LAPORAN TUGAS BESAR
MATA KULIAH BEPIKIR
KOMPUTASIONAL (WI1102)
TAHUN 2024

Analisis Algoritma Simulasi *Coffee Machine* Melalui Pendekatan Prinsip *Computational Thinking*.

Anggota Kelompok 13

Laurenisus Dani Rendragraha	(19624272)
Mineva Azzahra	(19624227)
Muhammad Faiz Alfada Dharma	(19624244)
Muhammad Zulfa Fauzan Nurhuda	(19624258)

MATA KULIAH WAJIB KURIKULUM
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA - KOMPUTASI
INSTITUT TEKNOLOGI BANDUNG
KOTA BANDUNG
TAHUN 2024

LAPORAN TUGAS BESAR

Computational Thinking

Analisis Algoritma Simulasi *Coffee Macine* Melalui Pendekatan Prinsip *Computational Thinking*.

1. Peran Masing-Masing Anggota Kelompok dalam Tugas ini

Nama Anggota	Peran
Laurenisus Dani Rendragraha	1. Membantu mengurai masalah masalah pada Dekomposisi 2. Menyusun presentasi di Canva 3. Menyusun laporan tugas 4. Mengedit video presentasi
Mineva Azzahra	1. Membantu mengenali hal-hal pada bagian Pengenalan Pola 2. Menyusun presentasi di Canva 3. Menyusun laporan tugas
Muhammad Faiz Alfada Dharma	1. Membantu menyeleksi masalah pada bagian Abstraksi 2. Menyusun presentasi di Canva 3. Menyusun laporan tugas
Muhammad Zulfa Fauzan Nurhuda	1. Membantu membuat langkah-langkah sistematis pada Algoritma 2. Membuat kode dalam python 3. Menyusun presentasi di Canva 4. Menyusun laporan tugas

2. Deskripsi dan Spesifikasi Program

Di era digital sekarang ini, *computational thinking* (atau berpikir komputasional) menjadi kemampuan penting yang perlu dimiliki mahasiswa. Kemampuan ini meliputi cara memecahkan masalah, merancang langkah-langkah sistematis, dan menyederhanakan proses yang kompleks.

Salah satu cara untuk menerapkan prinsip berpikir komputasional dalam kehidupan sehari-hari adalah dengan membuat simulasi algoritma *coffee machine*. *Coffee machine* memiliki proses yang terstruktur, seperti memilih jenis kopi, menentukan komposisi, jumlah pesanan, dan menangani pembayaran secara otomatis.

Berbagai fitur yang tersedia didalam algoritma *coffee machine* yang telah dibuat oleh kelompok 13, antara lain

1. Terhubung dengan Google Spreadsheet sebagai basis data,
2. Dapat menangani banyak pesanan sekaligus dari pengguna,
3. Menu yang dapat dikustomisasi sesuai data yang tersedia,
4. Dapat memilih suhu kopi yang diinginkan,
5. Dapat memilih sendiri komposisi dari bahan tambahan (mencakup gula, susu, krimer, dan cokelat),
6. Mendukung dua metode pembayaran, yaitu secara tunai dan QRIS (simulasi),
7. Dapat membuat simulasi, ketika pengguna telah memesan kopi daring, pengguna hanya tinggal menunjukkan kode QR untuk konfirmasi,
8. Pencatatan pesanan yang detail dan akurat,
9. *Error handling* yang tertata rapi,
10. Informasi untuk user di setiap langkah cukup jelas,
11. Terdapat validasi untuk setiap masukan pengguna,
12. Terdapat menu khusus admin yang diamankan dengan kode admin, menu admin mencakup shutdown program dan restock jenis kopi.

3. Analisis Empat Prinsip Utama *Computational Thinking*

a. Dekomposisi

Dekomposisi adalah proses memecah masalah besar menjadi bagian-bagian yang lebih kecil dan lebih mudah dipecahkan. Melalui pembagian masalah ini, akan lebih mudah untuk berfokus pada satu bagian dari masalah yang ingin dipecahkan pada saat itu.

Dekomposisi dari masalah-masalah yang ada dalam simulasi *coffee machine* antara lain, sebagai berikut

1. Pengelolaan Data Kopi:

- Nama kopi dan ID kopi,
- Jenis kopi (Arabica, Robusta, dll.),
- Harga kopi,
- Stok kopi yang tersedia,

2. Interaksi dengan Pengguna:

- Nama pembeli,
- Input pilihan kopi oleh pengguna,
- Pemilihan suhu kopi (hangat atau dingin),

- Pemilihan jumlah kopi yang dipesan,
- Pengaturan komposisi,

3. **Pengaturan Komposisi Bahan:**
   - Takaran gula, krimer, susu, dan cokelat (0-5)
   - Preferensi pengguna terhadap komposisi,
4. **Proses Pembayaran:**
   - Total harga pesanan,
   - Metode pembayaran (Tunai atau QRIS),
   - Proses validasi pembayaran (konfirmasi input kode QR, pengecekan jumlah uang tunai),
5. **Pembuatan Pesanan:**
   - Proses penyajian kopi (mesin hangat/dingin),
   - Peralatan yang digunakan (maintenance alat, kondisi mesin),
   - Proses pembuatan kopi (waktu, alat pengaduk, suhu optimal),
6. **Pencatatan dan Log Transaksi:**
   - Log data penjualan (jenis kopi, jumlah, suhu, harga total),
   - Metode pembayaran yang dicatat,
   - Waktu transaksi,
   - Update stok kopi dalam database,
7. **Pengelolaan Stok dan Maintenance:**
   - Pemantauan persediaan kopi,
   - Pengisian ulang stok (proses restocking),
   - Maintenance alat (kebersihan dan perbaikan mesin),
8. **Validasi dan Keamanan:**
   - Validasi input pengguna (menghindari input invalid),
   - Keamanan data transaksi,
   - Pembatasan waktu input untuk menghindari antrian yang terhenti,
9. **Antarmuka Pengguna:**
   - Desain tampilan menu kopi,
   - Pesan kesalahan dan konfirmasi,
   - Informasi pembayaran dan petunjuk input,
10. **Fungsi Tambahan:**
    - Pembuatan QR Code untuk pembayaran QRIS,
    - Pemberitahuan untuk konfirmasi ulang pesanan,
    - Sistem timeout jika pengguna tidak merespons dalam waktu tertentu,
11. **Manajemen Data Bahan Baku:**
    - Jenis gula yang digunakan (misalnya, gula pasir, gula tebu),
    - Jenis susu (susu full cream, skim, dll.),
    - Merek krimer yang tersedia,
    - Ketersediaan bahan tambahan lainnya (cokelat bubuk, kayu manis).

b. Abstraksi

Abstraksi adalah menyederhanakan masalah dengan mengabaikan detail yang tidak penting, sehingga kita bisa fokus pada hal-hal utama yang perlu diselesaikan.

Abstraksi dari masalah-masalah yang sudah dijabarkan diatas dengan penggolongan berdasarkan proses antara lain, sebagai berikut

1. **Proses pemilihan jenis pesanan kopi**

   Data Penting:

   - Nomor pilihan kopi,
   - Nama kopi,
   - Harga kopi,
   - Ketersediaan stok kopi,

   Data tidak diperlukan:

   - Deskripsi lengkap dari setiap kopi Informasi tambahan terkait supplier kopi,

2. **Proses pemilihan suhu pesanan kopi**

   Data Penting:

   - Pilihan suhu (hangat atau dingin),

   Data tidak diperlukan:

   - Detail suhu spesifik (misalnya, dalam derajat Celcius),
   - Informasi teknis tentang cara memanaskan atau mendinginkan kopi,

3. **Proses pemilihan komposisi bahan tambahan**

   Data Penting:

   - Jumlah takaran gula (0-5),
   - Jumlah takaran krimer (0-5),
   - Jumlah takaran susu (0-5),
   - Jumlah takaran cokelat (0-5),

   Data tidak diperlukan:

   - Detail Komposisi bahan secara kimiawi,
   - Merek spesifik bahan tambahan,

4. **Proses menentukan jumlah pesanan kopi**

   Data Penting:

   - Jumlah kopi yang ingin dipesan (harus lebih dari 0),

   Data tidak diperlukan:

   - Informasi terkait paket pesanan lain (misalnya, paket pesanan dengan diskon),

5. **Proses pembayaran pesanan kopi**

   Data Penting:

   - Pilihan metode pembayaran (tunai atau QRIS),
   - Kode konfirmasi QR jika memilih metode QRIS,
   - Total uang yang diberikan dalam metode tunai,

   Data tidak diperlukan:

   - Informasi tentang latar belakang QRIS atau teknologi yang digunakan,
   - Detail teknis dari sistem pembayaran,
   - Keamanan pembayaran,

6. **Proses pencatatan penjualan kopi**

   Data Penting:

   - Nama kopi yang terjual,
   - Suhu kopi yang dipesan,
   - Komposisi tambahan yang dipilih,
   - Harga total penjualan,
   - Jumlah kopi yang terjual,
   - Metode pembayaran yang dipilih,

   Data tidak diperlukan:

   - Histori penjualan sebelumnya,
   - Rincian log teknis dari database (misalnya, timestamp detail),

7. **Proses pada lingkup khusus admin**

   Data Penting:

   - Kode admin untuk autentikasi,
   - Jenis kopi yang ingin direstock,
   - Jumlah kopi yang ingin di restock,

   Data tidak diperlukan:

   - Jenis autentikasi yang digunakan,
   - Teknologi autentikasi yang digunakan,

8. **Proses scan QR pesanan media daring (simulasi)**

   Data Penting:

   - Gambar dalam bentuk kode QR,
   - Kode yang terkandung dalam kode QR,
   - Data yang tersedia pada basis data,

   Data tidak diperlukan:

   - Situs yang digunakan untuk memesan kopi secara daring.

c. Pengenalan pola

Pengenalan pola adalah kemampuan untuk mengenali kesamaan dalam sekumpulan data atau masalah yang berbeda. Dengan mengenali pola, kita bisa menerapkan solusi yang serupa pada masalah yang berbeda.

Pengenalan pola yang berhasil diamati dari masalah-masalah yang telah dikelompokkan dari proses abstraksi antara lain, sebagai berikut

1. **Pola Interaksi Pengguna**
   - Pengguna selalu diberikan pilihan dan instruksi jelas,
   - Input pengguna divalidasi dan ditangani jika terjadi kesalahan,
   - Pengguna dapat membatalkan proses kapan saja dengan memasukkan 'x',
2. **Pola Proses Pemesanan**
   - Setiap pemesanan mengikuti alur: pilih kopi → pilih suhu → atur komposisi → tentukan jumlah,
   - Setelah pesanan dikonfirmasi, dilanjutkan dengan pembayaran,
3. **Pola Validasi dan Error Handling**
   Program selalu mengecek apakah masukkan pengguna valid dan memberikan pesan kesalahan yang informatif,
   - Batas waktu input mencegah program menumpuk pesanan jika tidak ada aktivitas,
4. **Pola Pengelolaan Data**
   - Penggunaan database untuk menyimpan dan mengupdate data stok dan penjualan,
   - Pemisahan fungsi-fungsi untuk kemudahan pemeliharaan.

