

**LAPORAN UAS PEMROGRAMAN BERORIENTASI OBJEK (PBO)**  
**SISTEM MANAJEMEN KEBUTUHAN AIR BERSIH SAAT KEKERINGAN**



Disusun Oleh : Kelompok 2

2411102441241 Muhammad Fahbian Alzi Annur

2411102441250 Malik Sabarullah Akbar

2411102441164 Andi Reza

2411102441049 Rafa Haris

2411102441191 Muhammad Ishaq

2411102441162 Alan Yahya

2411102441046 Fahriandy Adithia

**FAKULTAS SAINS DAN TEKNOLOGI**  
**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR**

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Laporan ini disusun sebagai bagian dari pengerjaan Proyek Ujian Akhir Semester (UAS) pada mata kuliah Pemrograman Berorientasi Objek (PBO). Proyek UAS ini mengangkat studi kasus *Manajemen Kebutuhan Air Bersih Saat Kekeringan*, yang bertujuan untuk menerapkan konsep-konsep dasar Pemrograman Berorientasi Objek dalam penyelesaian permasalahan nyata.

Permasalahan kekurangan air bersih sering terjadi saat musim kemarau atau kondisi kekeringan, sehingga diperlukan suatu sistem yang mampu membantu proses pengelolaan, pencatatan, dan distribusi air bersih secara terstruktur dan efisien. Melalui proyek ini, mahasiswa diharapkan dapat merancang dan mengimplementasikan aplikasi berbasis Python dengan menerapkan prinsip PBO seperti enkapsulasi, pewarisan, abstraksi, serta pemisahan tanggung jawab antar kelas.

Dengan adanya laporan ini, diharapkan pembaca dapat memahami latar belakang pembuatan proyek, tujuan pengembangan sistem, serta bagaimana konsep Pemrograman Berorientasi Objek diterapkan dalam studi kasus manajemen kebutuhan air bersih saat kekeringan.

### **1.2 Tujuan Proyek**

Tujuan dari pembuatan proyek UAS Pemrograman Berorientasi Objek ini adalah untuk merancang dan mengimplementasikan sebuah sistem manajemen kebutuhan air bersih saat kekeringan dengan menerapkan konsep-konsep PBO. Selain itu, proyek ini bertujuan untuk melatih mahasiswa dalam menyusun struktur program yang terorganisir, modular, dan mudah dikembangkan menggunakan bahasa pemrograman Python.

### **1.3 Manfaat Proyek**

Manfaat dari pembuatan proyek ini adalah membantu pengguna dalam melakukan pengelolaan data kebutuhan dan distribusi air bersih secara lebih terstruktur dan efisien. Selain itu, proyek ini juga memberikan manfaat akademis bagi mahasiswa dalam memahami penerapan konsep Pemrograman Berorientasi Objek secara nyata, mulai dari perancangan kelas hingga implementasi sistem yang sesuai dengan studi kasus yang diberikan.

## **BAB II**

### **TINJAUAN TEORI**

#### **2.1 Pemrograman Berorientasi Objek (PBO)**

Pemrograman Berorientasi Objek (Object Oriented Programming/OOP) merupakan paradigma pemrograman yang menyusun program berdasarkan kelas dan objek sebagai representasi data dan perilaku. Pendekatan ini bertujuan untuk mempermudah pengelolaan kode, meningkatkan keterbacaan, serta memudahkan proses pengembangan dan pemeliharaan sistem. Dalam proyek manajemen kebutuhan air bersih saat kekeringan, PBO digunakan untuk memodelkan permasalahan nyata ke dalam objek-objek yang saling berinteraksi sehingga sistem menjadi lebih terstruktur dan mudah dikembangkan.

