CSC 120 Final Project — **Aakash Singh**
Object Oriented Design — 12/01/2025

- Analysis
  - Domain
    - **SingleBoat** – represents an individual boat
    - **FleetOfBoats** – manages the collection of boats
    - **FleetManager** – handles user interaction and program control
    - **BoatType** (enum) – defines boat type (SAILING or POWER)
    - Actors: User – selects menu options, inputs data
  - Function Points
    - User: add/remove boats, record expenses, print fleet report, exit program
    - System: load/save fleet data, compute totals, validate input
  - Scenarios
    - User starts program
    - User chooses to add a new boat
      - Program collects boat data
      - Program creates a SingleBoat object
      - Program stores it inside the FleetOfBoats
    - User chooses to display all boats
      - Program prints FleetOfBoats
    - User chooses to remove a boat
      - Program finds and removes the SingleBoat from FleetOfBoats
  - Design
    - Classes and Objects
      - **SingleBoat** – object representing a boat
      - **FleetOfBoats** – object managing the collection of boats
      - **FleetManager** – object controlling program flow and user input
      - **BoatType** – enum for boat types
  - Data of Objects and Classes
    - **SingleBoat**
      - Object data (per boat):
        - name: String
        - length: double
        - price: double
    - **FleetOfBoats**
      - Object data:
        - Boats: ArrayList <Boat>
  - Methods of Objects and Classes
    - **SingleBoat**
      - addExpense(amount) – adds an expense to the boat's total expenses
      - remainingBudget() – calculates and returns the remaining budget for the boat
      - Getters – return the values of the boat's fields (name, type, year, make/model, length, purchase price, expenses)

- toString() – returns a formatted string representation of the boat.
- **FleetOfBoats:**
  - addBoat(boat) – adds a SingleBoat object to the fleet
  - removeBoat(name) – removes a boat from the fleet by name
  - findBoat(name) – searches for and returns a boat matching the given name
  - getBoats() – returns the underlying collection/array/list of boats
- **FleetManager:**
  - main() – entry point; starts the program and menu loop
  - getMenuOption() – displays the menu and reads the user's choice
  - addBoatMenu() – collects input from the user and creates a new boat for the fleet
  - removeBoatMenu() – prompts the user for a boat name and removes the matching boat
  - spendOnBoatMenu() – prompts the user for an expense amount and applies it to the selected boat
  - printFleetReport() – outputs all boats and their details
  - loadCSVData() – loads fleet data from a CSV file
  - loadSerializedData() – loads fleet data from a serialized file
  - saveSerializedData() – saves the fleet data to a serialized file before exiting
- **BoatType (enum):**
  - Enum values: SAILING, POWER

- Overall control
  - FleetManager controls the full program flow using imperative techniques:
    - A menu loop repeatedly displays options and reads user input
    - if/else (or switch) statements determine which action the program performs
    - FleetManager calls methods on FleetOfBoats to add, remove, find, or list boats
    - FleetManager calls methods on SingleBoat to update expenses and retrieve boat information
    - Program begins by loading data (CSV or serialized) and ends by saving data before exit