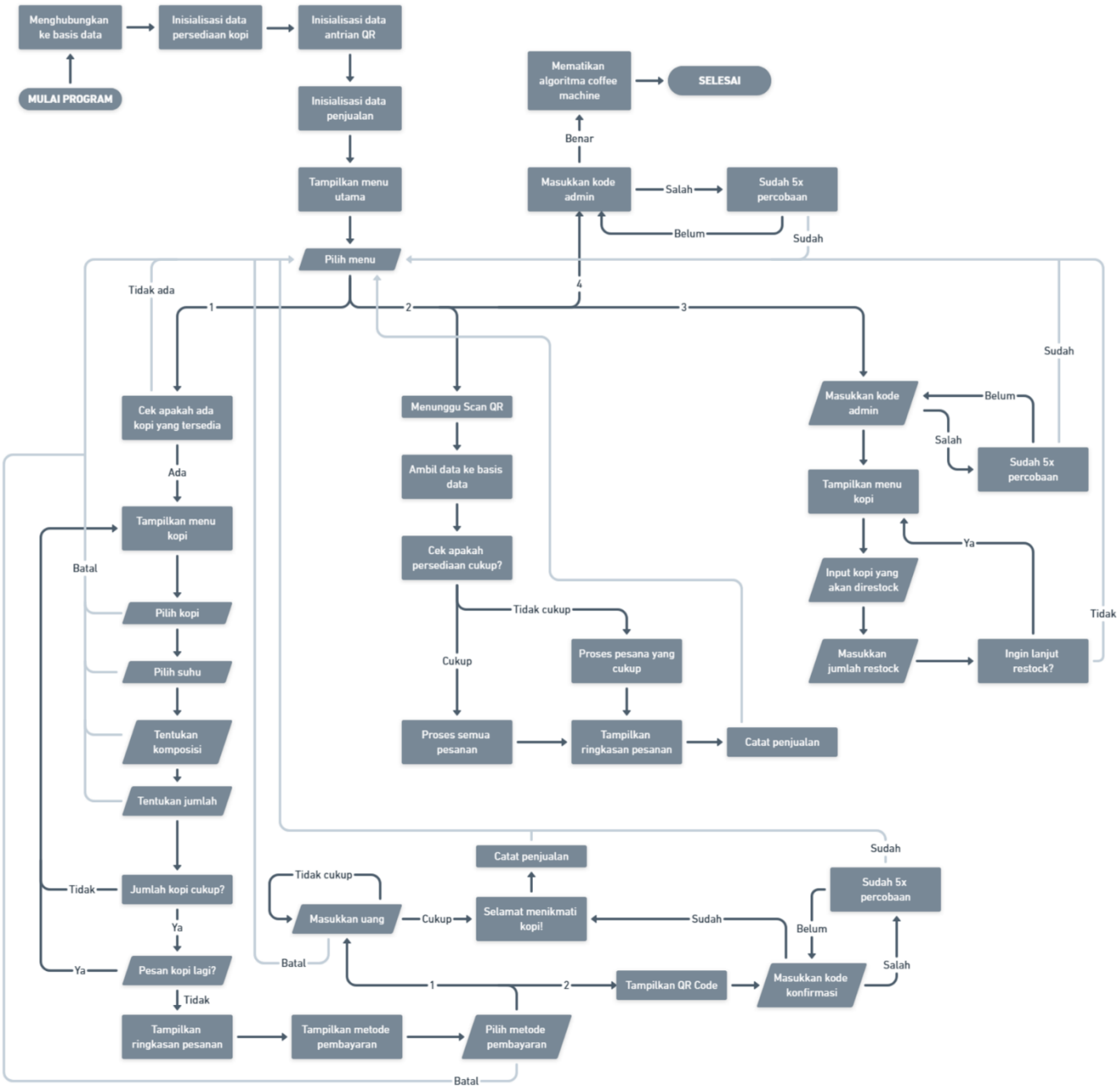
d. Algoritma

Algoritma adalah sekumpulan langkah atau instruksi yang jelas dan teratur untuk menyelesaikan suatu masalah. Algoritma harus spesifik dan bisa diikuti dengan urutan yang tepat.

Algoritma dari hasil akhir tiga proses sebelumnya dapat diakses melalui beberapa media antara lain, sebagai berikut

1. **Media Kode Pemrograman Python**
   Algoritma melalui kode pemrograman python dapat diakses melalui Google Colaboratory atau melalui Github.

2. Media Flowchart atau Diagram Alir



4. Analisis dan Dokumentasi Program *Coffee Machine*

a. Import Library yang Diperlukan

Berikut ini adalah semua library yang digunakkan dalam program python untuk mendukung berjalannya algoritma *coffee machine*.

```
import sys
import os
import random
import string

import gspread
from google.oauth2.service_account import Credentials
from inputimeout import inputimeout, TimeoutOccurred
import cv2
import qrcode_terminal

from dataclasses import dataclass
from typing import Dict, List, Tuple, Optional
from functools import lru_cache
```

Penjelasan masing masing library yang digunakkan serta fungsinya dalam program python adalah sebagai berikut

1. **Library** `sys` digunakan untuk berinteraksi dengan interpreter Python. Dalam konteks program *coffee machine* ini, library ini digunakan untuk menonaktifkan program.

2. **Library** `os` digunakan untuk berinteraksi dengan *operating system*. Dalam konteks program *coffee machine* ini, library ini digunakan untuk menentukan jalur file secara dinamis, seperti jalur menuju file kredensial Google Sheets. Hal ini dapat meminimalisir *error* dalam keterbacaan jalur file tujuan.

3. **Library** `random` digunakan digunakan untuk menghasilkan data acak. Dalam konteks program *coffee machine* ini, library ini digunakan untuk membuat string angka acak, misalnya, untuk membuat kode QR acak untuk pembayaran QRIS.

4. **Library** `string` berisi fungsi untuk memproses string standar Python. Dalam konteks program *coffee machine* ini, library ini dikombinasikan dengan library `random` untuk mendapatkan string angka acak.

5. **Library** `gspread` digunakan untuk berinteraksi dengan Google Sheets. Dalam konteks program *coffee machine* ini, library ini memungkinkan program untuk membaca, memperbarui, dan menulis data ke Google Sheets, yang digunakan sebagai basis data untuk persediaan kopi, penjualan, dan antrian QR.

6. **Library** `google.oauth2.service_account.Credentials` Library ini digunakan untuk autentikasi ke Google Sheets API menggunakan kredensial *service account*. Hal ini memungkinkan program untuk mengakses dan memanipulasi data di Google Sheets secara aman.

7. **Library** `inputimeout` digunakan untuk mengambil input dari pengguna dengan batas waktu tertentu. Dalam konteks program *coffee machine* ini, library ini membantu menangani input dengan waktu tunggu yang ditentukan untuk mencegah algoritma terhenti karena tidak ada aktivitas pengguna.

8. **Library** `cv2` adalah library yang dibentuk oleh OpenCV, yang digunakan untuk memproses gambar dan video. Dalam konteks program *coffee machine* ini, library ini digunakan untuk memindai QR Code dengan kamera, yang diperlukan untuk memverifikasi pesanan berbasis QR.

9. **Library** `qrcode_terminal` digunakan untuk menghasilkan dan menampilkan QR Code langsung di terminal. Dalam konteks program *coffee machine* ini, library ini digunakan untuk menampilkan QR Code pembayaran QRIS yang dapat dipindai oleh pengguna.

10. **Library** `dataclasses` menyediakan decorator `@dataclass` yang memudahkan pembuatan kelas data. Dalam konteks program *coffee machine* ini, library ini digunakan untuk membentuk dan mendefinisikan beberapa struktur data.

11. **Library** `typing` menyediakan petunjuk tipe untuk Python. Dalam program ini, `Dict`, `List`, `Tuple`, dan `Optional` digunakan untuk memberikan informasi tipe yang jelas, sehingga dapat meningkatkan keterbacaan kode.

12. **Library** `functools.lru_cache` adalah dekorator yang digunakan untuk menyimpan hasil fungsi yang dipanggil berulang-ulang, yang dapat meningkatkan efisiensi program. Dalam konteks program *coffee machine* ini, library ini digunakan untuk menyimpan data kopi dari Google Sheets agar tidak perlu mengambil data dari internet setiap kali dipanggil. Hal ini dapat meningkatkan kecepatan dan efisiensi program.

b. Inisialisasi Kelas yang Berisi Konfigurasi Global

```
class Config:
    """Kelas konfigurasi untuk menyimpan pengaturan program."""

    SHEET_ID = "google_sheet_id"
    SERVICE_ACCOUNT_FILE = os.path.join(
        os.getcwd(),
        "path_to_key_file",
        "key_file.json",
    )
    KODE_ADMIN = 1234567890
    TIMEOUT_DURATION = 60  # Durasi timeout input dalam detik
```

Kelas Config ini berisi semua konfigurasi yang dibutuhkan dalam menjalankan program coffee machine.

Kamus Variabel

1. **Variabel** `SHEET_ID`
   - Tipe: `string`.
   - Fungsi: Digunakan untuk menyimpan ID Google Sheet tempat data akan disimpan, seperti log transaksi atau konfigurasi lainnya.
2. **Variabel** `SERVICE_ACCOUNT_FILE`
   - Tipe: `string`.
   - Fungsi: Menyimpan jalur file kredensial Google Service Account yang dibutuhkan untuk autentikasi dan akses ke Google Sheets.
3. **Variabel** `KODE_ADMIN`
   - Tipe: `integer`.
   - Fungsi: Sebagai kode keamanan untuk mengakses fitur-fitur admin dalam program, seperti reset data atau perubahan konfigurasi.

4. **Variabel** `TIMEOUT_DURATION`
    - Tipe: `integer`.
    - Fungsi: Menentukan durasi maksimum waktu tunggu (timeout) untuk input dari pengguna. Jika waktu habis, sistem secara otomatis akan mengakhiri operasi atau mengambil tindakan default.

c. Inisialisasi Beberapa Dataclass

```python
@dataclass
class KopiData:
    """Kelas untuk menyimpan data kopi."""
    nama: str
    harga: int
    sisa: int
    row_number: int
    nomor: int = 0

@dataclass
class Komposisi:
    """Kelas untuk menyimpan komposisi bahan tambahan."""
    gula: int
    krimer: int
    susu: int
    cokelat: int

@dataclass
class PesananItem:
    """Kelas untuk menyimpan item pesanan."""
    kopi: KopiData
    jumlah: int
    suhu: str
    komposisi: Komposisi
```

`@dataclass` adalah fitur Python yang diperkenalkan di versi 3.7 untuk menyederhanakan pembuatan kelas yang berfungsi sebagai *data containers*.

Dataclass dalam Program

1. **@dataclass** `KopiData`
    - Dataclass ini berfungsi sebagai *data container* untuk menyimpan data kopi.
    - Properti:
        - `nama` – Nama jenis kopi (contoh: "Latte", "Espresso", dll).

- `harga` – Harga kopi per unit.
- `sisa` – Stok kopi yang masih tersedia.
- `row_number` – Nomor baris yang berkaitan dengan data di Google Sheets.
- `nomor` – Nomor kopi yang telah di-*assign* sesuai urutan.

2. @dataclass `Komposisi`

- Dataclass ini berfungsi sebagai *data container* untuk menyimpan data komposisi.
- Properti:
  - `gula` – Jumlah takaran gula dalam kopi.
  - `krimer` – Jumlah takaran krimer dalam kopi.
  - `susu` – Jumlah tarakan susu dalam kopi.
  - `cokelat` – Jumlah takaran cokelat dalam kopi.

3. @dataclass `PesananItem`

- Dataclass ini berfungsi sebagai *data container* untuk menyimpan data detail item tiap pesanan kopi.
- Properti:
  - `kopi` – Data kopi yang ada didalam pesanan.
  - `jumlah` – Jumlah kopi yang dipesan.
  - `suhu` – Suhu kopi yang dipesan.
  - `komposisi` – Konfigurasi komposisi dari kopi yang dipesan.

d. Manajer untuk Basis Data

```
class DatabaseManager:
    """Kelas untuk mengelola operasi database dengan Google Sheets."""

    def __init__(self):
        """Inisialisasi koneksi ke Google Sheets."""
        kredensial = Credentials.from_service_account_file(
            Config.SERVICE_ACCOUNT_FILE,
            scopes=["https://www.googleapis.com/auth/spreadsheets"],
        )
        client = gspread.authorize(kredensial)
        lembar_kerja = client.open_by_key(Config.SHEET_ID)
        self.persediaan = lembar_kerja.worksheet("PersediaanKopi")
        self.penjualan = lembar_kerja.worksheet("DataPenjualan")
        self.antrian_qr = lembar_kerja.worksheet("AntrianPesananQR")

    ...
```

Kelas **`DatabaseManager`** digunakan untuk mengelola berbagai operasi terkait *database* yang terintegrasi dengan Google Sheets, seperti mengelola persediaan kopi, mencatat penjualan, dan mengatur data pesanan QR.

Properti dalam Kelas

1. **`persediaan`** – Mengacu pada *worksheet* "PersediaanKopi" di Google Sheets yang berisi data persediaan kopi.
2. **`penjualan`** – Mengacu pada *worksheet* "DataPenjualan" untuk mencatat transaksi penjualan.
3. **`antrian_qr`** – Mengacu pada *worksheet* "AntrianPesananQR" untuk data pesanan berbasis QR.