#### **2.2 Konsep Dasar PBO**

Konsep dasar dalam Pemrograman Berorientasi Objek meliputi enkapsulasi, pewarisan, abstraksi, dan polimorfisme. Enkapsulasi berfungsi untuk membatasi akses langsung terhadap data, pewarisan memungkinkan penggunaan kembali kode, abstraksi menyederhanakan kompleksitas sistem, dan polimorfisme memungkinkan satu metode memiliki perilaku yang berbeda. Keempat konsep ini diterapkan dalam proyek untuk membangun sistem yang modular, fleksibel, dan sesuai dengan prinsip PBO.

#### **2.3 Penerapan PBO dalam Sistem**

Penerapan Pemrograman Berorientasi Objek dalam sistem manajemen kebutuhan air bersih saat kekeringan dilakukan dengan memisahkan program ke dalam beberapa kelas sesuai dengan fungsi dan tanggung jawabnya. Setiap kelas dirancang untuk merepresentasikan bagian tertentu dari sistem, seperti pengelolaan data, logika layanan, dan proses utama program, sehingga kode menjadi lebih terstruktur, mudah dipahami, serta mudah dikembangkan dan dipelihara di kemudian hari.

## BAB III

### IMPLEMENTASI

Bab ini membahas implementasi program yang telah dikembangkan, termasuk alur kerja sistem dan keseluruhan kode utama yang digunakan. Pada bagian ini ditunjukkan bagaimana alur program dan implementasi kode disusun serta dijalankan untuk merealisasikan sistem manajemen kebutuhan air bersih saat kekeringan, mulai dari struktur file, logika program, hingga proses eksekusi utama.

#### 3.1 Bagian UML

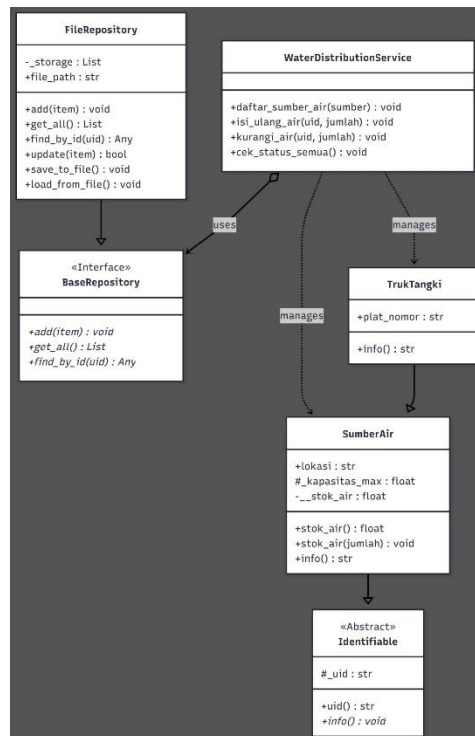


Diagram UML ini menunjukkan sistem distribusi air di mana **WaterDistributionService** berfungsi sebagai pengelola utama sumber air dan truk tangki, dengan dukungan **FileRepository** yang mengimplementasikan **BaseRepository** untuk penyimpanan dan pengambilan data. **SumberAir** mewarisi **Identifiable** agar memiliki UID unik serta menyimpan informasi stok dan kapasitas air, sedangkan **TrukTangki** merepresentasikan kendaraan distribusi. Relasi antar kelas menegaskan pembagian tanggung jawab yang jelas dan desain sistem yang modular.

