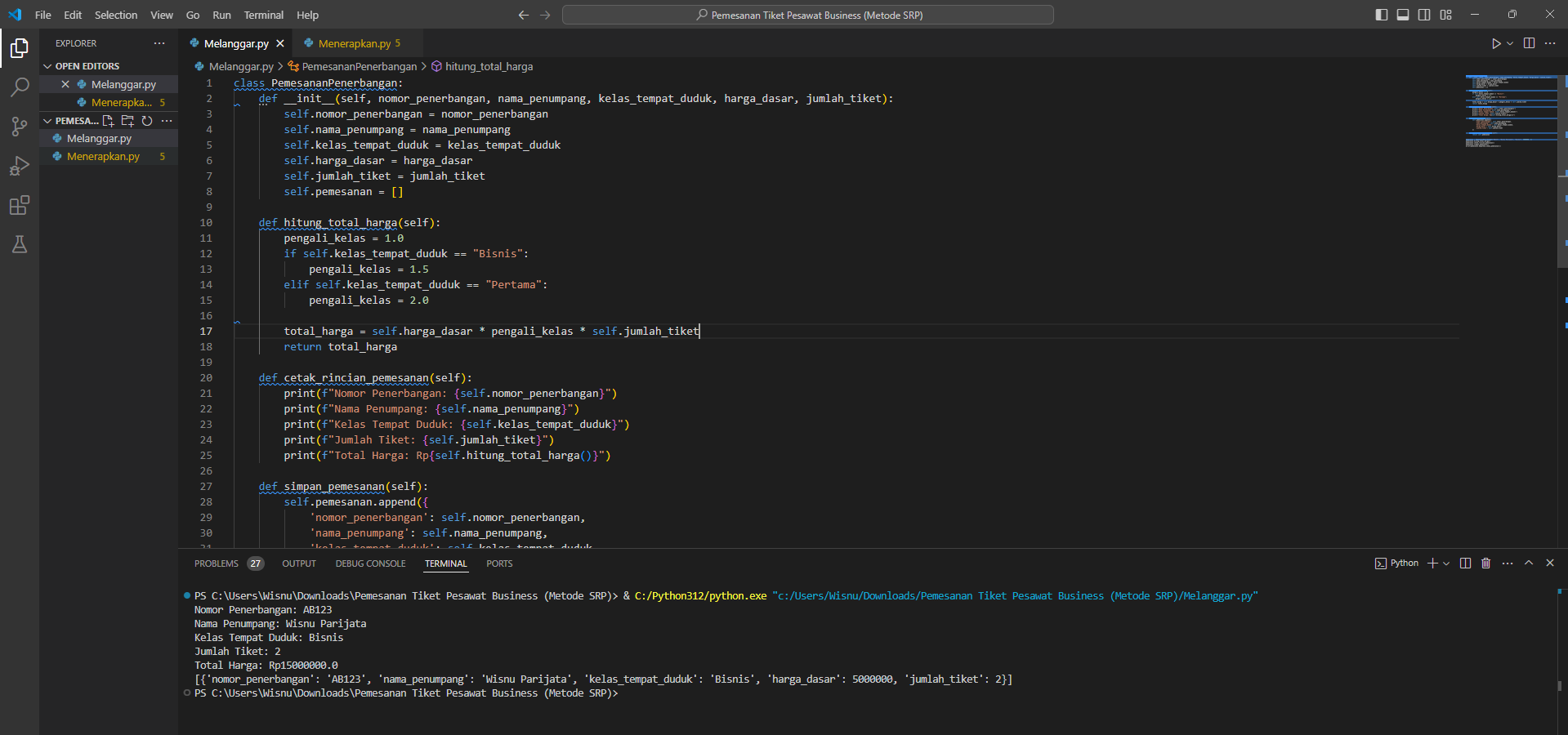
***(S) Single Responsibility Principle***