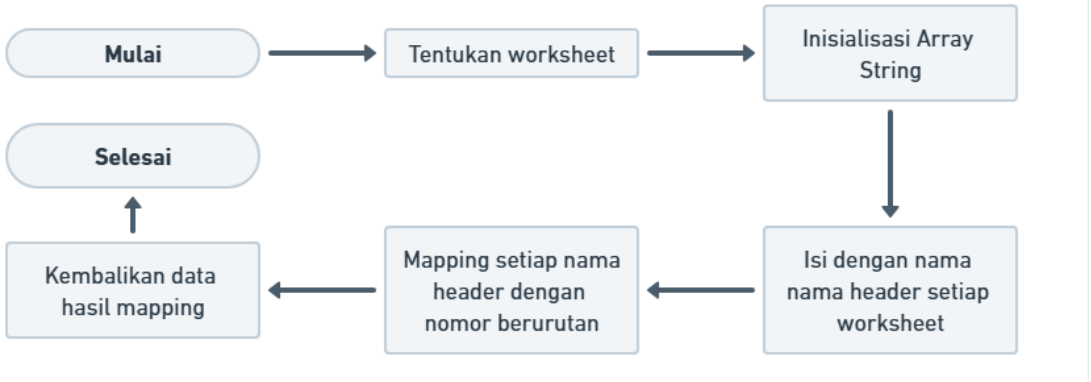
Fungsi Fungsi dalam Kelas

1. **Fungsi `get_column_indices`**

```
def get_column_indice(self, worksheet: gspread.Worksheet) -> Dict[str, int]:
    """Mendapatkan mapping nama kolom ke indeks kolom."""
    header = worksheet.row_values(1)
    return {name: idx + 1 for idx, name in enumerate(header)}
```

Fungsi ini bertujuan untuk mendapatkan mapping nama kolom ke indeks kolom dari baris header di *worksheet* tertentu.

- Flowchart dari fungsi



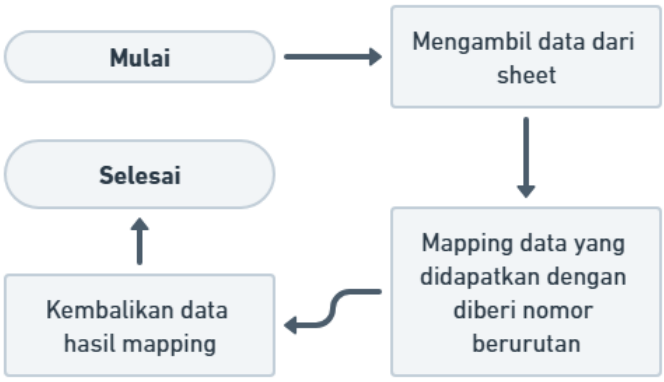
- Variabel dalam fungsi
  - `worksheet` – *Worksheet* Google Sheets yang ingin diproses.
  - `headers` – Daftar nama kolom yang ada di baris pertama *worksheet*.

2. Fungsi `ambil_data_kopi`

```
@lru_cache(maxsize=None)
def (self) -> Dict[str, KopiData]:
    """Mengambil data kopi dari lembar `PersediaanKopi`."""
    data_kopi = self.persediaan.get_all_records()
    return {
        baris["Jenis Kopi"]: KopiData(
            nama=baris["Jenis Kopi"],
            harga=int(baris["Harga"]),
            sisa=int(baris["Sisa Persediaan"]),
            row_number=i + 2,
        )
        for i, baris in enumerate(data_kopi)
    }
```

Fungsi ini bertujuan untuk mengambil data persediaan kopi dari *worksheet* "PersediaanKopi" dan mengonversinya menjadi objek `KopiData`. Fungsi ini menggunakan *cache* untuk meningkatkan efisiensi program.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `data_kopi` – Daftar *dictionary* yang merepresentasikan data di *worksheet*.
  - `baris` – *Dictionary* berisi data dari tiap baris *worksheet*.
  - `i` – Indeks dari baris.

3. Fungsi `update_stock`

```
def update_stok(self, kopi: KopiData, jumlah_terjual: int) -> None:
    """Mengupdate stok kopi di lembar `PersediaanKopi`."""
    stok_baru = max(0, kopi.sisa - jumlah_terjual)
    self.persediaan.update_cell(kopi.row_number, 3, stok_baru)
    kopi.sisa = stok_baru
```

Fungsi ini bertujuan untuk emperbarui stok kopi di *worksheet* "PersediaanKopi" setelah adanya penjualan.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `kopi` – Objek `KopiData` yang ingin diperbarui stoknya.

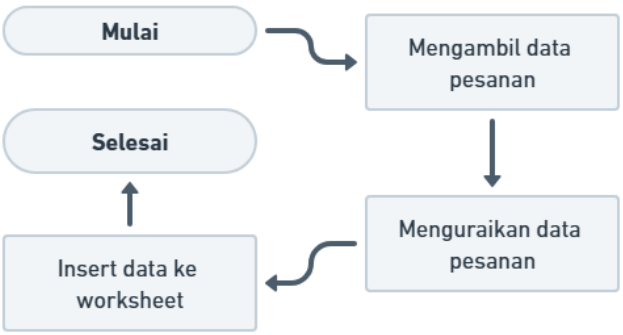
- `jumlah_terjual` – Jumlah pesanan kopi yang terjual.
- `stock_baru` – Stok baru setelah pengurangan pesanan.

4. Fungsi `catat_penjualan`

```
def catat_penjualan(
    self, item_pesanan: PesananItem, metode_pembayaran: str
) -> None:
    """Mencatat penjualan ke lembar `DataPenjualan`."""
    komposisi_all = (
        f"Gula ({item_pesanan.komposisi.gula} takaran), "
        f"Susu ({item_pesanan.komposisi.susu} takaran), "
        f"Krimer ({item_pesanan.komposisi.krimer} takaran), "
        f"Cokelat ({item_pesanan.komposisi.cokelat} takaran)"
    )
    self.penjualan.append_row(
        [
            item_pesanan.kopi.nama,
            item_pesanan.suhu.capitalize(),
            komposisi_all,
            f"x{item_pesanan.jumlah}",
            item_pesanan.kopi.harga * item_pesanan.jumlah,
            metode_pembayaran,
        ]
    )
```

Fungsi ini bertujuan untuk mencatat transaksi penjualan ke *worksheet* "DataPenjualan". Nantinya, data disini akan dianalisis untuk memunculkan rekomendasi *best seller*.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `item_pesanan` – Objek `PesananItem` yang berisi informasi pesanan..

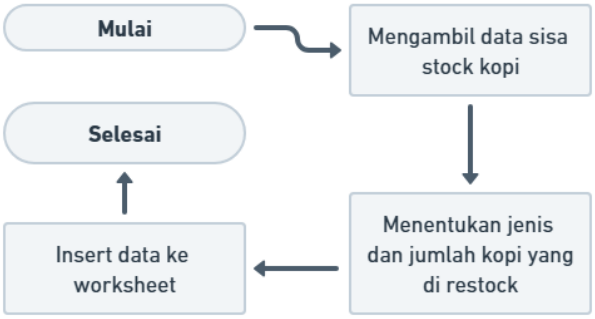
- `metode_pembayaran` – Metode pembayaran yang digunakan (misalnya: tunai atau QRIS).
- `komposisi_all` – String yang merepresentasikan komposisi semua bahan tambahan dari pesanan.

5. Fungsi `restock_kopi`

```python
def restock_kopi(self, kopi: KopiData, jumlah_restock: int) -> None:
    """Melakukan restock kopi di lembar `PersediaanKopi`."""
    stok_baru = kopi.sisa + jumlah_restock
    self.persediaan.update_cell(kopi.row_number, 3, stok_baru)
    kopi.sisa = stok_baru
```

Fungsi ini bertujuan untuk melakukan simulasi *restock* kopi, yaitu dengan menambah stok kopi di *worksheet* "PersediaanKopi".

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `kopi` – Objek `KopiData` yang ingin diperbarui stoknya.
  - `jumlah_restock` – Jumlah kopi tertentu yang akan di-*restock*.
  - `stock_baru` – Stok baru setelah dilakukan *restock*.

e. Fungsi Input dengan Timeout

```
def input_dengan_timeout(
    teks: str, batas_waktu: int = Config.TIMEOUT_DURATION
) -> str:
    """Fungsi untuk mengambil input dari pengguna dengan batas waktu."""
    try:
        return inputimeout(prompt=teks, timeout=batas_waktu)
    except TimeoutOccurred:
        print(
            "\n⌛ - Waktu habis! Tidak ada aktivitas selama 1 menit. Mesin
             kopi otomatis berhenti.\n\n"
        )
        mesin_kopi = MesinKopi()
        mesin_kopi.coffee_machine_simulasi()
```

Fungsi ini digunakan untuk menunggu input dari pengguna dengan batas waktu tertentu. Fungsi ini menampilkan prompt yang ditentukan dan menunggu input dari pengguna hingga waktu yang ditentukan. Jika pengguna tidak memberikan input dalam waktu yang ditetapkan, fungsi akan mengalami kondisi *timeout* dengan menampilkan pesan peringatan dan secara otomatis kembali ke menu utama.

- Variabel dalam fungsi
  - `teks` – Parameter yang berisi teks prompt yang akan ditampilkan kepada pengguna untuk meminta input.
  - `batas_waktu` – Parameter opsional yang menentukan durasi waktu maksimum (dalam detik) untuk menunggu input dari pengguna.
  - `mesin_kopi` – Variabel yang menjalankan inisialisasi ulang kelas `MesinKopi` sehingga setelah *timeout* akan kembali ke menu utama.

f. Manajer untuk Menampilkan Menu Kopi

```
class MenuManager:
    """Kelas untuk mengelola tampilan dan pemilihan menu kopi."""

    def __init__(self, daftar_kopi: Dict[str, KopiData]):
        self.daftar_kopi = daftar_kopi

    ...
```

Kelas `MenuManager` bertugas untuk mengelola tampilan menu kopi yang tersedia. Kelas ini berfungsi untuk menampilkan daftar kopi yang dapat dipilih oleh pengguna, serta memberikan nomor pada setiap kopi yang tersedia berdasarkan persediaan yang ada.

Properti dalam Kelas

1. `daftar_kopi` – Properti ini menyimpan data kopi yang tersedia dalam bentuk *dictionary*, dengan nama kopi sebagai kunci dan objek `KopiData` sebagai nilai.

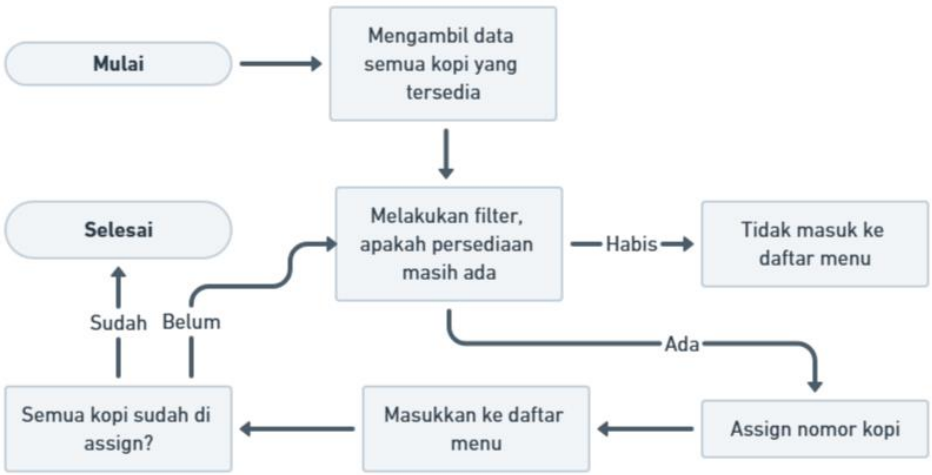
Fungsi Fungsi dalam Kelas

1. **Fungsi** `assign_kopi_numbers`

```python
def assign_kopi_numbers(self) -> None:
    """Memberikan nomor pada setiap kopi yang tersedia."""
    nomor = 1
    for kopi in self.daftar_kopi.values():
        if kopi.sisa > 0:
            kopi.nomor = nomor
            nomor += 1
```

Fungsi ini memberikan nomor urut pada setiap kopi yang tersedia dalam daftar kopi. Nomor hanya diberikan pada kopi yang masih ada persediaannya. Nomor ini kemudian dapat digunakan untuk menampilkan pilihan menu dengan urutan yang jelas bagi pengguna.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `nomor` – Variabel penghitung yang dimulai dari 1, digunakan untuk memberikan nomor urut pada setiap kopi yang tersedia. Setiap kopi yang masih ada persediaan akan diberikan nomor urut bertambah.