#### 3.2 Bagian Program Sistem Python

## Model/base\_model

```
1  from abc import ABC, abstractmethod
2
3  # Penerapan Abstraction
4  class Identifiable(ABC):
5      """
6      Class abstrak untuk objek yang dapat diidentifikasi dengan UID unik.
7
8      Class ini menerapkan prinsip Abstraction dalam OOP, menyediakan
9      kontrak dasar untuk semua objek yang memerlukan identifikasi unik.
10
11     Attributes:
12         _uid (str): ID unik yang bersifat protected untuk identifikasi objek
13     """
14
15     def __init__(self, uid: str):
16         """
17         Inisialisasi objek dengan UID unik.
18
19         Args:
20             uid (str): ID unik untuk mengidentifikasi objek
21         """
22         self._uid = uid # Protected attribute
23
24     @property
25     def uid(self):
26         """
27         Getter untuk mengakses UID objek.
28
29         Returns:
30             str: ID unik objek
31         """
32         return self._uid
33
34     @abstractmethod
35     def info(self):
36         """
37         Method abstrak untuk menampilkan informasi objek.
38
39         Method ini harus diimplementasikan oleh class turunan untuk
40         menampilkan informasi spesifik sesuai dengan jenis objek.
41
42         Returns:
43             str: Informasi detail tentang objek
44         """
45         pass
```

## Model/entities\_part1

```
1 from .base_model import Identifiable
2 # Penerapan Inheritance (Identifiable -> SumberAir)
3 class SumberAir(Identifiable):
4     """
5     Class untuk merepresentasikan sumber air yang dapat menyimpan air.
6
7     Class ini menerapkan prinsip Inheritance dari Identifiable dan Enkapsulasi
8     dengan menggunakan private attribute untuk stok air.
9
10    Attributes:
11        uid (str): ID unik sumber air (dari parent class)
12        lokasi (str): Lokasi sumber air
13        _kapasitas_max (float): Kapasitas maksimum air dalam liter (protected)
14        __stok_air (float): Stok air saat ini dalam liter (private, enkapsulasi)
15    """
16
17    def __init__(self, uid: str, lokasi: str, kapasitas_max: float):
18        """
19        Inisialisasi sumber air dengan parameter yang diberikan.
20
21        Args:
22            uid (str): ID unik untuk sumber air
23            lokasi (str): Lokasi sumber air
24            kapasitas_max (float): Kapasitas maksimum air dalam liter
25        """
26        super().__init__(uid)
27        self.lokasi = lokasi
28        self._kapasitas_max = kapasitas_max
29        self.__stok_air = 0.0 # Private attribute (Enkapsulasi) [cite: 57]
30
31    # Getter (Akses data private) [cite: 58]
32    @property
33    def stok_air(self):
34        """
35        Getter untuk mengakses stok air (enkapsulasi).
36
37        Returns:
38            float: Jumlah stok air saat ini dalam liter
39        """
40        return self.__stok_air
41
42    # Setter dengan Validasi [cite: 58]
43    @stok_air.setter
44    def stok_air(self, jumlah: float):
45        """
46        Setter untuk mengatur stok air dengan validasi.
47
48        Validasi memastikan stok tidak negatif dan tidak melebihi kapasitas.
49        Jika melebihi kapasitas, stok akan diset ke kapasitas maksimum.
50
51        Args:
52            jumlah (float): Jumlah air yang akan diset dalam liter
53
54        Raises:
55            ValueError: Jika jumlah bernilai negatif
56        """
57        if jumlah < 0:
58            raise ValueError("Stok air tidak boleh negatif.")
59        if jumlah > self._kapasitas_max:
60            self.__stok_air = self._kapasitas_max
61        else:
62            self.__stok_air = jumlah
```

