```
# Base Class: Pollutant
class Pollutant:
    def __init__(self, name, concentration):
        self.name = name
        self.__concentration = concentration  # Private attribute in
micrograms per cubic meter (\u03bcg/m^3)

    def get_concentration(self):
        return self.__concentration

    def set_concentration(self, concentration):
        if concentration >= 0:
            self.__concentration = concentration
        else:
            raise ValueError("Concentration must be non-negative.")

    def calculate_aqi(self):
        raise NotImplementedError("Subclasses must implement this
method.")

# Subclass: PM2.5
class PM25(Pollutant):
    def __init__(self, concentration):
        super().__init__("PM2.5", concentration)

    def calculate_aqi(self):
        """Simplified AQI calculation for PM2.5."""
        return min(500, max(0, (self.get_concentration() / 12.0) * 100))

# Subclass: Ozone
class Ozone(Pollutant):
    def __init__(self, concentration):
        super().__init__("Ozone", concentration)

    def calculate_aqi(self):
        """Simplified AQI calculation for Ozone."""
        return min(500, max(0, (self.get_concentration() / 0.070) * 100))
```

Base Class: `Pollutant`

The `Pollutant` class provides a foundation for different types of pollutants:

- **Attributes**
  - `name` (public): Name of the pollutant (e.g., PM2.5, Ozone).
  - `__concentration` (private): Concentration of the pollutant in µg/m³.
- **Encapsulation**
  - `get_concentration()` and `set_concentration()` ensure only valid values are assigned to `__concentration`.
- **Abstract Method**

- `calculate_aqi()`: Must be implemented by subclasses.

Example

```
pollutant = Pollutant("Generic", 50.0)  # Cannot directly instantiate
because of the abstract method.
```

Subclass: `PM25`

The `PM25` class specializes `Pollutant` for PM2.5:

- **Methods**
  - Implements `calculate_aqi()` using the AQI formula for PM2.5.

Example

```
pm25 = PM25(35.0)
print(pm25.calculate_aqi())  # Output: Simplified AQI for PM2.5
```

Subclass: `Ozone`

The `Ozone` class specializes `Pollutant` for ozone:

- **Methods**
  - Implements `calculate_aqi()` using the AQI formula for ozone.

Example

```
ozone = Ozone(0.060)
print(ozone.calculate_aqi())  # Output: Simplified AQI for Ozone
```

Summary

This pollutant monitoring example demonstrates:

1. **Encapsulation**: Protecting and validating pollutant concentrations.
2. **Inheritance**: Creating specialized pollutant classes for PM2.5 and Ozone.
3. **Abstraction**: Using a base class (`Pollutant`) to define shared behavior and enforce implementation of specific methods in subclasses.