2. **Fungsi** `tampilkan_menu_kopi`

```python
def tampilkan_menu_kopi(self) -> None:
    """Menampilkan menu kopi yang tersedia."""
    print("================ Menu Kopi =================")
    for kopi in self.daftar_kopi.values():
        if kopi.sisa > 0:
            print(
                f"{kopi.nomor}. {kopi.nama} - Rp{kopi.harga} - Persediaan:
                  {kopi.sisa}"
            )
    print("============================================")
```

Fungsi ini digunakan untuk menampilkan menu kopi yang tersedia. Fungsi ini akan menampilkan nama kopi, harga, dan persediaan dari setiap kopi yang masih ada persediaannya. Setiap kopi yang tersedia akan ditampilkan dengan nomor urut yang sudah diberikan sebelumnya.

- Flowchart dari fungsi

- Variabel dalam fungsi

  Tidak ada variabel khusu dalam fungsi ini, karena data yang diambil merupakan data dari induk kelasnya, yaitu `self.daftar_kopi`.

g. Manajer untuk Menangani Pemesanan Kopi

```
class PesananManager:
    """Kelas untuk mengelola proses pemesanan kopi."""

    def __init__(self, daftar_kopi: Dict[str, KopiData], menu_manager:
    MenuManager):
        self.daftar_kopi = daftar_kopi
        self.menu_manager = menu_manager

    ...
```

Kelas **`PesananManager`** bertanggung jawab untuk mengelola proses pemesanan kopi, termasuk pemilihan kopi, pengaturan suhu, komposisi bahan tambahan, jumlah kopi yang dipesan, dan pengelolaan daftar pesanan. Kelas ini juga memiliki fitur untuk menampilkan ringkasan pesanan dan menghitung total harga.

Properti dalam Kelas

1. **`daftar_kopi`** – Properti ini menyimpan data kopi yang tersedia dalam bentuk *dictionary*, dengan nama kopi sebagai kunci dan objek `KopiData` sebagai nilai.
2. **`menu_manager`** – Objek dari kelas `MenuManager` yang digunakan untuk menampilkan menu kopi dan mengelola nomor urut kopi yang tersedia.

Fungsi Fungsi dalam Kelas

1. **Fungsi `pilih_suhu`**

```
def pilih_suhu(self, nama_kopi: str) -> Optional[str]:
    """Memilih suhu kopi (hangat atau dingin)."""
    while True:
        suhu_input = input_dengan_timeout(
            f"🌡️ - Tentukan suhu untuk {nama_kopi}? (1. Hangat | 2. Dingin
              | 'x' untuk batal): "
        )
        if suhu_input.lower() == "x":
            print("❌ - Membatalkan pemilihan suhu.")
            return None
        elif suhu_input == "1":
            return "hangat"
        elif suhu_input == "2":
            return "dingin"
```

```
        else:
            print(
                "⚠ - Input tidak valid. Silakan masukkan '1' untuk hangat
                 atau '2' untuk dingin."
            )
```

Fungsi ini digunakan untuk memilih suhu kopi yang akan dipesan (hangat atau dingin). Fungsi ini memandu pengguna dengan opsi input, meminta pengguna memilih suhu dengan memasukkan angka 1 (hangat) atau 2 (dingin). Jika pengguna mengetikkan 'x', proses pemilihan suhu dibatalkan.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `nama_kopi` – Nama kopi yang ingin dipesan oleh pengguna.
  - `suhu_input` – Input dari pengguna yang menentukan suhu kopi.

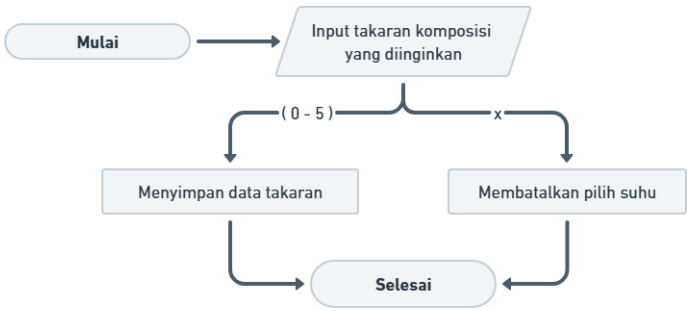
2. Fungsi `atur_jumlah`

```
def atur_jumlah(self, tipe_bahan: str) -> Optional[int]:
    """Mengatur jumlah komposisi bahan tambahan (0-5 takaran)."""
    while True:
        jumlah_input = input_dengan_timeout(
            f"> 🎨 - Atur kadar {tipe_bahan} (0-5 takaran | 'x' untuk
              batal): "
        )
        if jumlah_input.lower() == "x":
            print("❌ - Membatalkan pengaturan komposisi.")
            return None
```

```
        try:
            jumlah = int(jumlah_input)
            if 0 <= jumlah <= 5:
                return jumlah
            else:
                print("⚠ - Jumlah harus antara 0 hingga 5.")
        except ValueError:
            print("⚠ - Input tidak valid. Silakan masukkan angka.")
```

Fungsi ini digunakan untuk menentukan jumlah takaran bahan tambahan untuk kopi. Pengguna diminta memasukkan jumlah takaran dalam kisaran 0 hingga 5. Jika input tidak valid (angka di luar rentang atau bukan angka), fungsi akan meminta ulang input.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `tipe_bahan` – Jenis bahan tambahan yang ingin diatur, seperti gula, krimer, susu, atau cokelat.
  - `jumlah_input` – Input pengguna untuk jumlah takaran bahan tambahan.
  - `jumlah` – Nilai integer yang merupakan intepretasi `jumlah_input` yang telah divalidasi dari input pengguna.

3. Fungsi `pilih_komposisi`

```
def pilih_komposisi(self) -> Optional[Komposisi]:
    """Memilih komposisi bahan tambahan."""
    gula = self.atur_jumlah("gula")
    if gula is None:
        return None
    krimer = self.atur_jumlah("krimer")
```

```
    if krimer is None:
        return None
    susu = self.atur_jumlah("susu")
    if susu is None:
        return None
    cokelat = self.atur_jumlah("cokelat")
    if cokelat is None:
        return None
    return Komposisi(gula, krimer, susu, cokelat)
```

Fungsi ini meminta pengguna memilih jumlah takaran untuk setiap bahan tambahan (gula, krimer, susu, cokelat). Prosesnya menggunakan fungsi `atur_jumlah` untuk mengatur komposisi setiap bahan. Jika pengguna membatalkan pada salah satu bahan, keseluruhan proses pemilihan komposisi akan dibatalkan.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `gula` – Jumlah takaran gula.
  - `krimer` – Jumlah takaran krimer.
  - `susu` – Jumlah takaran susu.
  - `cokelat` – Jumlah takaran cokelat.

4. Fungsi `pesan_jumlah`

```
def pesan_jumlah(self) -> Optional[int]:
    """Menentukan jumlah kopi yang akan dipesan."""
    while True:
        jumlah_input = input_dengan_timeout(
            "Pesan berapa kopi dengan komposisi ini? ('x' untuk batal): "
        )
        if jumlah_input.lower() == "x":
            print("❌ - Membatalkan pemesanan.\n")
            return None
```

```
        try:
            jumlah = int(jumlah_input)
            if jumlah > 0:
                return jumlah
            else:
                print("⚠ - Jumlah kopi harus lebih dari 0.")
        except ValueError:
            print("⚠ - Input tidak valid. Silakan masukkan angka.")
```

Fungsi ini meminta pengguna untuk menentukan jumlah kopi yang ingin dipesan. Input pengguna divalidasi agar berupa angka positif. Jika input invalid atau pengguna mengetik **'x'**, proses pemesanan dibatalkan.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `jumlah_input` – Input pengguna untuk jumlah pesanan kopi.
  - `jumlah` – Nilai integer yang merupakan intepretasi `jumlah_input` yang telah divalidasi dari input pengguna.

5. Fungsi `komposisi_sama`

```
def komposisi_sama(self, komp1: Komposisi, komp2: Komposisi) -> bool:
    """Membandingkan dua komposisi bahan tambahan."""
    return komp1 == komp2
```

Fungsi ini membandingkan dua objek `Komposisi` untuk memeriksa apakah bahan tambahannya sana. Fungsi ini berguna untuk memastikan apakah pesanan yang baru dibuat memiliki komposisi yang sama dengan pesanan yang sebelumnya sudah ada.

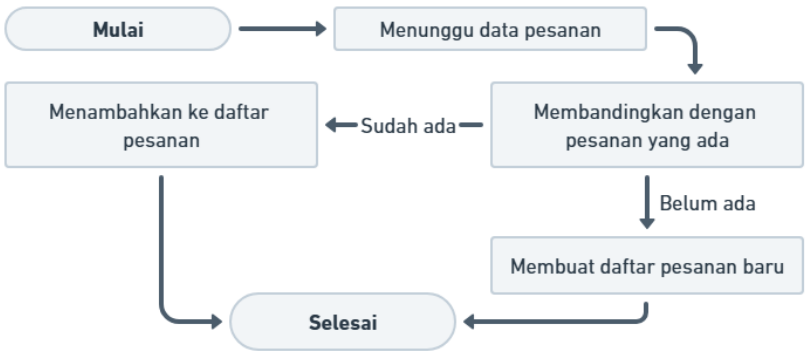
- Variabel dalam fungsi
  - `komp1` – Komposisi pesanan 1.
  - `komp2` – Komposisi pesanan 2.

6. **Fungsi** `tambah_pesanan`

```python
def tambah_pesanan(
    self,
    data_pesanan: List[PesananItem],
    kopi: KopiData,
    suhu: str,
    komposisi: Komposisi,
    jumlah: int,
) -> List[PesananItem]:
    """Menambah atau menggabungkan pesanan ke dalam daftar pesanan."""
    existing = next(
        (
            item
            for item in data_pesanan
            if item.kopi.nama == kopi.nama
            and item.suhu == suhu
            and self.komposisi_sama(item.komposisi, komposisi)
        ),
        None,
    )
    if existing:
        existing.jumlah += jumlah
    else:
        data_pesanan.append(PesananItem(kopi, jumlah, suhu, komposisi))
    return data_pesanan
```

Fungsi ini menambahkan pesanan kopi baru ke daftar pesanan yang ada. Jika pesanan dengan jenis kopi, suhu, dan komposisi yang sama sudah ada, jumlah pesanan akan ditambahkan ke item yang sudah ada. Jika belum ada, item baru akan dibuat.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `data_pesanan` – List yang berisi data seluruh pesanan pengguna.

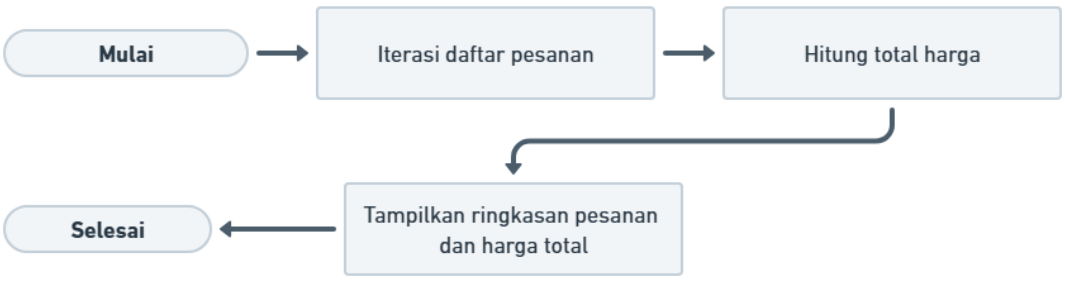
- `kopi` – Objek `KopiData` untuk jenis kopi yang dipesan.
- `suhu` – Suhu kopi (hangat atau dingin).
- `komposisi` – Komposisi bahan tambahan.
- `jumlah` – Jumlah kopi yang dipesan.

