Take Home 5 — OOP Experiment

Observer Pattern & Final Take Home

Objectives
To practice on Observer Pattern

Activities
Extend the previous homework as;
When the ‘stop simulation’ command occurs, each object such as engine, tank, and valve prints a message.
Imagine three tanks, three valves, and one engine defined so expected output will be;

```
Engine: Simulation stopped
Tank 1: Simulation stopped
Valve 1: Simulation stopped
Tank 2: Simulation stopped
Valve 2: Simulation stopped
Tank 3: Simulation stopped
Valve 3: Simulation stopped
```

Hint keyword: Observer Pattern

- **Append new function to your command list.**
- **For sum operation all of the given tanks’ valves must be on open state.**
- **Implement Try, Except on necessary places (you chose and explain the reasons in comment line)**
- **Finalize your project. Add a comment line to the beginning of each class, function, design pattern, etc. you use. Let your final submission be in the form of a project ready for presentation.**

The command list;

```
start_engine;
stop_engine;

change_engine_block;
repair_engine;
full_throttle <seconds>;

add_fuel_tank <capacity>;
list_fuel_tanks;
print_fuel_tank_count;
remove_fuel_tank <tank_id>;
connect_fuel_tank_to_engine <tank_id>;
disconnect_fuel_tank_from_engine <tank_id>;
list_connected_tanks;
print_total_fuel_quantity;
print_total_consumed_fuel_quantity;
print_tank_info <tank_id>;
fill_tank <tank_id> <fuel_quantity>;

open_valve <tank_id>;
close_valve <tank_id>;
sum_operator<tank_id, tank_id, tank_id>;

break_fuel_tank <tank_id>;
repair_fuel_tank <tank_id>;

wait <seconds>;
stop_simulation;
```

Task List;

1. Draw a UML diagram about the system.
2. Implement the classes. The classes need to include possible attributes and methods.
3. Simulate the system with several input files not only given example input file.

Problem Solving Tips

1. UML and source code has to match
2. Do not implement logic in Main. Do it in Class, which is responsible.
3. Create your own input files to test every aspects of your program.