## Model/entities\_part2

```
1      # Polymorphism (Override method info) [cite: 62]
2      def info(self):
3          """
4              Menampilkan informasi sumber air (implementasi dari abstract method).
5
6              Menerapkan prinsip Polymorphism dengan meng-override method dari parent class.
7
8              Returns:
9                  str: String berisi informasi lokasi dan stok air
10             """
11             return f"Sumber Air di {self.lokasi} | Stok: {self.__stok_air}/{self._kapasitas_max} L"
12
13
14     class TrukTangki(SumberAir):
15         """
16             Class untuk merepresentasikan truk tangki air.
17
18             Class ini merupakan turunan dari SumberAir yang menerapkan prinsip
19             Inheritance dan Polymorphism dengan spesialisasi untuk truk tangki.
20
21             Attributes:
22                 uid (str): ID unik truk (dari grandparent class)
23                 lokasi (str): Selalu "Mobil" untuk truk (dari parent class)
24                 _kapasitas_max (float): Kapasitas tangki dalam liter (dari parent class)
25                 plat_nomor (str): Nomor plat kendaraan truk
26         """
27
28         def __init__(self, uid: str, plat_nomor: str, kapasitas: float):
29             """
30                 Inisialisasi truk tangki dengan parameter yang diberikan.
31
32                 Args:
33                     uid (str): ID unik untuk truk
34                     plat_nomor (str): Nomor plat kendaraan
35                     kapasitas (float): Kapasitas tangki dalam liter
36             """
37             super().__init__(uid, "Mobil", kapasitas)
38             self.plat_nomor = plat_nomor
39
40     # Polymorphism: Truk punya info spesifik
41     def info(self):
42         """
43             Menampilkan informasi truk tangki (polymorphism).
44
45             Meng-override method info() dari parent class untuk menambahkan
46             informasi spesifik truk yaitu nomor plat.
47
48             Returns:
49                 str: String berisi nomor plat dan informasi detail truk
50         """
51         base_info = super().info()
52         return f"[TRUK {self.plat_nomor}] {base_info}"
```

## Repositories/base\_repository

```
1 from abc import ABC, abstractmethod
2 from typing import List, Any
3
4 # Kontrak (Interface) untuk penyimpanan data
5 class BaseRepository(ABC):
6     """
7     Abstract Base Class untuk repository pattern.
8
9     Class ini mendefinisikan kontrak (interface) untuk semua implementasi
10    repository yang mengelola penyimpanan dan pengambilan data.
11    Menerapkan prinsip Abstraction dan Dependency Inversion.
12    """
13
14    @abstractmethod
15    def add(self, item: Any):
16        """
17        Menambahkan item baru ke dalam storage.
18
19        Args:
20            item (Any): Objek yang akan ditambahkan ke repository
21        """
22        pass
23
24    @abstractmethod
25    def get_all(self) -> List[Any]:
26        """
27        Mengambil semua item dari storage.
28
29        Returns:
30            List[Any]: List berisi semua item yang tersimpan
31        """
32        pass
33
34    @abstractmethod
35    def find_by_id(self, uid: str):
36        """
37        Mencari item berdasarkan UID.
38
39        Args:
40            uid (str): ID unik item yang dicari
41
42        Returns:
43            Any: Item yang ditemukan, atau None jika tidak ada
44        """
45        pass
```

## Repositories/file\_repository - 1

```
1  from .base_repository import BaseRepository
2  from models.entities import TrukTangki, SumberAir
3  from utils.logger import logger
4  from typing import List, Any
5  import json
6  import os
7
8  class FileRepository(BaseRepository):
9      """
10         Repository dengan penyimpanan persisten ke file JSON.
11
12         Implementasi dari BaseRepository yang menyimpan data ke file JSON
13         untuk persistensi data. Mendukung auto-save setiap ada perubahan data.
14
15         Attributes:
16             _storage (List): List internal untuk menyimpan objek di memory
17             file_path (str): Path ke file JSON untuk penyimpanan
18         """
19
20     def __init__(self, file_path: str = "data_truk.json"):
21         """
22             Inisialisasi repository dan memuat data dari file.
23
24             Args:
25                 file_path (str): Path ke file JSON. Default: "data_truk.json"
26         """
27         self._storage = []
28         self.file_path = file_path
29         self.load_from_file()
30
31     def add(self, item: Any):
32         """
33             Menambah item baru dan langsung menyimpan ke file.
34
35             Args:
36                 item (Any): Objek yang akan ditambahkan (harus punya atribut uid)
37         """
38         self._storage.append(item)
39         self.save_to_file()
40         logger.info(f>Data {item.uid} ditambahkan dan disimpan ke file.")
41
42     def get_all(self) -> List[Any]:
43         """
44             Mengambil semua item dari storage.
45
46             Returns:
47                 List[Any]: List berisi semua item yang tersimpan
48         """
49         return self._storage
50
51     def find_by_id(self, uid: str):
52         """
53             Mencari item berdasarkan UID.
54
55             Args:
56                 uid (str): ID unik item yang dicari
57
58             Returns:
59                 Any: Item yang ditemukan, atau None jika tidak ditemukan
60         """
61         for item in self._storage:
62             if item.uid == uid:
63                 return item
64         return None
```