7. **Fungsi** `ringkasan_pesanan`

```
def ringkasan_pesanan(self, pesanan: List[PesananItem]) -> int:
    """Menampilkan ringkasan pesanan dan menghitung total harga."""
    print("========== Ringkasan Pesanan ==========")
    total_harga = 0
    for idx, item in enumerate(pesanan, 1):
        harga_kopi = item.kopi.harga * item.jumlah
        total_harga += harga_kopi
        print(
            f"{idx}. {item.kopi.nama} ({item.suhu}) x{item.jumlah} -
              Rp{harga_kopi}"
        )
        print(
            f"   Gula: {item.komposisi.gula} takaran, "
            f"Krimer: {item.komposisi.krimer} takaran, "
            f"Susu: {item.komposisi.susu} takaran, "
            f"Cokelat: {item.komposisi.cokelat} takaran"
        )
    print(f">>> Total Harga: Rp{total_harga}")
    print("=======================================")
    return total_harga
```

Fungsi ini menampilkan ringkasan pesanan, mencakup jenis kopi, suhu, jumlah, dan bahan tambahan untuk setiap item pesanan. Fungsi juga menghitung dan menampilkan total harga untuk semua pesanan.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `pesanan` – Daftar semua pesanan yang telah dibuat.
  - `total_harga` – Total harga untuk semua pesanan.

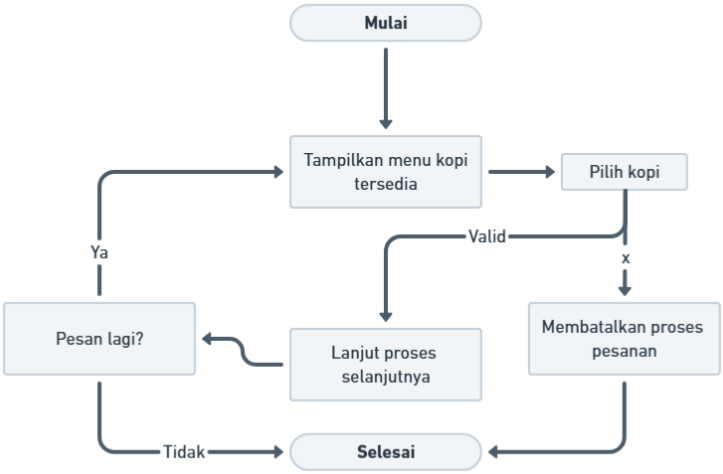
8. Fungsi `ringkasan_pesanan`

```
def pilih_kopi(self) -> List[PesananItem]:
    """Menangani proses pemilihan kopi oleh pengguna."""
    kopi_nama_by_nomor = {
        kopi.nomor: kopi
            for kopi in self.daftar_kopi.values() if kopi.sisa > 0
    }
    data_pesanan: List[PesananItem] = []
    pesanan_ke = 1
    while True:
        print(f"\n\n********** Pesanan ke-{pesanan_ke}: Pilih Kopi \
              **********")
        self.menu_manager.tampilkan_menu_kopi()
        pilihan = input_dengan_timeout(
            "Pilih nomor kopi ('x' untuk batal): "
        )
        if pilihan.lower() == "x":
            print("❌ - Membatalkan proses pemesanan.\n")
            data_pesanan = []
            break
        elif pilihan.isdigit():
            pilihan_int = int(pilihan)
            if pilihan_int in kopi_nama_by_nomor:
                kopi = kopi_nama_by_nomor[pilihan_int]
                print(f"\nAnda memilih {kopi.nama}".upper())
                suhu = self.pilih_suhu(kopi.nama)
                if suhu is None:
                    continue
                komposisi = self.pilih_komposisi()
                if komposisi is None:
                    continue
                jumlah = self.pesan_jumlah()
                if jumlah is None:
                    continue
                data_pesanan = self.tambah_pesanan(
                    data_pesanan, kopi, suhu, komposisi, jumlah
                )
```

```
                print("\nApakah Anda ingin memesan kopi lagi?")
                lagi = input_dengan_timeout(
                    "Ketik 'y' untuk Ya atau 'n' untuk Tidak: "
                ).lower()
                if lagi in ["n", "no", "tidak", "gak"]:
                    print("\nMelanjutkan ke proses pembayaran...")
                    break
                else:
                    pesanan_ke += 1
            else:
                print("⚠ - Pilihan tidak tersedia. Silakan pilih lagi.")
        else:
            print(
                "⚠ - Input tidak valid. Silakan masukkan angka atau
                 perintah yang benar."
            )
    return data_pesanan
```

Fungsi ini menangani keseluruhan proses pemesanan kopi, mulai dari pemilihan kopi dari menu, pengaturan suhu, pengaturan bahan tambahan, hingga jumlah kopi. Fungsi ini terus berjalan hingga pengguna memutuskan untuk tidak menambah pesanan lagi.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `kopi_nama_by_nomor` – *Dictionary* yang menghubungkan nomor menu kopi dengan objek `KopiData`.
  - `data_pesanan` – List yang berisi data seluruh pesanan pengguna.
  - `pesanan_ke` – Nomor pesanan yang sedang diproses.

- `pilihan` – Input pengguna untuk menentukan nomor kopi yang dipilih.

h. Manajer untuk Menangani Proses Pembayaran

```
class PembayaranManager:
    """Kelas untuk mengelola proses pembayaran."""

    ...
```

Kelas **`PembayaranManager`** bertanggung jawab untuk menangani seluruh proses pembayaran. Fitur yang disediakan mencakup pilihan metode pembayaran (tunai atau QRIS), penanganan pembayaran sesuai metode yang dipilih, dan validasi pembayaran untuk memastikan transaksi berhasil dilakukan.

Properti dalam Kelas

Kelas ini tidak memiliki properti instance tertentu karena sebagian besar fungsionalitasnya dijalankan melalui fungsi-fungsi yang bersifat mandiri. Namun, terdapat atribut lokal dalam setiap fungsi untuk menjalankan proses pembayaran.

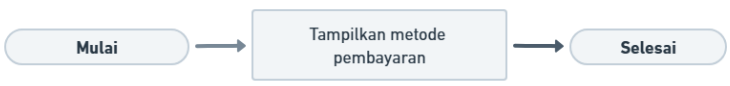
Fungsi Fungsi dalam Kelas

1. **Fungsi** `tampilkan_metode_pembayaran`

```
def tampilkan_metode_pembayaran(self) -> None:
    """Menampilkan metode pembayaran yang tersedia."""
    print("=== Metode Pembayaran Tersedia ===")
    print("1. Tunai")
    print("2. QRIS")
    print("================================")
```

Fungsi ini digunakan untuk menampilkan metode pembayaran yang tersedia kepada pengguna. Pilihan yang diberikan adalah pembayaran tunai dan pembayaran QRIS.

- Flowchart dari fungsi



- Variabel dalam fungsi

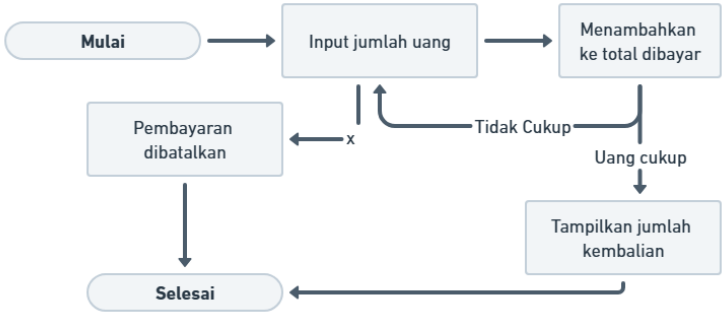
  Tidak ada variabel lokal karena fungsi ini hanya bertugas untuk menampilkan informasi tentang metode pembayaran yang tersedia.

2. **Fungsi** `tampilkan_metode_pembayaran`

```
def proses_pembayaran_tunai(self, total_harga: int) -> Tuple[bool, str]:
    """Menangani pembayaran tunai."""
    total_dibayar = 0
    while total_dibayar < total_harga:
        uang_input = input_dengan_timeout(
            f"Masukkan uang pembayaran ('x' untuk batal): Rp"
        )
        if uang_input.lower() == "x":
            print("❌ - Membatalkan pembayaran tunai.\n")
            return (False, "Tunai")
        try:
            uang = int(uang_input)
            if uang <= 0:
                print(
                    "⚠ - Jumlah uang harus lebih besar dari Rp0. Silakan
                     masukkan kembali."
                )
                continue
            total_dibayar += uang
            if total_dibayar >= total_harga:
                kembalian = total_dibayar - total_harga
                print(
                    f"\n✅ - Pembayaran berhasil. Kembalian Anda:
                       Rp{kembalian}\n"
                )
                return (True, "Tunai")
            else:
                kurang = total_harga - total_dibayar
                print(
                    f"ℹ - Uang kurang. Anda masih kurang Rp{kurang}. Silakan
                      masukkan kembali."
                )
        except ValueError:
            print("⚠ - Input tidak valid. Silakan masukkan angka.")
    return (False, "Tunai")
```

Fungsi ini menangani metode pembayaran tunai. Pengguna diminta untuk memasukkan uang secara bertahap hingga total yang dibayarkan memenuhi atau melebihi jumlah yang harus dibayar. Jika jumlah yang dibayarkan melebihi total harga, fungsi akan menghitung dan memberikan kembalian. Pengguna dapat membatalkan proses dengan mengetikkan **'x'**.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `total_harga` – Jumlah total yang harus dibayar oleh pengguna.
  - `total_dibayar` – Jumlah uang yang telah diterima dari pengguna.
  - `uang_input` – Input dari pengguna untuk nominal uang yang dimasukkan.
  - `kembalian` – Sisa uang yang harus dikembalikan kepada pengguna jika pembayaran melebihi total harga.
  - `kurang` – Jumlah uang yang masih kurang untuk melengkapi pembayaran.

3. Fungsi `generate_random_string`

```
def generate_random_string(self, length: int = 10) -> str:
    """Menghasilkan string angka acak sepanjang 'length' karakter."""
    return "".join(random.choices(string.digits, k=length))
```

Fungsi ini menghasilkan string angka acak sepanjang karakter yang ditentukan. String ini digunakan untuk membuat kode QR unik dalam proses pembayaran QRIS.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `length` – Panjang string angka acak yang dihasilkan.

4. **Fungsi** `generate_qr_terminal`

```python
def generate_qr_terminal(self, data: str) -> None:
    """Menampilkan QR Code di terminal."""
    qrcode_terminal.draw(data)
```

Fungsi ini menampilkan kode QR dalam bentuk ASCII di terminal menggunakan library `qrcode_terminal`. QR ini berisi string data yang dihasilkan oleh fungsi `generate_random_string`.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `data` – String yang akan dikonversi menjadi QR Code.

5. **Fungsi** `proses_pembayaran_qris`

```python
def proses_pembayaran_qris(self, total_harga: int) -> Tuple[bool, str]:
    """Menangani pembayaran QRIS."""
    code_to_qr = self.generate_random_string()
    print("\n=== Scan QRIS di bawah ini untuk melakukan pembayaran ===")
    self.generate_qr_terminal(code_to_qr)
    print(
        "\nMasukkan kode konfirmasi yang Anda terima setelah melakukan
         pembayaran.\n"
    )
    attempts = 5
    for _ in range(1, attempts + 1):
        code_from_user = input_dengan_timeout(
            f"Masukkan kode konfirmasi ('x' untuk batal): "
        )
        if code_from_user.lower() == "x":
            print("❌ - Membatalkan pembayaran QRIS.\n")
            return (False, "QRIS")
        if code_to_qr == code_from_user:
            print("\n✅ - Pembayaran berhasil.\n")
            return (True, "QRIS")
        else:
            print("⚠ - Kode konfirmasi salah, silakan coba lagi.")
    print("⚠ - Kesempatan memasukkan kode telah habis.")
    return (False, "QRIS")
```

Fungsi ini menangani pembayaran menggunakan metode QRIS. Fungsi menghasilkan kode QR yang perlu di-scan oleh pengguna. Setelah itu, pengguna diminta untuk memasukkan kode konfirmasi yang diterima dari aplikasi pembayaran mereka. Proses validasi berlangsung hingga 5 kali percobaan. Jika kode yang dimasukkan cocok dengan kode yang dihasilkan, pembayaran dianggap berhasil.

- Flowchart dari fungsi

- Variabel dalam fungsi
  - `total_harga` – Jumlah total yang harus dibayar oleh pengguna.
  - `code_to_qr` – String acak yang diubah menjadi QR Code.
  - `attempts` – Jumlah maksimum percobaan untuk memasukkan kode konfirmasi.
  - `code_from_user` – Kode konfirmasi yang dimasukkan oleh pengguna.

