# Suburb Model Python Classes Documentation

This document describes the Python classes generated to model a suburban area based on geographical and structural elements observed in a map image. These classes are designed to be decoupled and reusable for representing different parts of a suburb in a software system, particularly for applications like smart waste management.
The classes are saved in separate files within a suburb_model directory by the accompanying bash script.

## 1. Location Class

- **File:** suburb_model/location.py
- **Description:** Represents a specific geographical point using latitude and longitude coordinates. This is a fundamental building block for defining the position of other elements like houses, driveways, or bins.
- **Parameters (__init__)**:
    - latitude (float): The latitude coordinate of the location.
    - longitude (float): The longitude coordinate of the location.
- **Fields (Attributes)**:
    - latitude (float): Stores the latitude.
    - longitude (float): Stores the longitude.
- **Methods**:
    - __str__(self) -> str: Returns a user-friendly string representation (e.g., "Location(lat=..., lon=...)").
    - __repr__(self) -> str: Returns a developer-friendly string representation.
    - __eq__(self, other) -> bool: Compares two Location objects for equality based on their coordinates.
    - to_dict(self) -> dict: Returns the location as a dictionary {"latitude": ..., "longitude": ...}.

## 2. Driveway Class

- **File:** suburb_model/driveway.py
- **Description:** Represents a driveway, which serves as a potential access or placement point for a bin associated with a house. It has a specific geographical location.
- **Parameters (__init__)**:
    - location (Location): The geographical location of the driveway access point (an instance of the Location class).
    - identifier (str, optional): A unique string identifier for the driveway. Defaults to a generated ID if not provided.
- **Fields (Attributes)**:

    - location (Location): Stores the Location object of the driveway.
    - identifier (str): Stores the unique identifier.
- **Methods**:
    - __str__(self) -> str: Returns a user-friendly string representation.
    - __repr__(self) -> str: Returns a developer-friendly string representation.
    - get_location(self) -> Location: Returns the Location object of the driveway.
    - get_identifier(self) -> str: Returns the unique identifier of the driveway.

## 3. House Class

- **File:** suburb_model/house.py
- **Description:** Represents an individual house or property. A house has an address, a general location, and can have one or more associated driveways.
- **Parameters (__init__)**:
    - address (str): The street address of the house (e.g., "123 Main St").
    - location (Location): The geographical location of the house (e.g., the centroid of the property).
    - property_id (str, optional): A unique string identifier for the property. Defaults to a generated ID.
    - driveways (List[Driveway], optional): A list of Driveway objects associated with this house. Defaults to an empty list.
- **Fields (Attributes)**:
    - address (str): Stores the street address.
    - location (Location): Stores the Location object of the house.
    - property_id (str): Stores the unique identifier.
    - driveways (List[Driveway]): Stores a list of associated Driveway objects.
- **Methods**:
    - __str__(self) -> str: Returns a user-friendly string representation.
    - __repr__(self) -> str: Returns a developer-friendly string representation.
    - add_driveway(self, driveway: Driveway): Adds a Driveway object to the house's list of driveways.
    - get_address(self) -> str: Returns the address of the house.
    - get_location(self) -> Location: Returns the geographical location of the house.
    - get_driveways(self) -> List[Driveway]: Returns the list of driveways associated with the house.

## 4. Street Class

- **File:** suburb_model/street.py
- **Description:** Represents a street within the suburb. A street is primarily defined by its name and the collection of houses located along it.
- **Parameters (__init__)**:
    - name (str): The name of the street (e.g., "Main St").
    - street_id (str, optional): A unique string identifier for the street. Defaults to a

generated ID.
    - houses (List[House], optional): A list of House objects located on this street. Defaults to an empty list.
- **Fields (Attributes)**:
    - name (str): Stores the name of the street.
    - street_id (str): Stores the unique identifier.
    - houses (List[House]): Stores a list of House objects on the street.
    - *(Future Enhancement):* Could include geometry data (e.g., a list of Location points defining the street's path).
- **Methods**:
    - __str__(self) -> str: Returns a user-friendly string representation.
    - __repr__(self) -> str: Returns a developer-friendly string representation.
    - add_house(self, house: House): Adds a House object to the street's list of houses.
    - get_name(self) -> str: Returns the name of the street.
    - get_houses(self) -> List[House]: Returns the list of houses located on the street.

## 5. Suburb Class

- **File:** suburb_model/suburb.py
- **Description:** Represents the entire suburban area being modeled. It acts as a container for multiple streets and provides methods to access elements within the suburb.
- **Parameters (__init__)**:
    - name (str): The name of the suburb.
    - suburb_id (str, optional): A unique string identifier for the suburb. Defaults to a generated ID.
    - streets (List[Street], optional): A list of Street objects within this suburb. Defaults to an empty list.
- **Fields (Attributes)**:
    - name (str): Stores the name of the suburb.
    - suburb_id (str): Stores the unique identifier.
    - streets (List[Street]): Stores a list of Street objects in the suburb.
    - *(Future Enhancement):* Could include boundary data (e.g., a polygon of Location points defining the suburb's area).
- **Methods**:
    - __str__(self) -> str: Returns a user-friendly string representation.
    - __repr__(self) -> str: Returns a developer-friendly string representation.
    - add_street(self, street: Street): Adds a Street object to the suburb's list of streets.
    - get_name(self) -> str: Returns the name of the suburb.
    - get_streets(self) -> List[Street]: Returns the list of streets within the suburb.
    - get_all_houses(self) -> List[House]: Returns a flattened list of all House objects contained within all streets in the suburb.

These classes provide a basic but decoupled structure for modeling your suburb based on the visual elements in the map image. You can extend them in the future with more attributes and methods as your project requires (e.g., adding bin objects to driveways or houses, defining street geometries, adding property boundaries).