# **Petrol Pump Management System**

## **OVERVIEW**

The Petrol Pump Management System is a Python program designed to manage fuel prices and sales at a petrol pump. The program allows users to update fuel prices, sell fuel, and generate sales receipts.

PROGRAM’S COMPONENTS:

1. \*Class Definition\*:

- The program defines a class named PetrolPump. This class represents a petrol pump station.

- Inside the class, there are several methods to perform operations such as loading and saving fuel prices, updating prices, selling fuel, and calculating liters sold.

2. \*Initialization Method\* (\_\_init\_\_):

- The \_\_init\_\_ method serves as the constructor for the class.

- It initializes the prices attribute by calling the load\_prices method. This attribute stores the fuel prices.

3. \*Loading Prices\* (load\_prices method):

- This method tries to load fuel prices from a file named fuel\_prices.json.

- If the file exists, it loads the prices from the file and returns them as a dictionary. If the file doesn't exist, it returns an empty dictionary.

4. \*Saving Prices\* (save\_prices method):

- This method saves the current fuel prices stored in the prices attribute to the fuel\_prices.json file.

5. \*Updating Prices\* (update\_prices method):

- This method allows the user to update the prices of different types of fuel.

- It prompts the user to enter the fuel type and the price per liter.

- It updates the prices attribute with the new price and saves the updated prices to the file.

6. \*Calculating Liters Sold\* (calculate\_liters method):

- This method calculates the number of liters sold based on the amount spent and the price per liter.

- It returns the calculated value, handling the case where the price per liter is zero to avoid division by zero errors.

7. \*Selling Fuel\* (sell\_fuel method):

- This method simulates selling fuel to a customer.

- It prompts the user to enter the fuel type and the amount spent on fuel.

- It calculates the number of liters sold using the calculate\_liters method.

- It creates a receipt data dictionary containing information about the sale.

- It saves the receipt data to a file named sales\_receipt.json.

8.\*Main Program\*:

- In the main part of the program, an instance of the PetrolPump class is created.

- A menu is displayed to the user, allowing them to choose between updating fuel prices, selling fuel, or exiting the program.

- Based on the user's choice, the corresponding method of the PetrolPump class is called to perform the selected operation.

- The program continues running until the user chooses to exit.

## **How It Works**

### **1. Initialization**

* Upon execution, the program initializes a **PetrolPump** object.
* It loads existing fuel prices from a JSON file named **fuel\_prices.json**. If the file doesn't exist, it initializes an empty dictionary.

### **2. Menu Options**

* The program displays a menu with the following options:
  + Update Fuel Prices
  + Sell Fuel
  + Exit

### **3. Updating Fuel Prices**

* Selecting option 1 allows the user to update fuel prices.
* The user is prompted to enter the fuel type and the price per liter.
* The program updates the fuel prices dictionary and saves it to the **fuel\_prices.json** file.

### **4. Selling Fuel**

* Selecting option 2 allows the user to sell fuel.
* The user is prompted to enter the fuel type and the amount spent on fuel.
* The program calculates the number of liters sold based on the entered amount and the corresponding price per liter.
* It then generates a sales receipt containing:
  + Fuel Type
  + Liters Sold
  + Amount Spent
* The receipt is saved as a JSON file named **sales\_receipt.json**.

### **5. Exiting**

* Selecting option 3 terminates the program.

## **Conclusion:**

The Petrol Pump Management System provides a simple yet effective way to manage fuel prices and sales at a petrol pump. It offers convenience to pump operators by automating price updates and generating sales receipts accurately.