Object-Oriented Analysis

Key Concepts:

- **Boat**: Represents an individual boat.
- **Fleet**: A collection of boats with methods to manage them.
- **Fleet Management System**: The interface between the user and the fleet.

Actors:

- **User**: Interacts with the system via a menu to perform operations.

Responsibilities:

- **Boat**:
  - Store boat details.
  - Provide functionality to track expenses and check spending limits.
- **Fleet**:
  - Manage the collection of boats.
  - Provide methods to add/remove boats and track expenses.
  - Generate fleet reports.
- **Fleet Management System**:
  - Handle user interaction.
  - Load and save data.
  - Delegate tasks to the `Fleet` class.

Classes:

1. **Boat**:
   - Attributes:
     - `type`: Type of boat (`SAILING` or `POWER`).
     - `name`: Name of the boat.
     - `year`: Year of manufacture.
     - `make`: Make or model.
     - `length`: Length in feet.
     - `price`: Purchase price.

        - `expenses`: Maintenance expenses.
    - Methods:
        - `addExpense(double amount)`: Adds an expense, ensuring it doesn't exceed the purchase price.
        - `getName()`: Returns the boat's name.
        - `getExpenses()`: Returns the total expenses.
        - `toString()`: Returns a formatted string representation of the boat.

2. **BoatType**:
    - Enum with values: `SAILING`, `POWER`.

3. **Fleet**:
    - Attributes:
        - `fleet`: A collection of boats (e.g., `ArrayList<Boat>`).
    - Methods:
        - `loadCSVFile(String fileName)`: Loads fleet data from a CSV file.
        - `addBoat(String csvString)`: Adds a boat to the fleet.
        - `removeBoat(String name)`: Removes a boat by name.
        - `addExpense(String name, double amount)`: Adds an expense to a specific boat.
        - `boatExists(String name)`: Checks if a boat exists by name.
        - `findName(String name)`: Finds a boat by name.
        - `fleetReport()`: Generates a report of the entire fleet.

4. **FleetManagementSystem**:
    - Attributes:
        - `fleet`: An instance of the `Fleet` class.
        - `keyboard`: Scanner for user input.
        - `DATABASE_FILE`: File path for the serialized database.
    - Methods:
        - `start(String[] args)`: Initializes the system, loads data, and manages the menu loop.
        - `printMenu()`: Prints the menu options.
        - `menuActionItem(char option)`: Executes the user's selected option.
        - `addBoat()`: Adds a new boat to the fleet.
        - `removeBoat()`: Removes a boat from the fleet.

        - `manageExpense()`: Manages expenses for a boat.
        - `loadCSVFile(String fileName)`: Loads fleet data from a CSV file.
        - `loadFromDatabase()`: Loads fleet data from a serialized file.
        - `saveToDatabase()`: Saves fleet data to a serialized file.

Data:

1. **Initialization**:
    - Load data from `FleetData.csv` if provided or from `FleetData.db` otherwise.
2. **User Interaction**:
    - Present a menu of options (`Print`, `Add`, `Remove`, `Expense`, `Exit`).
    - Execute the selected action by invoking methods from the `Fleet` class.
3. **Database**:
    - Save updated fleet data to `FleetData.db` on exit.