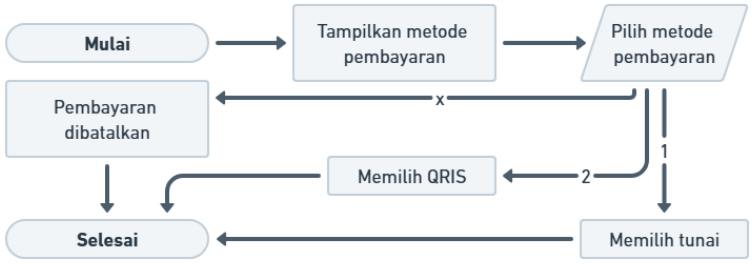
6. Fungsi `proses_pembayaran`

```
def proses_pembayaran(
    self, total_harga: int
) -> Tuple[bool, Optional[str]]:
    """Menangani proses pembayaran."""
    print("\n\n********** Pilih Metode Pembayaran **********")
    self.tampilkan_metode_pembayaran()
    print(f">>> Total yang harus dibayar: Rp{total_harga}")
    while True:
        metode_pembayaran_input = input_dengan_timeout(
            "Pilih metode pembayaran ( 1 | 2 | 'x' untuk batal): "
        )
        if metode_pembayaran_input.lower() == "x":
            print("❌ - Membatalkan proses pembayaran.\n")
            return (False, None)
```

```
        try:
            metode_pembayaran = int(metode_pembayaran_input)
            if metode_pembayaran == 1:
                return self.proses_pembayaran_tunai(total_harga)
            elif metode_pembayaran == 2:
                return self.proses_pembayaran_qris(total_harga)
            else:
                print(
                    "⚠ - Metode pembayaran tidak tersedia. Silakan pilih 1
                     atau 2."
                )
        except ValueError:
            print("⚠ - Input tidak valid. Silakan masukkan angka.")
```

Fungsi utama yang mengelola proses pembayaran. Fungsi ini menampilkan pilihan metode pembayaran kepada pengguna (tunai atau QRIS), lalu memanggil fungsi yang sesuai berdasarkan pilihan pengguna. Jika pengguna mengetikkan **'x'**, proses pembayaran dibatalkan.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `total_harga` – Jumlah total yang harus dibayar oleh pengguna.
  - `metode_pembayaran_input` – Input dari pengguna untuk memilih metode pembayaran.
  - `metode_pembayaran` – Nilai integer yang merupakan intepretasi `metode_pembayaran_input` yang telah divalidasi dari input pengguna.

i. Manajer untuk Menangani Proses Pesanan Melalui Kode QR

```
class ScanQR:
    """Kelas untuk mengelola proses pemindaian QR Code."""

    def __init__(self, db_manager: DatabaseManager, pesanan_manager:
    PesananManager):
        self.db_manager = db_manager
        self.pesanan_manager = pesanan_manager
        self.daftar_kopi = pesanan_manager.daftar_kopi
        self.menu_manager = pesanan_manager.menu_manager

    ...
```

Kelas **`ScanQR`** digunakan untuk mengelola proses pemindaian QR Code dalam sistem pemesanan. Fungsi utama kelas ini adalah memindai QR Code, memvalidasi pesanan, memperbarui status pesanan, dan mencatat transaksi.

Properti dalam Kelas

1. **`db_manager`** – Objek dari kelas `DatabaseManager` untuk mengelola koneksi dan interaksi dengan database.
2. **`pesanan_manager`** – Objek dari kelas `PesananManager` untuk mengelola data pesanan.
3. **`daftar_kopi`** – Menyimpan daftar jenis kopi yang tersedia.
4. **`menu_manager`** – Properti dari `pesanan_manager` yang mengelola menu kopi.

Fungsi Fungsi dalam Kelas

1. **Fungsi `scan_qr`**

```
    def scan_qr(self) -> None:
        """Fungsi untuk memindai QR dan mengonfirmasi pesanan dengan status
           `Pending`."""
        detector = cv2.QRCodeDetector()
        cap = cv2.VideoCapture(0)

        if not cap.isOpened():
            print("\n⚠ - Tidak dapat membuka pemindai QR.\n\n")
            return
```

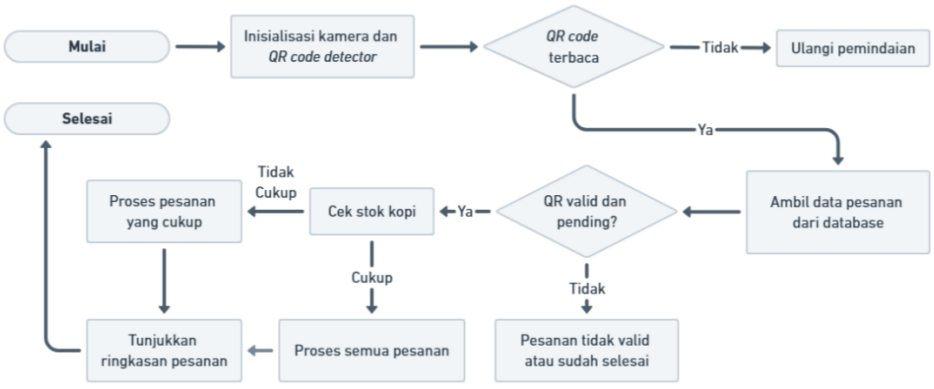
```
        print("\n\n********** Pindai QR Code ***********")
        print("Dekatkan QR Code ke alat pemindai QR")
        qr_code = ""
        try:
            while True:
                ret, frame = cap.read()
                if not ret:
                    print("⚠ - Tidak dapat membaca frame dari pemindai
                            QR.\n\n")
                    break
                data, _, _ = detector.detectAndDecode(frame)
                if data:
                    qr_code = data
                    break
                if cv2.waitKey(1) & 0xFF == ord("q"):
                    print("❌ - Pemindaian QR Code dibatalkan.\n\n")
                    break
        finally:
            cap.release()
            cv2.destroyAllWindows()
        if not qr_code:
            print("⚠ - Tidak ada QR Code yang dipindai.\n\n")
            return

        data_qr = self.db_manager.antrian_qr.get_all_records()
        antrian_qr_columns = self.db_manager.get_column_indices(
            self.db_manager.antrian_qr
        )
        persediaan_columns = self.db_manager.get_column_indices(
            self.db_manager.persediaan
        )
        self.daftar_kopi = self.db_manager.ambil_data_kopi()
        self.menu_manager.assign_kopi_numbers()

        pesanan = []
        habis = False
        valid = False
        for index, baris in enumerate(data_qr, start=2):
            if str(baris["QR"]) == qr_code and baris["Status"] == "Pending":
                kopi_nama = baris["Jenis kopi"]
                jumlah_dipesan = int(baris["Jumlah"])
                if kopi_nama not in self.daftar_kopi:
                    print(f"ℹ - Maaf, {kopi_nama} tidak tersedia di mesin
                            ini")
                    continue
```

```
                kopi = self.daftar_kopi[kopi_nama]
                stok_saat_ini = kopi.sisa

                if stok_saat_ini <= 0:
                    habis = True
                    print(f"ℹ - Stok {kopi_nama} habis, silahkan coba di
                            mesin lain")
                    continue
                if stok_saat_ini < jumlah_dipesan:
                    print(f"ℹ - Stok {kopi_nama} tidak mencukupi. Tersisa
                            {stok_saat_ini} cup.")
                    jumlah_dapat_diproses = stok_saat_ini
                else:
                    jumlah_dapat_diproses = jumlah_dipesan

                komposisi = Komposisi(
                    gula=int(baris.get("Gula", 0)),
                    krimer=int(baris.get("Krimer", 0)),
                    susu=int(baris.get("Susu", 0)),
                    cokelat=int(baris.get("Cokelat", 0)),
                )
                suhu = baris["Suhu"].lower()

                # Menambah pesanan ke daftar
                self.pesanan_manager.tambah_pesanan(
                    pesanan, kopi, suhu, komposisi, jumlah_dapat_diproses
                )

                # Memperbarui status pesanan di Google Sheets
                if jumlah_dapat_diproses == jumlah_dipesan:
                    self.db_manager.antrian_qr.update_cell(
                        index, antrian_qr_columns["Status"], "Selesai"
                    )
                    self.db_manager.antrian_qr.update_cell(
                        index, antrian_qr_columns["Jumlah"], 0
                    )
                else:
                    sisa_pesanan = jumlah_dipesan - jumlah_dapat_diproses
                    self.db_manager.antrian_qr.update_cell(
                        index, antrian_qr_columns["Jumlah"], sisa_pesanan
                    )
```

```
                stok_baru = kopi.sisa - jumlah_dapat_diproses
                self.db_manager.persediaan.update_cell(
                    kopi.row_number,
                    persediaan_columns["Sisa Persediaan"],
                    stok_baru
                )
                kopi.sisa = stok_baru

            elif str(baris["QR"]) == qr_code and baris["Status"] == "Selesai":
                valid = True

        if pesanan:
            print(" ")
            self.pesanan_manager.ringkasan_pesanan(pesanan)
            print("\n☕ - Terima kasih! Silakan ambil kopi Anda.\n\n")
            for item in pesanan:
                self.db_manager.catat_penjualan(item, "Pembelian Daring
                Website")
        else:
            if not habis:
                if valid:
                    print("ℹ - Semua pesanan dengan QR ini sudah
                          selesai.\n\n")
                else:
                    print("⚠ - QR yang diberikan tidak valid.\n\n")
```

Fungsi ini digunakan untuk memindai QR Code dengan kamera, membaca informasi pesanan dari database, memvalidasi stok yang tersedia, dan memperbarui status pesanan. Jika valid, pesanan akan diproses dan data penjualan dicatat.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `detector` – Objek dari `cv2.QRCodeDetector()` untuk mendeteksi QR Code.
  - `cap` – Objek dari `cv2.VideoCapture(0)` untuk membuka kamera.

- `qr_code` – Menyimpan hasil data QR Code yang dipindai.
- `data_qr` – Data pesanan dari database (Google Sheets).
- `antrian_qr_columns` – Indeks kolom dari database `antrian_qr`.
- `persediaan_columns` – Indeks kolom dari database persediaan.
- `pesanan` – List pesanan yang ada didalam database.
- `habis` – Boolean untuk status stok habis.
- `valid` – Boolean untuk status QR Code valid tetapi selesai.

j. Manajer untuk Menangani Proses Restock Kopi

```
class RestockKopi:
    """Kelas untuk mengelola proses restock kopi."""

    def __init__(self, db_manager: DatabaseManager, menu_manager:
    MenuManager):
        self.db_manager = db_manager
        self.menu_manager = menu_manager
        self.daftar_kopi = menu_manager.daftar_kopi

    ...
```

Kelas **`RestockKopi`** digunakan untuk mengelola proses restock kopi yang hanya dapat diakses oleh admin. Kelas ini menyediakan mekanisme autentikasi admin, memilih kopi yang akan direstock, dan memperbarui stok di database.

Properti dalam Kelas

1. `db_manager` – Objek dari kelas `DatabaseManager` untuk mengelola koneksi dan interaksi dengan database.
2. `menu_manager` – Properti dari `pesanan_manager` yang mengelola menu kopi.
3. `daftar_kopi` – Menyimpan daftar jenis kopi yang tersedia.

