Milestone 1: Power System Equipment Classes

Joseph Rivera and Michael Bliesath

Keywords: Bus, Generator, Load, Transformer, Transmission Line.

Abstract. This report presents Milestone 1 of a power system modeling project implemented in Python. The objective of this milestone is to develop foundational equipment classes representing core components of an electric power system, including buses, generators, loads, transformers, and transmission lines. Each class encapsulates key electrical parameters and establishes a structured object-oriented architecture that will be extended in future milestones for power flow analysis and system simulation.

1 Introduction

Modern power system analysis relies heavily on structured software models that accurately represent electrical equipment and network topology. The purpose of Milestone 1 is to establish the foundational equipment classes that will serve as the building blocks for a full power flow solver.

This milestone focuses on creating structured Python classes for the following components:

- Bus
- Generator
- Load
- Transformer
- Transmission Line

Each class defines the necessary electrical parameters as attributes and prepares the overall architecture for integration into a larger solver framework.

2 Data and Methods

This milestone implements object-oriented modeling using Python. Each equipment component is represented as an independent class with defined inputs (constructor parameters) and stored attributes.

2.1 Bus Class

The Bus class represents a node in the power system network. Each bus contains:

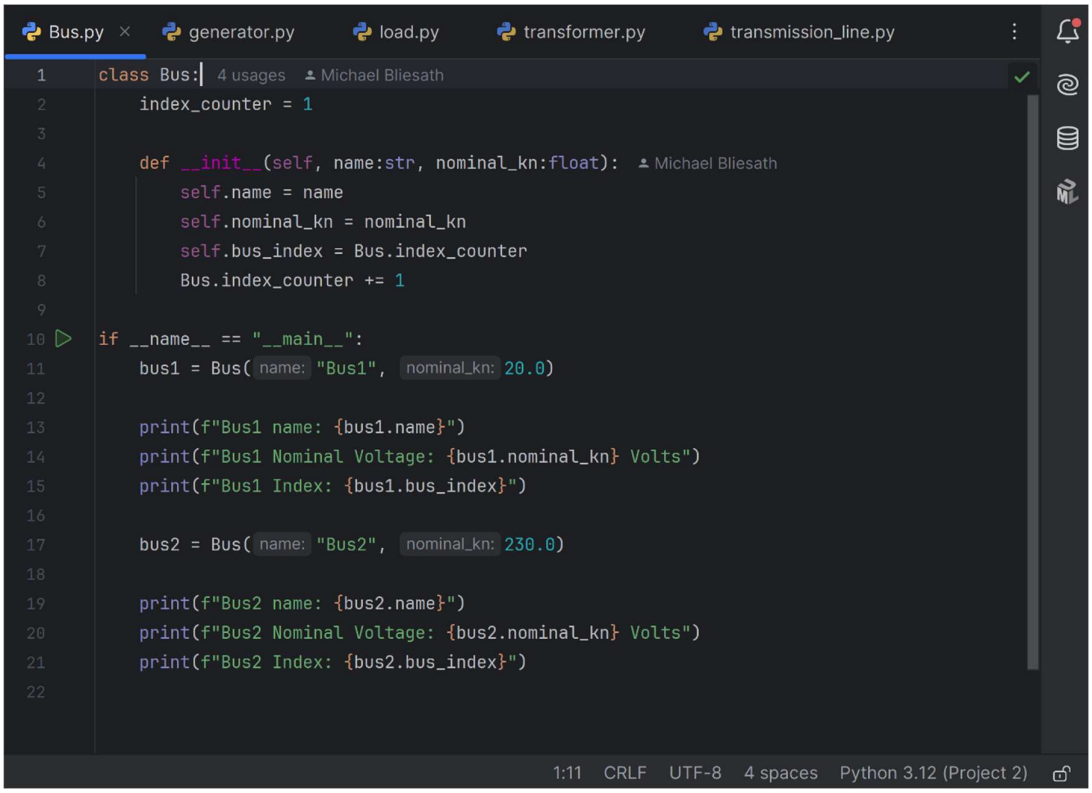
- **name** (string): Unique identifier of the bus
- **nominal_kv** (float): Nominal voltage level in kilovolts
- **bus_index** (integer): Automatically assigned index