## Repositories/file\_repository - 2

```
1  def update(self, item: Any):
2      """
3      Update item yang sudah ada dan simpan ke file.
4
5      Args:
6          item (Any): Objek baru yang akan menggantikan objek lama dengan UID sama
7
8      Returns:
9          bool: True jika berhasil update, False jika item tidak ditemukan
10     """
11     for i, stored_item in enumerate(self._storage):
12         if stored_item.uid == item.uid:
13             self._storage[i] = item
14             self.save_to_file()
15             logger.info(f"Data {item.uid} diupdate dan disimpan ke file.")
16             return True
17     return False
18
19 def save_to_file(self):
20     """
21     Menyimpan semua data ke file JSON.
22
23     Melakukan serialisasi objek menjadi dictionary dan menyimpannya
24     ke file JSON dengan format yang mudah dibaca (indented).
25     Mendukung objek TrukTangki dan SumberAir.
26
27     Raises:
28         Exception: Jika terjadi error saat menulis ke file
29     """
30     try:
31         data_to_save = []
32         for item in self._storage:
33             if isinstance(item, TrukTangki):
34                 data_to_save.append(
35                     {
36                         "type": "TrukTangki",
37                         "uid": item.uid,
38                         "plat_nomor": item.plat_nomor,
39                         "kapasitas_max": item._kapasitas_max,
40                         "stok_air": item.stok_air,
41                     }
42                 )
43             elif isinstance(item, SumberAir):
44                 data_to_save.append(
45                     {
46                         "type": "SumberAir",
47                         "uid": item.uid,
48                         "lokasi": item.lokasi,
49                         "kapasitas_max": item._kapasitas_max,
50                         "stok_air": item.stok_air,
51                     }
52                 )
53
54         with open(self.file_path, "w", encoding="utf-8") as f:
55             json.dump(data_to_save, f, indent=4, ensure_ascii=False)
56
57         logger.info(f"Data berhasil disimpan ke {self.file_path}")
58     except Exception as e:
59         logger.error(f"Gagal menyimpan data ke file: {e}")
```

## Repositories/file\_repository - 3

```
1  def load_from_file(self):
2      """
3      Memuat data dari file JSON ke memory.
4
5      Melakukan deserialisasi data dari file JSON menjadi objek
6      TrukTangki atau SumberAir. Jika file tidak ada atau corrupt,
7      akan memulai dengan storage kosong.
8
9      Raises:
10         json.JSONDecodeError: Jika format JSON tidak valid
11         Exception: Jika terjadi error lain saat membaca file
12      """
13     if not os.path.exists(self.file_path):
14         logger.info(f"File {self.file_path} tidak ditemukan. Membuat file baru.")
15         self._storage = []
16         return
17
18     try:
19         with open(self.file_path, "r", encoding="utf-8") as f:
20             data = json.load(f)
21
22         self._storage = []
23         for item_data in data:
24             if item_data["type"] == "TrukTangki":
25                 truk = TrukTangki(
26                     uid=item_data["uid"],
27                     plat_nomor=item_data["plat_nomor"],
28                     kapasitas=item_data["kapasitas_max"],
29                 )
30                 truk.stok_air = item_data["stok_air"]
31                 self._storage.append(truk)
32             elif item_data["type"] == "SumberAir":
33                 sumber = SumberAir(
34                     uid=item_data["uid"],
35                     lokasi=item_data["lokasi"],
36                     kapasitas_max=item_data["kapasitas_max"],
37                 )
38                 sumber.stok_air = item_data["stok_air"]
39                 self._storage.append(sumber)
40
41         logger.info(
42             f"Berhasil memuat {len(self._storage)} data dari {self.file_path}"
43         )
44     except json.JSONDecodeError:
45         logger.warning(
46             f"File {self.file_path} kosong atau format tidak valid. Memulai dengan data kosong."
47         )
48         self._storage = []
49     except Exception as e:
50         logger.error(f"Gagal memuat data dari file: {e}")
51         self._storage = []
```