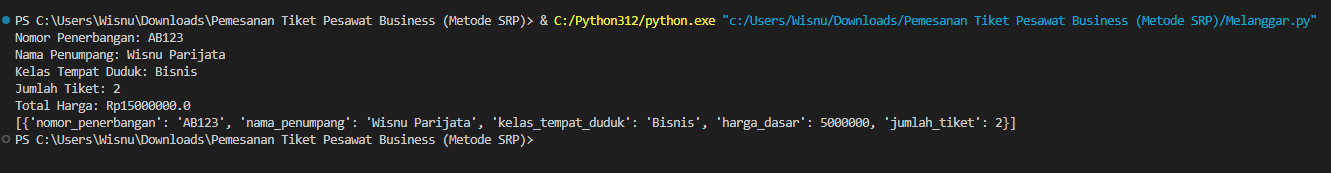
*Single Responsibility Principle* (SRP) adalah salah satu dari Lima Prinsip dalam SOLID Principles, SRP ini memiliki prinsip dimana, sebuah kelas atau method seharusnya memiliki satu tanggung jawab atau satu pekerjaan saja, setiap kelas harus memiliki satu alasan untuk berubah, yaitu hanya ada satu tanggung jawab yang harus dipenuhi oleh kelas tersebut.

1. ***Source Code* yang Melanggar Aturan *Single Responsibility Principle* (SRP) :**

|  |
| --- |
| class PemesananPenerbangan:  def \_\_init\_\_(self, nomor\_penerbangan, nama\_penumpang, kelas\_tempat\_duduk, harga\_dasar, jumlah\_tiket):  self.nomor\_penerbangan = nomor\_penerbangan  self.nama\_penumpang = nama\_penumpang  self.kelas\_tempat\_duduk = kelas\_tempat\_duduk  self.harga\_dasar = harga\_dasar  self.jumlah\_tiket = jumlah\_tiket  self.pemesanan = []  def hitung\_total\_harga(self):  pengali\_kelas = 1.0  if self.kelas\_tempat\_duduk == "Bisnis":  pengali\_kelas = 1.5  elif self.kelas\_tempat\_duduk == "Pertama":  pengali\_kelas = 2.0    total\_harga = self.harga\_dasar \* pengali\_kelas \* self.jumlah\_tiket  return total\_harga  def cetak\_rincian\_pemesanan(self):  print(f"Nomor Penerbangan: {self.nomor\_penerbangan}")  print(f"Nama Penumpang: {self.nama\_penumpang}")  print(f"Kelas Tempat Duduk: {self.kelas\_tempat\_duduk}")  print(f"Jumlah Tiket: {self.jumlah\_tiket}")  print(f"Total Harga: Rp{self.hitung\_total\_harga()}")  def simpan\_pemesanan(self):  self.pemesanan.append({  'nomor\_penerbangan': self.nomor\_penerbangan,  'nama\_penumpang': self.nama\_penumpang,  'kelas\_tempat\_duduk': self.kelas\_tempat\_duduk,  'harga\_dasar': self.harga\_dasar,  'jumlah\_tiket': self.jumlah\_tiket  })  def dapatkan\_semua\_pemesanan(self):  return self.pemesanan  pemesanan = PemesananPenerbangan("AB123", "Wisnu Parijata", "Bisnis", 5000000, 2)  pemesanan.hitung\_total\_harga()  pemesanan.cetak\_rincian\_pemesanan()  pemesanan.simpan\_pemesanan()  print(pemesanan.dapatkan\_semua\_pemesanan()) |

Hasil *Running* :





Kode diatas termasuk melanggar prinsip *Single Responsibility Principle* (SRP) karena class PemesananPenerbangan memiliki lebih dari satu tanggung jawab. SRP diyatakan bahwa setiap kelas harus memiliki satu alasan untuk berubah, yang berarti bahwa kelas hanya boleh memiliki satu tugas atau tanggung jawab. Namun, class PemesananPenerbangan menangani beberapa tanggung jawab sekaligus diantaranya:

1. Menghitung total harga tiket berdasarkan kelas tempat duduk.

2. Mencetak rincian pemesanan.

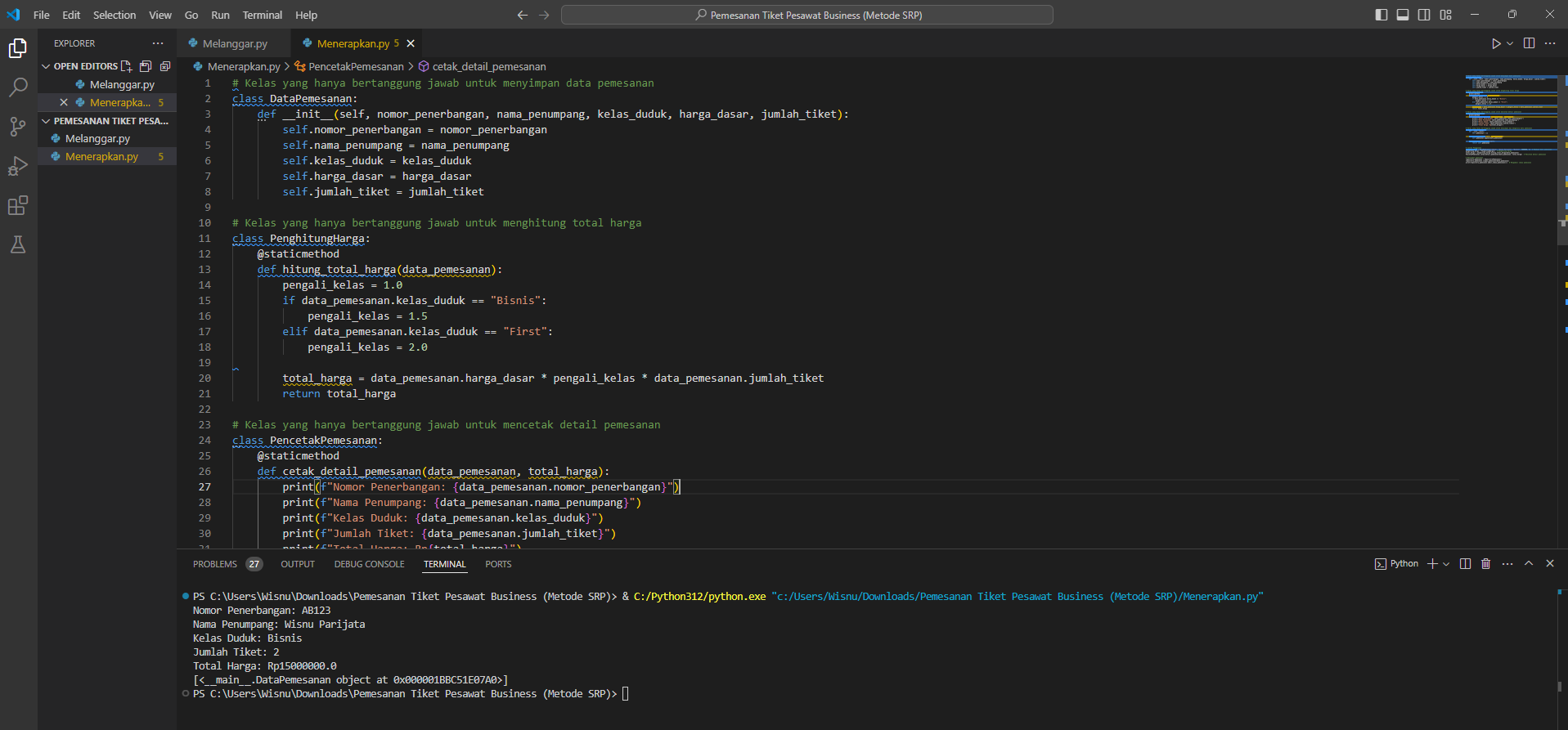
3. Menyimpan pemesanan ke dalam daftar.

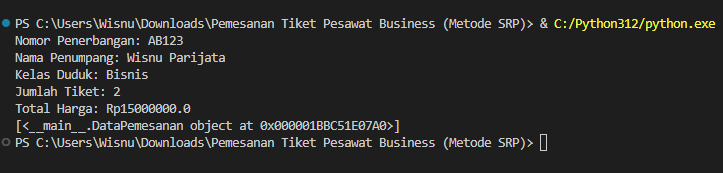
4. Mengelola daftar semua pemesanan yang disimpan.