The bus index is generated using a class-level counter:

class Bus :

index_counter = 1
def __init__ (self name : str, nominal_kv : float) :
self.name = name
self.nominal_kv = nominal_kv
self.bus_index = Bus.index counter
Bus.index_counter += 1

This ensures that each bus receives a unique index automatically upon instantiation.



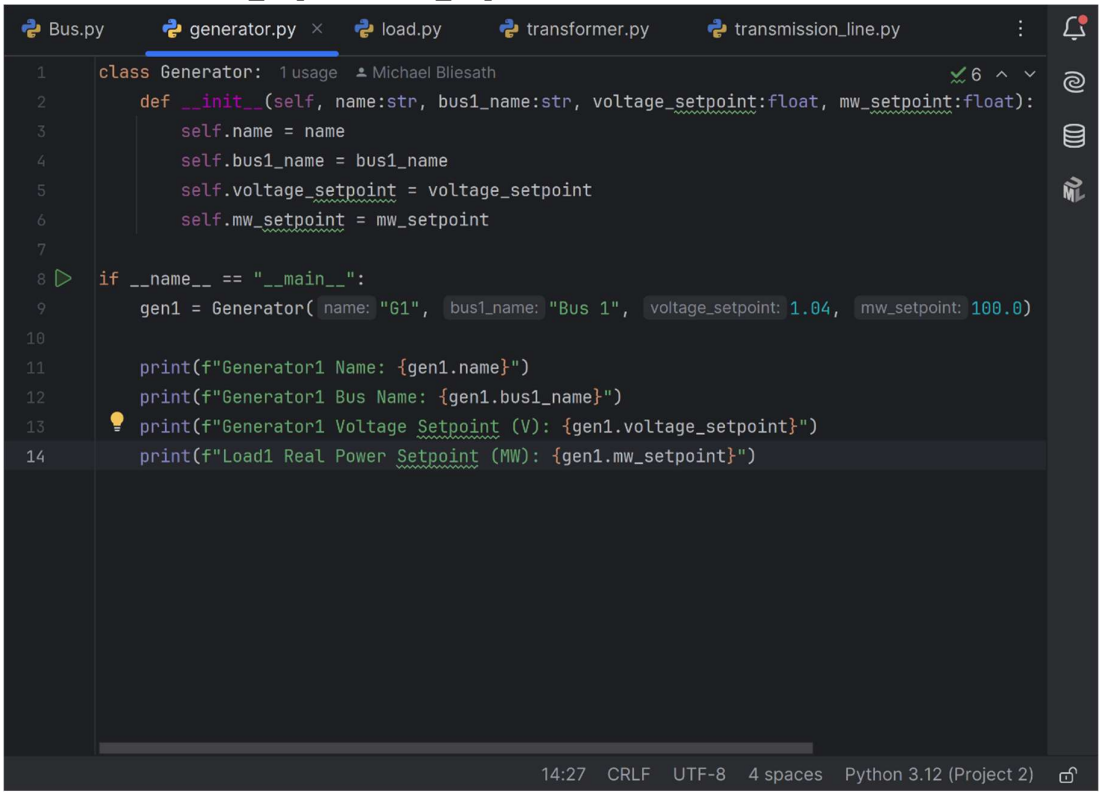
2.2 Generator Class

The Generator class models a generator connected to a bus. Its inputs include:

- **name** (string)
- **bus1_name** (string)
- **voltage_setpoint** (float, per-unit)
- **mw_setpoint** (float, MW)

class Generator :
def __init__ (self, name : str , bus1_name : str , voltage_setpoint : float , mw_setpoint : float) :
self.name = name
self.bus1_name = bus1_name
self.voltage_setpoint = voltage_setpoint

self.mw_setpoint = mw_setpoint



2.3 Load Class

The Load class represents a power demand connected to a bus. Inputs include:

- **name** (string)
- **bus1_name** (string)
- **mw** (float): Real power demand
- **mvar** (float): Reactive power demand

class Load :

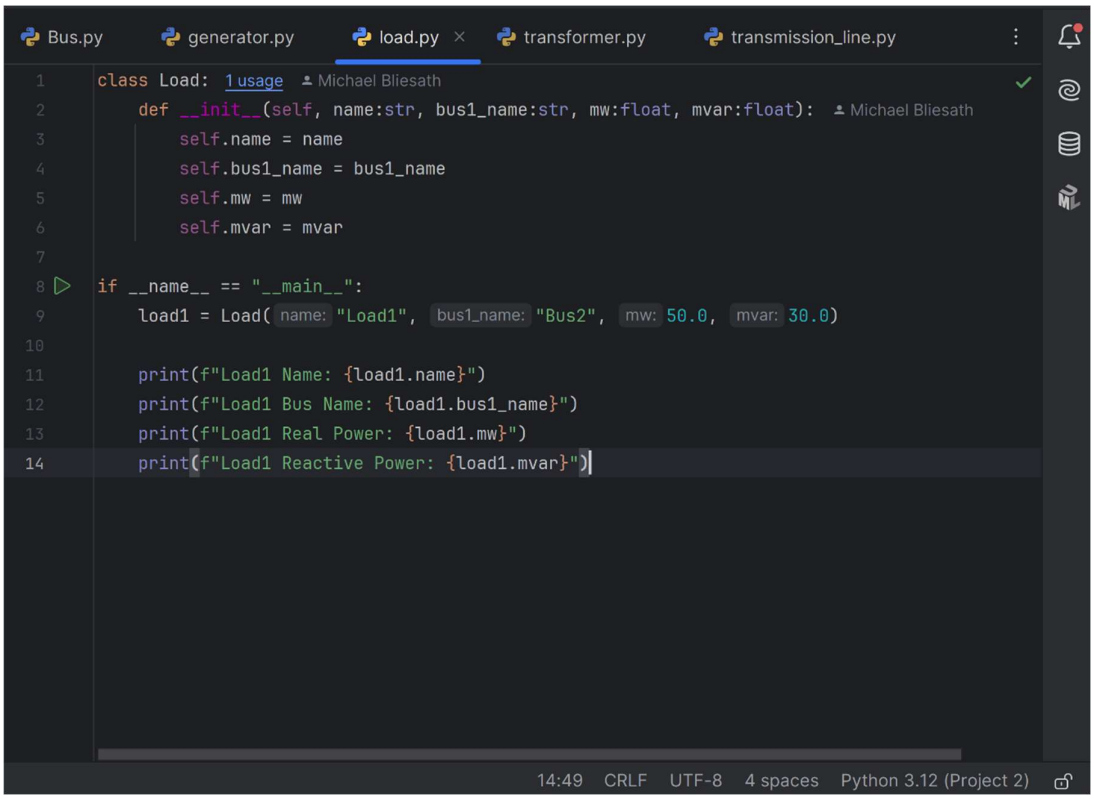
def __init__ (self , name : str, bus1_name : str, mw : float, mvar : float) :