## Repositories/ memory\_repository



```
1  from .base_repository import BaseRepository
2  from typing import List, Any
3
4  # Implementasi penyimpanan sederhana (List)
5  class MemoryRepository(BaseRepository):
6      def __init__(self):
7          self._storage = []
8
9      def add(self, item: Any):
10         self._storage.append(item)
11
12     def get_all(self) -> List[Any]:
13         return self._storage
14
15     def find_by_id(self, uid: str):
16         for item in self._storage:
17             if item.uid == uid:
18                 return item
19         return None
```

## Services/water\_service - 1

```
1 from repositories.base_repository import BaseRepository
2 from models.entities import SumberAir
3 from utils.logger import logger
4 import datetime # [cite: 70]
5
6
7 class WaterDistributionService:
8     """
9     Service layer untuk mengelola distribusi air.
10
11     Class ini mengimplementasikan business logic untuk manajemen air,
12     termasuk pendaftaran sumber air, pengisian, pengurangan, dan pengecekan status.
13     Menerapkan prinsip Dependency Injection untuk loose coupling.
14
15     Attributes:
16         water_repo (BaseRepository): Repository untuk penyimpanan data sumber air
17     """
18
19     # Dependency Injection: Repository dimasukkan lewat constructor [cite: 68]
20     def __init__(self, water_repo: BaseRepository):
21         """
22         Inisialisasi service dengan repository (Dependency Injection).
23
24         Args:
25             water_repo (BaseRepository): Instance repository untuk data persistence
26         """
27         self.water_repo = water_repo
28
29     def daftar_sumber_air(self, sumber: SumberAir):
30         """
31         Mendaftarkan sumber air baru ke sistem.
32
33         Args:
34             sumber (SumberAir): Objek sumber air yang akan didaftarkan
35         """
36         self.water_repo.add(sumber)
37         logger.info(f"Sumber air {sumber.uid} berhasil didaftarkan.") # Logging
38         print(f"✓ Truk {sumber.uid} berhasil ditambahkan dan disimpan!")
39
40     def isi_ulang_air(self, uid_sumber: str, jumlah: float):
41         """
42         Mengisi ulang air ke sumber air tertentu.
43
44         Mencari sumber air berdasarkan UID, kemudian menambah stok air
45         sesuai jumlah yang diberikan. Perubahan langsung disimpan ke file.
46
47         Args:
48             uid_sumber (str): ID unik sumber air yang akan diisi
49             jumlah (float): Jumlah air dalam liter yang akan ditambahkan
50         """
51         sumber = self.water_repo.find_by_id(uid_sumber)
52         if sumber:
53             try:
54                 # Menggunakan setter yang sudah ada validasinya
55                 stok_sekarang = sumber.stok_air
56                 sumber.stok_air = stok_sekarang + jumlah
57                 logger.info(
58                     f"Isi ulang {jumlah}L ke {uid_sumber} berhasil. Waktu: {datetime.datetime.now()}"
59                 )
60                 # Simpan perubahan ke file
61                 (
62                     self.water_repo.save_to_file()
63                     if hasattr(self.water_repo, "save_to_file")
64                     else None
65                 )
66                 print(
67                     f"✓ Berhasil mengisi {jumlah}L air. Stok sekarang: {sumber.stok_air}L"
68                 )
69             except ValueError as e:
70                 logger.error(f"Gagal isi ulang: {e}")
71         else:
72             logger.warning(f"Sumber air {uid_sumber} tidak ditemukan.")
```