Fungsi Fungsi dalam Kelas

1. Fungsi `restock_kopi`

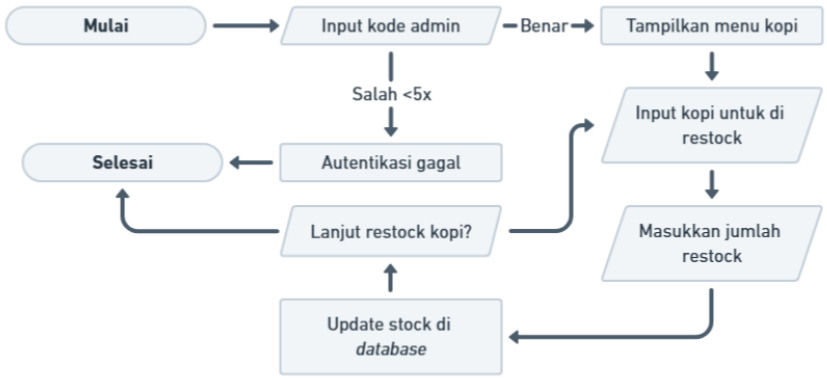
```
def restock_kopi(self) -> None:
    """Fitur khusus admin untuk restock kopi."""
    print("\n\n********** Menu Restock Kopi ***********")
    max_attempts = 5
    attempts = 0
    while attempts < max_attempts:
        kode_admin_input = input_dengan_timeout(
            "Masukkan kode admin ('x' untuk batal): "
        )
        if kode_admin_input.lower() == "x":
            print("❌ - Membatalkan restock kopi.\n\n")
            return
        try:
            kode_admin = int(kode_admin_input)
        except ValueError:
            print("⚠ - Kode admin harus berupa angka.")
            attempts += 1
            continue
        if kode_admin == Config.KODE_ADMIN:
            while True:
                self.menu_manager.assign_kopi_numbers()
                self.menu_manager.tampilkan_menu_kopi()
                kopi_nama_by_nomor = {
                    kopi.nomor: kopi
                    for kopi in self.daftar_kopi.values()
                    if kopi.sisa >= 0
                }

                # Loop untuk memilih kopi yang akan di restock
                while True:
                    pilihan_kopi_input = input_dengan_timeout(
                        "Pilih kopi untuk restock ('x' untuk batal): "
                    )
                    if pilihan_kopi_input.lower() == "x":
                        print("❌ - Membatalkan restock kopi.\n\n")
                        return
                    if not pilihan_kopi_input.isdigit():
                        print(
                            "⚠ - Input tidak valid. Silakan masukkan nomor
                             kopi atau 'x' untuk batal."
                        )
                        Continue
```

```
            pilihan_kopi = int(pilihan_kopi_input)
            if pilihan_kopi not in kopi_nama_by_nomor:
                print(
                    "⚠ - Pilihan kopi tidak valid. Silakan pilih
                     nomor yang tersedia."
                )
                continue
            kopi = kopi_nama_by_nomor[pilihan_kopi]
            break  # Keluar loop setelah memilih kopi yang valid

        # Loop untuk memasukkan jumlah restock
        while True:
            jumlah_restock_input = input_dengan_timeout(
                "Masukkan jumlah restock ('x' untuk batal): "
            )
            if jumlah_restock_input.lower() == "x":
                print("❌ - Membatalkan restock kopi.\n\n")
                return
            try:
                jumlah_restock = int(jumlah_restock_input)
                if jumlah_restock <= 0:
                    print("⚠ - Jumlah restock harus lebih dari 0.")
                    continue
                self.db_manager.restock_kopi(kopi, jumlah_restock)
                print(
                    f"✅ - Berhasil restock {kopi.nama} sebanyak
                     {jumlah_restock}. Stok baru: {kopi.sisa}.\n\n"
                )
                break  # Keluar dari loop setelah sukses restock
            except ValueError:
                print("⚠ - Masukkan harus berupa angka.")

        # Pertanyaan apakah ingin melakukan restock lagi
        while True:
            lagi_input = input_dengan_timeout(
                "\nApakah ingin melakukan restock lagi? ('y' untuk
                 Ya | 'n' untuk Tidak): "
            ).lower()
            if lagi_input in ["y", "ya"]:
                break  # Kembali ke awal loop restock
            elif lagi_input in ["n", "tidak"]:
                print("\n🔃 - Kembali ke menu utama.\n\n")
                return
```

```
                else:
                    print(
                        "⚠ - Input tidak valid. Silakan masukkan 'y'
                         atau 'n'."
                    )
            return
        else:
            attempts += 1
            print("⚠ - Kode admin salah! Coba lagi.")
    print("⚠ - Autentikasi administrator gagal.")
```

Fungsi ini adalah fitur khusus untuk admin yang memungkinkan pengguna dengan autentikasi yang valid untuk menambahkan stok kopi ke database. Fungsi ini mencakup beberapa tahap, termasuk autentikasi admin, memilih jenis kopi untuk di-restock, memasukkan jumlah stok baru, dan memperbarui data di database.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `max_attempts` – Jumlah maksimum percobaan untuk memasukkan kode admin.
  - `kode_admin_input` – Input yang dimasukkan oleh pengguna sebagai kode admin.
  - `kode_admin` – Nilai integer yang merupakan intepretasi `kode_admin_input` yang telah divalidasi dari input pengguna.
  - `kopi_nama_by_nomor` – Kamus yang memetakan nomor kopi ke objek kopi untuk mempermudah pemilihan kopi.
  - `pilihan_kopi` – Nomor kopi yang dipilih oleh pengguna.
  - `kopi` – Objek kopi yang dipilih berdasarkan nomor kopi.

- `jumlah_restock_input` – Input jumlah stok yang ingin ditambahkan.
- `jumlah_restock` – Nilai integer yang merupakan intepretasi `jumlah_restock_input` yang telah divalidasi dari input pengguna.
- `lagi_input` – Input untuk menentukan apakah ingin melakukan restock lagi atau tidak.

k. Program Utama untuk Menangani Mesin Kopi

```
class MesinKopi:
    """Kelas utama untuk mengontrol operasi mesin kopi."""

    def __init__(self):
        """Inisialisasi mesin kopi dan memuat data awal."""
        self.db_manager = DatabaseManager()
        self.daftar_kopi: Dict[str, KopiData] =
            self.db_manager.ambil_data_kopi()
        self.menu_manager = MenuManager(self.daftar_kopi)
        self.pesanan_manager = PesananManager(
            self.daftar_kopi,
            self.menu_manager
        )
        self.pembayaran_manager = PembayaranManager()
        self.scan_qr_manager = ScanQR(
            self.db_manager,
            self.pesanan_manager
        )
        self.restock_manager = RestockKopi(
            self.db_manager,
            self.menu_manager
        )
        self.menu_manager.assign_kopi_numbers()

  ...
```

Kelas **`MesinKopi`** merupakan kelas utama yang bertanggung jawab untuk mengelola seluruh alur kerja Mesin Kopi Virtual, mulai dari pemesanan, pembayaran, scan QR, restock kopi, hingga penonaktifan program. Kelas ini memanfaatkan beberapa manajer pendukung untuk menangani tugas-tugas spesifik.

Properti dalam Kelas

1. `db_manager` – Objek dari kelas `DatabaseManager` untuk mengelola koneksi dan interaksi dengan database.
2. `daftar_kopi` – Menyimpan daftar jenis kopi yang tersedia.
3. `menu_manager` – Properti dari `pesanan_manager` yang mengelola menu kopi.
4. `pesanan_manager` – Objek dari kelas `PesananManager` untuk mengelola data pesanan.
5. `pembayaran_manager` – Objek dari `PembayaranManager` untuk menangani pembayaran.
6. `scan_qr_manager` – Objek dari `ScanQR` untuk membaca QR code pesanan.
7. `restock_manager` – Objek dari `RestockKopi` untuk menangani restock kopi oleh admin.

Fungsi Fungsi dalam Kelas

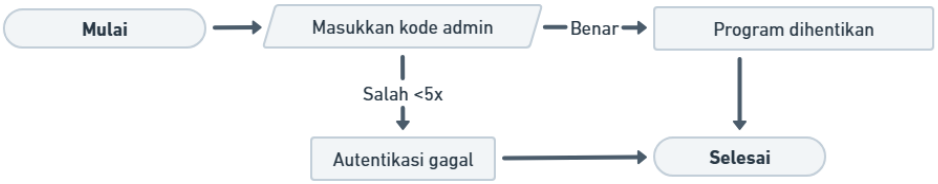
1. **Fungsi** `shutdown_program`

```
def shutdown_program(self):
    """Fungsi untuk mematikan program setelah otentikasi admin dengan
    maksimal 5 percobaan."""
    print("\n\n********** Menonaktifkan Program ***********")
    max_attempts = 5
    attempts = 0
    while attempts < max_attempts:
        kode_admin_input = input_dengan_timeout(
            "Masukkan kode admin ('x' untuk batal): "
        )
        if kode_admin_input.lower() == "x":
            print("❌ - Membatalkan penonaktifan program.\n\n")
            return
        try:
            kode_admin = int(kode_admin_input)
        except ValueError:
            print("⚠ - Kode admin harus berupa angka.")
            attempts += 1
            continue
```

```
        if kode_admin == Config.KODE_ADMIN:
            print("🛑 - Algoritma dimatikan. Program akan keluar.")
            sys.exit(0)
        else:
            attempts += 1
            print(
                f"⚠ - Kode admin salah! {max_attempts - attempts} percobaan
                  tersisa."
            )
    print("⚠ - Autentikasi administrator gagal. Kembali ke menu utama.")
```

Fungsi untuk menonaktifkan program dengan autentikasi admin. Admin diberikan maksimal 5 percobaan untuk memasukkan kode admin yang valid.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `max_attempts` – Jumlah maksimum percobaan untuk memasukkan kode admin.
  - `kode_admin_input` – Input yang dimasukkan oleh pengguna sebagai kode admin.
  - `kode_admin` – Nilai integer yang merupakan intepretasi `kode_admin_input` yang telah divalidasi dari input pengguna.

2. Fungsi `coffee_machine_simulasi`

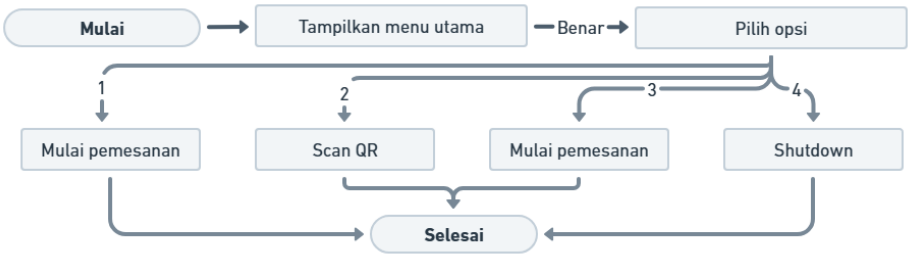
```
def coffee_machine_simulasi(self) -> None:
    """Fungsi utama yang menangani seluruh proses Mesin Kopi Virtual."""
    print("\n=== Selamat datang di Mesin Kopi Virtual! ===\n")
    while True:
        print("======= Pilihan Menu =======")
        print("1. Mulai Pemesanan")
        print("2. Scan QR")
        print("3. Restock Kopi (Admin)")
        print("4. Shutdown Program (Admin)")
        print("============================")
```

```
        pilihan = input("Pilih opsi (1, 2, 3, 4): ")
        if pilihan == "1":
            if not self.daftar_kopi:
                print(
                    "\n💔 - Maaf, semua kopi telah habis. Silakan kembali
                     lain waktu.\n\n"
                )
                continue
            pesanan = self.pesanan_manager.pilih_kopi()
            if pesanan:
                stok_cukup = True
                for item in pesanan:
                    if item.kopi.sisa < item.jumlah:
                        print(
                            f"☕ - Stok {item.kopi.nama} tidak mencukupi.
                              Tersisa {item.kopi.sisa}."
                        )
                        stok_cukup = False
                if not stok_cukup:
                    print(
                        "🔃 - Silakan ulangi pemesanan dengan jumlah yang
                         tersedia.\n\n"
                    )
                    continue
                print(" \n ")
                total_harga = self
                    .pesanan_manager.ringkasan_pesanan(pesanan, 2)
                (
                    pembayaran_sukses,
                    metode,
                ) = self.pembayaran_manager.proses_pembayaran(total_harga)
                if pembayaran_sukses:
                    print("☕ - Terima kasih! Silakan ambil kopi
                           Anda.\n\n")
                    for item in pesanan:
                        self.db_manager.catat_penjualan(item, metode)
                        self.db_manager.update_stok(item.kopi, item.jumlah)
                    self.daftar_kopi = self.db_manager.ambil_data_kopi()
                    self.menu_manager.assign_kopi_numbers()
                else:
                    if metode is None:
                        print(
                            "💰 - Pembayaran dibatalkan. Kembali ke menu
                             utama.\n\n"
                        )
                    else:
                        print("📉 - Pembayaran gagal. Kembali ke menu
                               utama.\n\n")
```

```
            else:
                print(
                    "📄 - Pemesanan dibatalkan atau tidak ada pesanan.
                    Kembali ke menu utama.\n\n"
                )
        elif pilihan == "2":
            self.scan_qr_manager.scan_qr()
        elif pilihan == "3":
            self.restock_manager.restock_kopi()
            self.daftar_kopi = self.db_manager.ambil_data_kopi()
            self.menu_manager.assign_kopi_numbers()
        elif pilihan == "4":
            self.shutdown_program()
        else:
            print("⚠ - Pilihan tidak valid. Silakan pilih 1, 2, 3, atau 4")
```

Fungsi utama yang mengatur seluruh proses interaksi mesin kopi, termasuk pemesanan, pembayaran, scan QR, restock kopi, dan shutdown program.