self. name = name

self.bus1_name = bus1_name

self.mw = mw

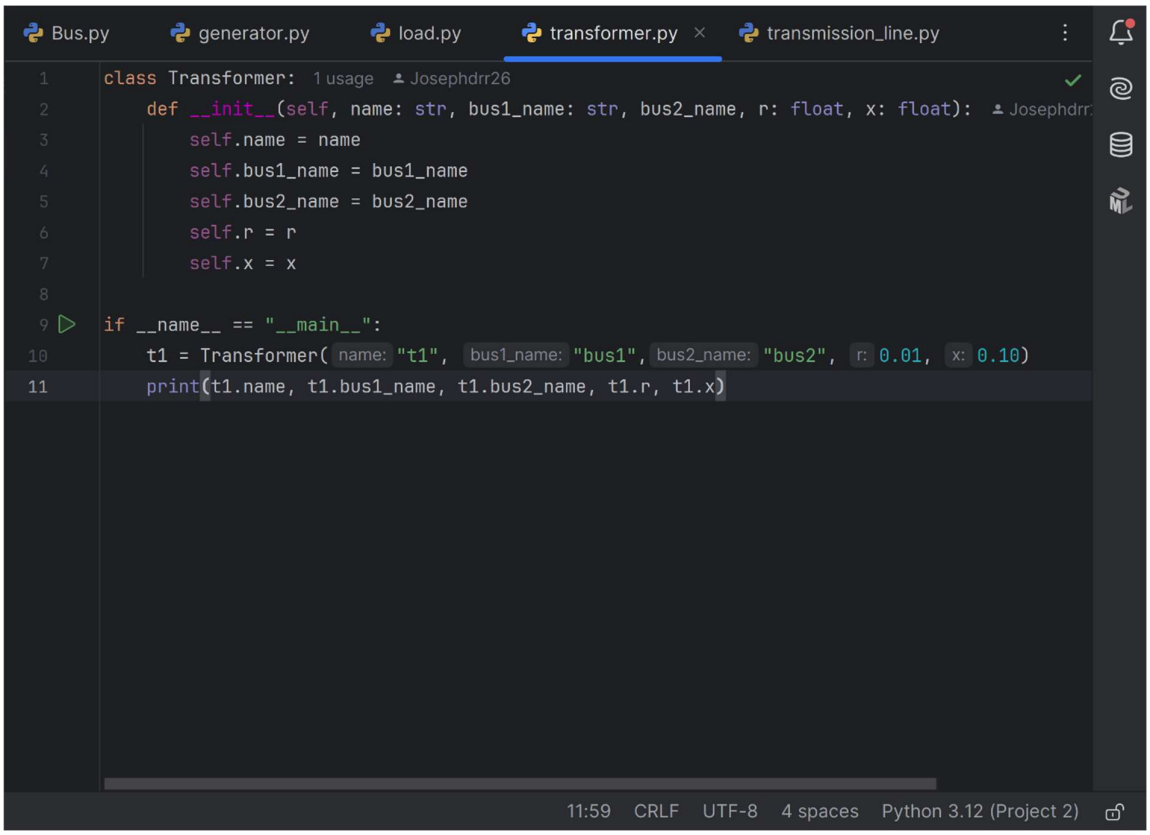
self.mvar = mvar



2.4 Transformer Class

The Transformer class models a transformer connecting two buses. It includes resistance and reactance parameters:

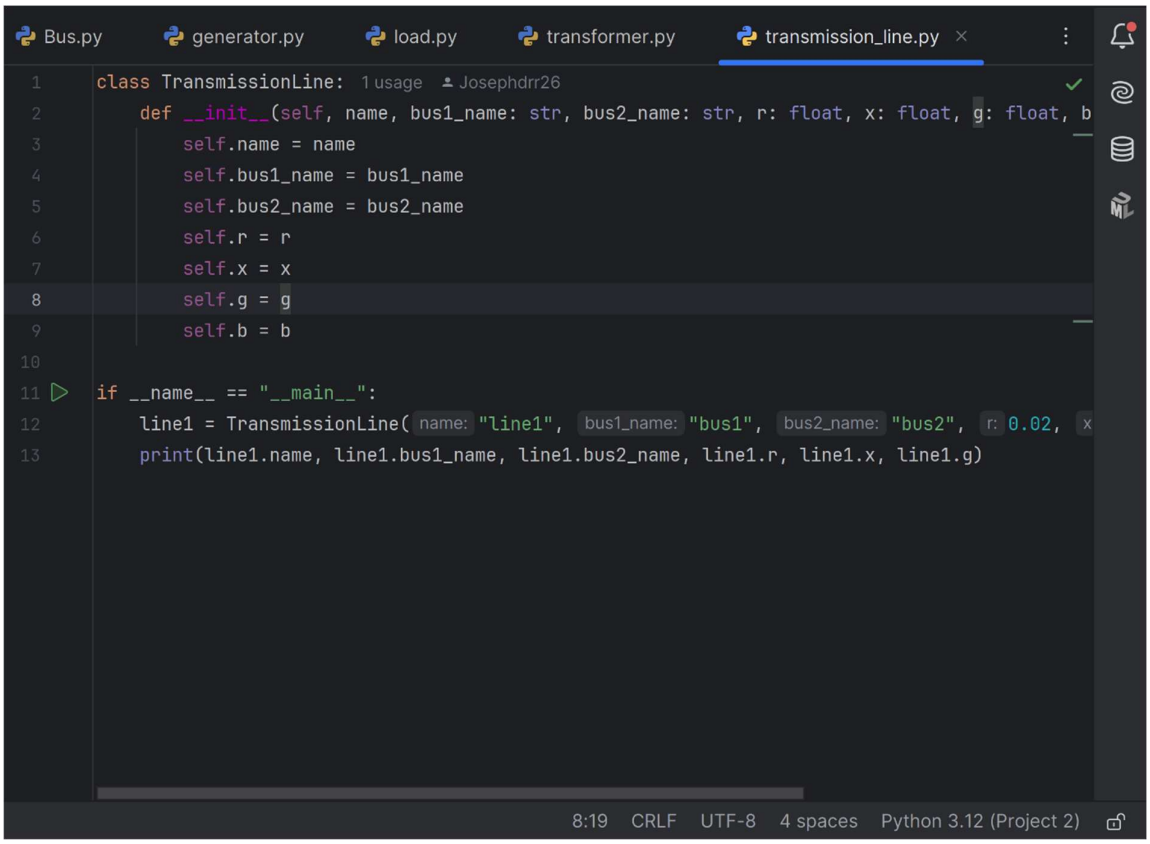
```
class Transformer :
    def __init__ (self , name : str, bus1_name : str, bus2_name : str, r
                  : float, x : float) :
        self.name = name
        self.bus1_name = bus1_name
        self.bus2_name = bus2_name
        self.r = r
        self.x = x
```



2.5 Transmission Line Class

The TransmissionLine class models a transmission line using series impedance and shunt admittance parameters:

```
class        TransmissionLine :
    def __init__ (self, name : str, bus1_name : str, bus2_name : str , r
                        : float, x : float, g : float, b : float) :
        self.name = name
        self.bus1_name = bus1_name
        self.bus2_name = bus2_name
        self.r = r
        self.x = x
        self.g = g
        self.b = b
```



3 Results

The implemented classes successfully instantiate equipment objects with proper attribute storage. Each class can be independently tested using a main block, confirming:

- Proper attribute assignment
- Unique bus index generation
- Structured storage of electrical parameters

These validated equipment models establish the structural foundation for subsequent solver development.

4 Conclusion

Milestone 1 establishes the object-oriented architecture required for a power system modeling framework. By implementing core equipment classes, the project now has structured representations of buses, generators, loads, transformers, and transmission lines.

The automatic indexing of buses ensures consistent system organization, while clear parameter definitions prepare the system for integration into future power flow solvers.

Future milestones will integrate these classes into a unified circuit representation and implement numerical solution methods for steady-state power flow analysis.

Conflict of interests

None.

Acknowledgments

Not applicable for Milestone 1.

Funding

Not applicable.

Availability of data and software code

All software code was developed in Python as part of this milestone and is maintained within the project repository.