## Services/water\_service - 2

```
1 def kurangi_air(self, uid_sumber: str, jumlah: float):
2     """
3     Mengurangi isi air dari sumber air (misalnya untuk distribusi ke warga).
4
5     Mencari sumber air berdasarkan UID, memvalidasi ketersediaan stok,
6     kemudian mengurangi stok air. Perubahan langsung disimpan ke file.
7     Akan menampilkan error jika stok tidak mencukupi.
8
9     Args:
10         uid_sumber (str): ID unik sumber air yang akan dikurangi
11         jumlah (float): Jumlah air dalam liter yang akan dikurangi
12     """
13     sumber = self.water_repo.find_by_id(uid_sumber)
14     if sumber:
15         try:
16             stok_sekarang = sumber.stok_air
17             if jumlah > stok_sekarang:
18                 logger.error(
19                     f"Gagal kurangi air: Jumlah {jumlah}L melebihi stok ({stok_sekarang}L)"
20                 )
21                 print(
22                     f"✗ Stok air tidak mencukupi! Stok saat ini: {stok_sekarang}L"
23                 )
24             else:
25                 sumber.stok_air = stok_sekarang - jumlah
26                 # Simpan perubahan ke file
27                 (
28                     self.water_repo.save_to_file()
29                     if hasattr(self.water_repo, "save_to_file")
30                     else None
31                 )
32                 logger.info(
33                     f"Kurangi {jumlah}L dari {uid_sumber} berhasil. Sisa: {sumber.stok_air}L. Waktu: {datetime.datetime.now()}"
34                 )
35                 print(
36                     f"✓ Berhasil mengurangi {jumlah}L air. Sisa stok: {sumber.stok_air}L"
37                 )
38         except ValueError as e:
39             logger.error(f"Gagal kurangi air: {e}")
40     else:
41         logger.warning(f"Sumber air {uid_sumber} tidak ditemukan.")
42         print(f"✗ Truk dengan ID {uid_sumber} tidak ditemukan.")
43
44 def cek_status_semua(self):
45     """
46     Menampilkan status semua sumber air yang terdaftar.
47
48     Mengambil semua sumber air dari repository dan menampilkan
49     informasi detail masing-masing sumber air ke console.
50     """
51     semua_sumber = self.water_repo.get_all()
52     print("\n--- STATUS KETERSEDIAAN AIR ---")
53     for s in semua_sumber:
54         print(s.info())
```

## Utils/logger.py

```
1 import logging
2
3 def setup_logger():
4     """
5     Mengkonfigurasi dan membuat logger untuk aplikasi.
6
7     Mengatur format log dengan timestamp, level, dan pesan.
8     Logger dikonfigurasi dengan level INFO untuk mencatat
9     semua aktivitas penting dalam sistem.
10
11     Returns:
12         logging.Logger: Instance logger yang sudah dikonfigurasi
13     """
14     # Mengatur format log
15     logging.basicConfig(
16         level=logging.INFO,
17         format="%(asctime)s - %(levelname)s - %(message)s",
18         datefmt="%Y-%m-%d %H:%M:%S",
19     )
20     return logging.getLogger("WaterSystem")
21
22 logger = setup_logger()
```

## data\_truk.json

```
1 [
2     {
3         "type": "TrukTangki",
4         "uid": "01",
5         "plat_nomor": "111",
6         "kapasitas_max": 5000.0,
7         "stok_air": 500.0
8     },
9     {
10         "type": "TrukTangki",
11         "uid": "02",
12         "plat_nomor": "222",
13         "kapasitas_max": 3000.0,
14         "stok_air": 0.0
15     }
16 ]
```

