Tyler Kocadag

**Analysis**

**Domain:**

* Fleet Management System

**Function Points:**

1. **Actions Performed by Actors:**
   * A user can load fleet data from a CSV file.
   * A user can load fleet data from a serialized file if the CSV file is not provided.
   * A user can display information about the fleet.
   * A user can add a new boat to the fleet.
   * A user can remove a boat from the fleet by name.
   * A user can update a boat’s expenses.
   * A user can save the fleet data to a serialized file upon exiting.
2. **Actions Performed by Non-Actors:**
   * Reading fleet data from a CSV file.
   * Serializing and deserializing fleet data.
   * Parsing CSV lines to create Boat objects.

**Scenarios:**

1. **Loading Fleet Data from CSV File:**
   * When the program starts, it reads fleet data from a CSV file specified as a command-line argument.
   * Each valid line is parsed into a Boat object and added to the fleet.
2. **Menu Interaction:**
   * The user is presented with a menu to perform actions like viewing the fleet, adding a boat, removing a boat, and updating expenses.
3. **Adding a Boat:**
   * The user inputs boat details in CSV format.
   * The system validates and adds the new boat to the fleet.
4. **Removing a Boat:**
   * The user specifies the name of a boat.
   * If the boat exists, it is removed from the fleet.
5. **Saving Fleet Data:**
   * When exiting the program, the fleet is serialized to a file for later use.

**Design**

**Classes and Objects**

**Classes:**

1. **FleetManagementSystem**
   * Manages the entire fleet and user interactions.
2. **Boat**
   * Represents a single boat.
3. **BoatType (Enum)**
   * Represents different types of boats (e.g., SAIL, POWER).

**Data of Objects and Classes**

**FleetManagementSystem Class:**

* **Data Members:**
  + fleet : ArrayList<Boat>
    - Stores all boats in the fleet.
  + keyboard : Scanner
    - Reads user input.
* **Methods:**
  + run(String[] args)
    - Starts the system and handles menu interactions.
  + readFleetFromCSV(String fileName)
    - Reads data from a CSV file and populates the fleet.
  + readFleetFromSerializedFile()
    - Loads fleet data from a serialized file.
  + saveFleetToSerializedFile()
    - Saves the fleet data to a serialized file.
  + printFleet()
    - Displays all boats in the fleet.
  + addBoat()
    - Prompts the user to add a new boat.
  + removeBoat()
    - Prompts the user to remove a boat by name.
  + requestExpense()
    - Updates the expenses for a boat.
  + findBoatByName(String name)
    - Finds a boat by name in the fleet.

**Boat Class:**

* **Data Members:**
  + type : BoatType
    - The type of the boat (e.g., SAIL, POWER).
  + name : String
    - Name of the boat.
  + year : int
    - Year of manufacture.
  + makeModel : String
    - Boat’s make and model.
  + length : double
    - Length of the boat.
  + purchasePrice : double
    - Purchase price of the boat.
  + expenses : double
    - Total expenses incurred for the boat.
* **Methods:**
  + Boat(BoatType type, String name, int year, String makeModel, double length, double purchasePrice)
    - Constructor to initialize a new boat.
  + addExpense(double amount)
    - Adds an expense to the boat.
  + getRemainingExpense()
    - Calculates the remaining allowable expense (if there’s a limit).
  + toString()
    - Returns a string representation of the boat.
  + **Accessors:**
    - getName()
    - getPurchasePrice()
    - getExpenses()

**BoatType Enum:**

* **Values:**
  + SAIL
  + POWER

**UML Class Diagram**

1. **FleetManagementSystem**
   * Data:
     + fleet : ArrayList<Boat>
     + keyboard : Scanner
   * Methods:
     + run(String[] args)
     + readFleetFromCSV(String fileName)
     + readFleetFromSerializedFile()
     + saveFleetToSerializedFile()
     + printFleet()
     + addBoat()
     + removeBoat()
     + requestExpense()
     + findBoatByName(String name)
2. **Boat**
   * Data:
     + type : BoatType
     + name : String
     + year : int
     + makeModel : String
     + length : double
     + purchasePrice : double
     + expenses : double
   * Methods:
     + Boat(...)
     + addExpense(double amount)
     + getRemainingExpense()
     + toString()
     + **Accessors:**
       - getName()
       - getPurchasePrice()
       - getExpenses()
3. **BoatType (Enum)**