1. ***Source Code* yang Menerapkan Aturan *Single Responsibility Principle* (SRP) :**

|  |
| --- |
| **# Kelas yang hanya bertanggung jawab untuk menyimpan data pemesanan**  **class DataPemesanan:**  **def \_\_init\_\_(self, nomor\_penerbangan, nama\_penumpang, kelas\_duduk, harga\_dasar, jumlah\_tiket):**  **self.nomor\_penerbangan = nomor\_penerbangan**  **self.nama\_penumpang = nama\_penumpang**  **self.kelas\_duduk = kelas\_duduk**  **self.harga\_dasar = harga\_dasar**  **self.jumlah\_tiket = jumlah\_tiket**  **# Kelas yang hanya bertanggung jawab untuk menghitung total harga**  **class PenghitungHarga:**  **@staticmethod**  **def hitung\_total\_harga(data\_pemesanan):**  **pengali\_kelas = 1.0**  **if data\_pemesanan.kelas\_duduk == "Bisnis":**  **pengali\_kelas = 1.5**  **elif data\_pemesanan.kelas\_duduk == "First":**  **pengali\_kelas = 2.0**    **total\_harga = data\_pemesanan.harga\_dasar \* pengali\_kelas \* data\_pemesanan.jumlah\_tiket**  **return total\_harga**  **# Kelas yang hanya bertanggung jawab untuk mencetak detail pemesanan**  **class PencetakPemesanan:**  **@staticmethod**  **def cetak\_detail\_pemesanan(data\_pemesanan, total\_harga):**  **print(f"Nomor Penerbangan: {data\_pemesanan.nomor\_penerbangan}")**  **print(f"Nama Penumpang: {data\_pemesanan.nama\_penumpang}")**  **print(f"Kelas Duduk: {data\_pemesanan.kelas\_duduk}")**  **print(f"Jumlah Tiket: {data\_pemesanan.jumlah\_tiket}")**  **print(f"Total Harga: Rp{total\_harga}")**  **# Kelas yang hanya bertanggung jawab untuk menyimpan dan mengelola data pemesanan**  **class RepositoryPemesanan:**  **def \_\_init\_\_(self):**  **self.pemesanan = []**  **def simpan\_pemesanan(self, data\_pemesanan):**  **self.pemesanan.append(data\_pemesanan)**  **def ambil\_semua\_pemesanan(self):**  **return self.pemesanan**  **# Contoh Penggunaan**  **data\_pemesanan = DataPemesanan("AB123", "Wisnu Parijata", "Bisnis", 5000000, 2) # Membuat data pemesanan**  **penghitung\_harga = PenghitungHarga() # Menghitung harga**  **total\_harga = penghitung\_harga.hitung\_total\_harga(data\_pemesanan)**  **PencetakPemesanan.cetak\_detail\_pemesanan(data\_pemesanan, total\_harga) # Mencetak detail pemesanan**  **# Menyimpan pemesanan**  **repository\_pemesanan = RepositoryPemesanan()**  **repository\_pemesanan.simpan\_pemesanan(data\_pemesanan)**  **print(repository\_pemesanan.ambil\_semua\_pemesanan()) # Mengambil semua pemesanan** |

Hasil *Running* :

****

****

Kode diatas menerapkan prinsip *Single Responsibility Principle* (SRP). Karena setiap kelas dalam kode tersebut memiliki satu tanggung jawab yang jelas dan spesifik. Bahwa setiap kelas harus memiliki satu alasan untuk berubah, yang berarti kelas tersebut hanya melakukan satu tugas atau fungsi tertentu, diantaranya sebagai kode berikut :

1. ‘DataPemesanan’ hanya bertanggung jawab untuk menyimpan informasi atau data pemesanan.

2. ‘PenghitungHarga’ bertanggung jawab untuk menghitung total harga berdasarkan data pemesanan yang diberikan.

3. ‘PencetakPemesanan’ fokus pada tugas mencetak detail pemesanan ke *output*.

4. ‘RepositoryPemesanan’ bertugas untuk menyimpan dan mengelola daftar semua pemesanan yang dibuat.

Dengan memisahkan tanggung jawab ke dalam kelas yang spesifik, kode menjadi lebih mudah dipelihara, dikelola dan lebih fleksibel dalam hal perubahan.