## Main.py - 1

```
1  """Application entry point untuk sistem manajemen air bersih.
2
3  Script ini menyediakan interface CLI untuk mengelola distribusi air
4  melalui armada truk tangki, termasuk fitur:
5  - Pendaftaran truk baru
6  - Pengisian ulang air
7  - Penggunaan/distribusi air
8  - Monitoring status air
9
10 Data disimpan secara persisten ke file JSON.
11 """
12
13 from models.entities import TrukTangki
14 from repositories.file_repository import FileRepository
15 from services.water_service import WaterDistributionService
16 from utils.logger import logger
17
18
19 def main():
20     """
21     Fungsi utama aplikasi manajemen air bersih.
22
23     Menginisialisasi komponen sistem (repository dan service),
24     kemudian menjalankan loop interaktif untuk menerima input
25     pengguna dan menjalankan operasi yang sesuai.
26     """
27     logger.info("Aplikasi Manajemen Air Bersih Dimulai...")
28
29     # 1. Setup Dependency Injection
30     # Kita buat repository dengan file persistence
31     repo_air = FileRepository("data_truk.json")
32
33     # Kita masukkan wadah data ke service (logika)
34     service = WaterDistributionService(repo_air)
35
36     print(
37         f"\n📁 Data dimuat dari file. Total truk terdaftar: {len(repo_air.get_all())}"
38     )
```

## Main.py - 2

```
1      # 2. Simulasi Interaksi User (CLI Loop bisa ditambahkan di sini)
2      while True:
3          print("\n=== MENU POSKO KEKERINGAN ===")
4          print("1. Tambah Armada Truk Air")
5          print("2. Isi Ulang Air")
6          print("3. Gunakan Air dari Truk")
7          print("4. Cek Status Air")
8          print("5. Keluar")
9
10         pilihan = input("Pilih menu: ")
11
12         if pilihan == "1":
13             uid = input("ID Truk: ")
14             plat = input("Plat Nomor: ")
15             kap = float(input("Kapasitas (Liter): "))
16             truk_baru = TrukTangki(uid, plat, kap)
17             service.daftar_sumber_air(truk_baru)
18
19         elif pilihan == "2":
20             uid = input("ID Truk yang diisi: ")
21             jml = float(input("Jumlah Air (Liter): "))
22             service.isi_ulang_air(uid, jml)
23
24         elif pilihan == "3":
25             uid = input("ID Truk yang akan digunakan: ")
26             jml = float(input("Jumlah Air yang akan digunakan (Liter): "))
27             service.kurangi_air(uid, jml)
28
29         elif pilihan == "4":
30             service.cek_status_semua()
31
32         elif pilihan == "5":
33             logger.info("Aplikasi ditutup.")
34             break
35         else:
36             print("Pilihan tidak valid.")
37
38
39     if __name__ == "__main__":
40         main()
```

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Berdasarkan hasil perancangan dan implementasi proyek UAS Pemrograman Berorientasi Objek dengan judul Manajemen Kebutuhan Air Bersih Saat Kekeringan, dapat disimpulkan bahwa sistem yang dibangun mampu menggambarkan proses pengelolaan kebutuhan air bersih secara terstruktur melalui pemanfaatan kode program. Proyek ini berhasil menunjukkan alur kerja sistem serta implementasi logika program yang disusun secara modular dan terorganisir sesuai dengan kebutuhan studi kasus yang diberikan.

#### **Link GitHub**

**[https://github.com/ItsAltoo/UAS\\_Manajemen-Kebutuhan-Air-Bersih-saat-Kekeringan Kelompok 2](https://github.com/ItsAltoo/UAS_Manajemen-Kebutuhan-Air-Bersih-saat-Kekeringan_Kelompok_2)**