Take Home — OOP Practise

Classes & UML Design

Objectives

To practice on UML
To practice on Class, its attributes, and methods
To practice the Single Responsibility Principle and Singleton Pattern

Activities

The users will provide an input and output file from the command line. Write a program that reads commands from the input file and prints output to the output file.

The input file contains the basic commands.

The command list;

start_engine;
stop_engine;
~~**absorb_fuel <quantity>;**~~
give_back_fuel <quantity>;

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>;

break_fuel_tank <tank_id>;
repair_fuel_tank <tank_id>;

wait <seconds>;

stop_simulation;

- The program must run until it takes a “stop_simulation;” command.
- There is only one engine. The engine’s attributes are;
  - fuel_per_second: double // **it will always be 5.5**
  - status: boolean // **true means running**
- The engine has an internal tank to store fuel. **Internal tank capacity will be 55.0**
- **There is no max tank count (Unlimited)**
- **Each command takes 1 second. So, after the command is executed, the engine consumes some fuel if it is running.**
- **The wait command consumes fuel within seconds if the engine is running.**
- **print_tank_info command prints all information about the selected tank.**
- **The engine absorbs fuel from a connected tank when the internal tank capacity exceeds 20.0. The connected tank is selected randomly, and another tank will be chosen if there is insufficient fuel.**
- **The engine has to return fuel in its internal tank to the connected tanks before it is stopped. The remaining fuel must go to a connected tank with the minimum fuel.**
- There are several fuel tanks. Tank’s attributes are;
  - capacity: double
  - fuel_quantity: double
  - broken: boolean
- The engine needs a minimum of one connected tank to start; otherwise, the engine can not start.
- Each tank has a valve to connect the tanks and the engine.

Task List;

1. Draw a UML diagram of 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 the 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. Have a look at the example input file.

```
start_engine;
add_fuel_tank 100;
add_fuel_tank 150;
add_fuel_tank 100;
add_fuel_tank 250;
add_fuel_tank 100;
fill_tank 1 100;
fill_tank 2 150;
fill_tank 3 100;
connect_fuel_tank_to_engine 1;
connect_fuel_tank_to_engine 2;
connect_fuel_tank_to_engine 3;
connect_fuel_tank_to_engine 4;
remove_fuel_tank 5;
connect_fuel_tank_to_engine 5;
disconnect_fuel_tank_from_engine 4;
give_back_fuel <quantity>;
open_valve 1;
open_valve 2;
fill_tank 1 100;
fill_tank 2 150;
fill_tank 3 100;
start_engine;
wait 5;
list_fuel_tanks;
print_fuel_tank_count;
list_connected_tanks;
print_total_fuel_quantity;
print_total_consumed_fuel_quantity;
print_tank_info 1;
print_tank_info 2;
close_valve <tank_id>;
wait 5;
fill_tank 1 100;
fill_tank 2 150;
fill_tank 3 100;
print_tank_info 1;
print_tank_info 2;
print_tank_info 3;
stop_engine;
print_tank_info 1;
print_tank_info 2;
stop_simulation;
```