- Flowchart dari fungsi



- Variabel dalam fungsi
  - `pilihan` – Input pilihan menu dari pengguna.
  - `pesanan` – Objek pesanan yang dihasilkan dari `PesananManager`.
  - `stock_cukup` – Boolean untuk mengecek apakah stok cukup untuk memenuhi pesanan.
  - `total_harga` – Total harga pesanan yang dihitung oleh `PesananManager`.
  - `pembayaran_sukses` – Boolean yang menunjukkan apakah pembayaran berhasil.
  - `metode` – Metode pembayaran yang dipilih oleh pengguna.

5. Catatan Tambahan untuk Program *Coffee Machine*

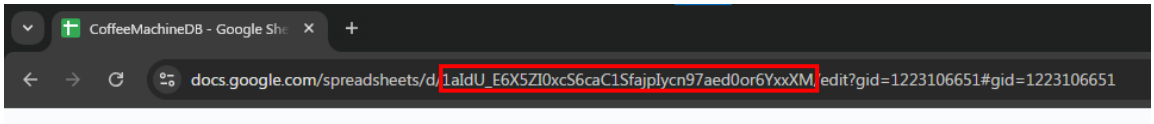
Beberapa catatan tambahan yang perlu ditambahkan agar mendukung berjalannya program *coffee machine* ini dijelaskan dibawah ini.

1. Apabila ingin melihat kamus dari semua parameter atau variabel yang ada pada program, dapat mengakses pranala berikut github.com.

2. Untuk melihat semua flowchart yang tertera pada bagian sebelumnya, dapat diakses melalui whimsical.com.

3. Program *coffee machine* ini dibuat dengan memanfaatkan Google Spreadsheet sebagai database. Oleh karena itu, untuk menjalankan file ini diperlukan layanan API Google Spreadsheet yang disediakan oleh google (yang nanti file *key* akan menjadi dasar autentikasi). Untuk petunjuk penggunaan layanan API Google Spreadsheet dapat ditemukan pada pranala berikut datalab-docs.datacamp.com.

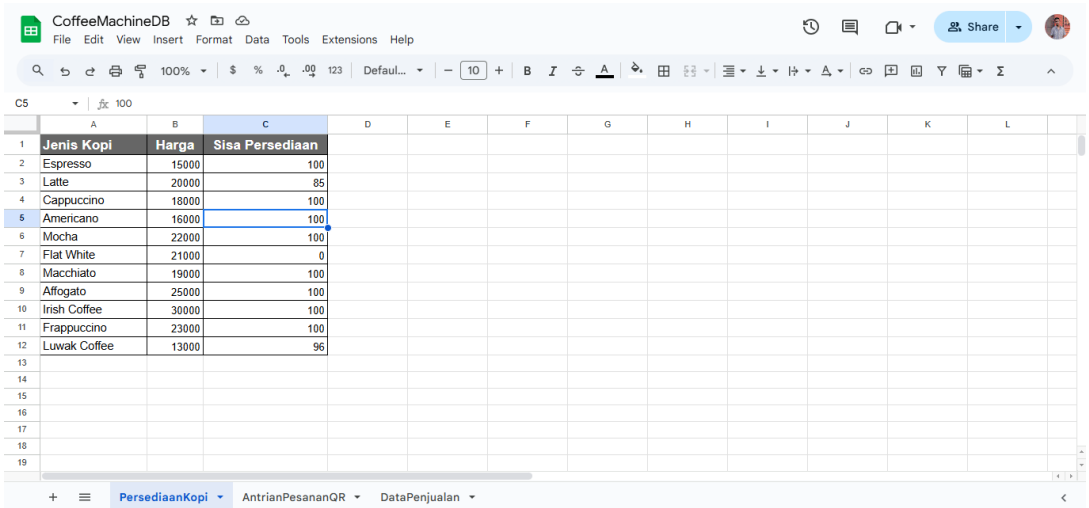
4. Untuk mendapatkan Google Sheet ID, berikut adalah rinciannya.



Dari contoh pranala yang sudah diberikan tersebut, Google Sheet ID nya adalah **1aIdU_E6X5ZI0xcS6caC1SfajpIycn97aed0or6YxxXM.**

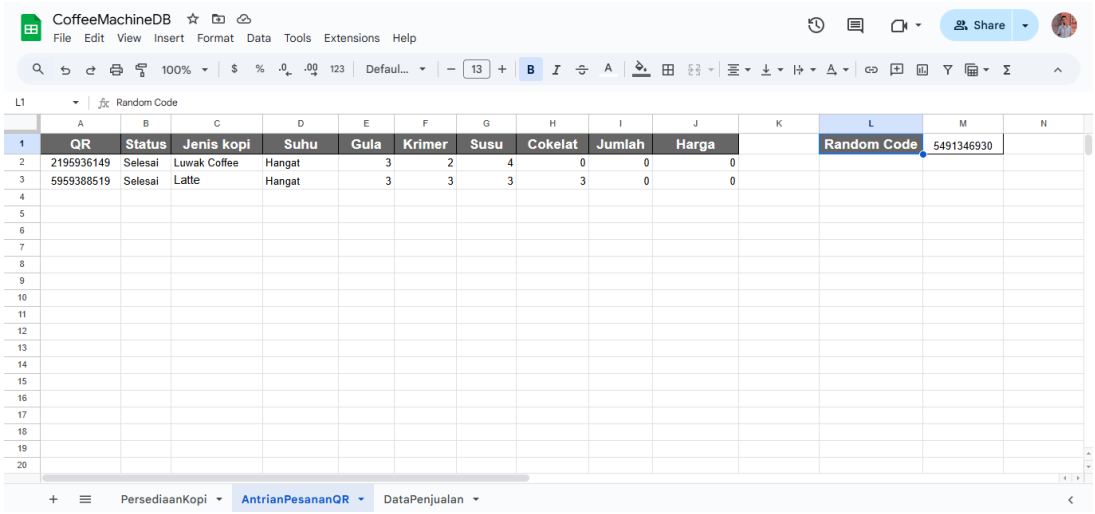
5. Untuk isi dari Google Spreadsheet, terdiri dari tiga Worksheet, dengan rincian sebagai berikut. Sedikit catatan bahwa untuk warna dan besar teks di header tidak perlu sama, tetapi teks yang diberikan harus sama dengan yang tertera pada gambar.

   - Worksheet **PersediaanKopi**



	A	B	C
1	Jenis Kopi	Harga	Sisa Persediaan
2	Espresso	15000	100
3	Latte	20000	85
4	Cappuccino	18000	100
5	Americano	16000	100
6	Mocha	22000	100
7	Flat White	21000	0
8	Macchiato	19000	100
9	Affogato	25000	100
10	Irish Coffee	30000	100
11	Frappuccino	23000	100
12	Luwak Coffee	13000	96

- Worksheet **AntrianPesananQR**



CoffeeMachineDB

File Edit View Insert Format Data Tools Extensions Help

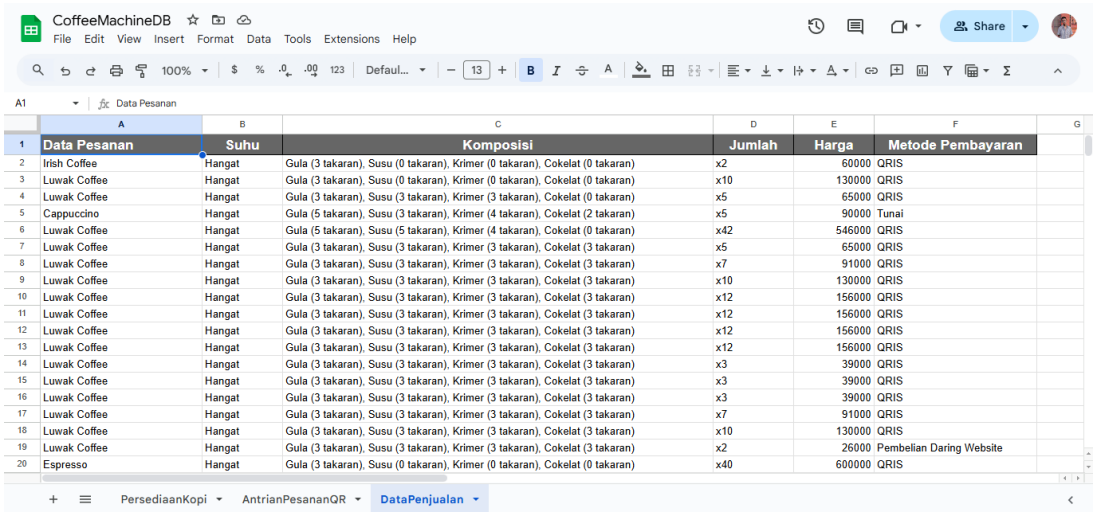
Share

L1 | Random Code

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	QR	Status	Jenis kopi	Suhu	Gula	Krimer	Susu	Cokelat	Jumlah	Harga		Random Code	5491346930	
2	2195936149	Selesai	Luwak Coffee	Hangat	3	2	4	0	0	0				
3	5959388519	Selesai	Latte	Hangat	3	3	3	3	0	0				

PersediaanKopi | AntrianPesananQR | DataPenjualan

- Worksheet **DataPenjualan**



CoffeeMachineDB

File Edit View Insert Format Data Tools Extensions Help

Share

A1 | Data Pesanan

	A	B	C	D	E	F	G
1	Data Pesanan	Suhu	Komposisi	Jumlah	Harga	Metode Pembayaran	
2	Irish Coffee	Hangat	Gula (3 takaran), Susu (0 takaran), Krimer (0 takaran), Cokelat (0 takaran)	x2	60000	QRIS	
3	Luwak Coffee	Hangat	Gula (3 takaran), Susu (0 takaran), Krimer (0 takaran), Cokelat (0 takaran)	x10	130000	QRIS	
4	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (0 takaran)	x5	65000	QRIS	
5	Cappuccino	Hangat	Gula (5 takaran), Susu (3 takaran), Krimer (4 takaran), Cokelat (2 takaran)	x5	90000	Tunai	
6	Luwak Coffee	Hangat	Gula (5 takaran), Susu (5 takaran), Krimer (4 takaran), Cokelat (0 takaran)	x42	546000	QRIS	
7	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x5	65000	QRIS	
8	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x7	91000	QRIS	
9	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x10	130000	QRIS	
10	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x12	156000	QRIS	
11	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x12	156000	QRIS	
12	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x12	156000	QRIS	
13	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x12	156000	QRIS	
14	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x3	39000	QRIS	
15	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x3	39000	QRIS	
16	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x3	39000	QRIS	
17	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x7	91000	QRIS	
18	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x10	130000	QRIS	
19	Luwak Coffee	Hangat	Gula (3 takaran), Susu (3 takaran), Krimer (3 takaran), Cokelat (3 takaran)	x2	26000	Pembelian Daring Website	
20	Espresso	Hangat	Gula (3 takaran), Susu (0 takaran), Krimer (0 takaran), Cokelat (0 takaran)	x40	600000	QRIS	

PersediaanKopi | AntrianPesananQR | DataPenjualan

6. Luaran yang Dihasilkan dari Tugas Ini

- Laporan berupa file pdf yang berisi penjelasan spesifikasi program, dekomposisi, abstraksi, pengenalan pola, dan algoritma dari program *coffee machine* ini (file ini).
- File yang dijadikan dasar untuk melaksanakan presentasi, file ini berisi hal yang sama dengan yang ada di dalam laporan ini, hanya saja lebih singkat. File tersebut bisa diakses melalui Canva.
- Video hasil presentasi, yang berisi penjelasan tentang program *coffee machine* yang kami buat beserta dengan analisis 4 aspek utama berpikir komputasional. Video tersebut bisa diakses melalui Google Drive.
- Source Code yang bisa diakses dalam file **.zip** yang sudah disediakan, atau dapat diakses melalui Github dan Google Colaboratory.