CSC 120 – S

HYUN KIM

FLEET MANAGEMENT – OBJECT ORIENTENTED ANALYSIS & DESIGN

ANALYSIS

Domain:

- Boat (Non-actor)
- Fleet (Actor)

Function Points:

- Boat: No action
- Fleet: Adds a boat, removes a boat, finds for a boat, authorizes expense for a boat

Scenario:

- Add a boat to the fleet in csv format through the csv or database file
- Remove a boat from the fleet by its name if the boat exists in the fleet
- Search for a boat in the fleet by its name and return its location in the fleet
- Request to spend expense on a boat:
  - Authorize request if the total expense spent on the boat does not exceed its price
  - Decline request if the total expense spent on the boat exceeds its price

DESIGN

Boat Class:

- Data
  - BoatType: enum, non-final, private, immutable, non-accessible
  - type: BoatType, non-final, private, immutable, accessible
  - name: String, non-final, private, immutable, accessible
  - model: String, non-final, private, immutable, accessible
  - year: int, non-final, private, immutable, accessible, cannot be negative or beyond this year (2022)
  - length: int, non-final, private, immutable, accessible, cannot exceed 100
  - price: double, non-final, private, immutable, accessible, cannot exceed 1 million
  - expense: double, non-final, private, mutable, accessible, cannot exceed price of the boat
- Method
  - Constructor: initializes boatType, name, model, year, length, price based on csv or database data and initializes expense to 0
  - getName: returns the name of the boat
  - getPrice: returns the price of the boat
  - getExpense: returns the expense of the boat
  - setExpense: sets the expense of the boat

- toString: returns the boat data in string representation

Fleet Class:

- Data
  - boatList: ArrayList<boat>, non-final, private, mutable, non-accessible
  - fleetPrice: double, non-final, private, mutable, accessible
  - fleetExpense: double, non-final, private, mutable, accessible
- Method
  - Constructor: creates a new ArrayList of boat objects called boatList, initializes fleetPrice and fleetExpense to 0
  - addBoat: adds a boat to the boatList
  - removeBoat: removes a boat from the boatList
  - searchBoat: searches for a boat in the boatList and returns the boat's location index
  - authorizeExpense: authorizes expense for a boat in the boatList
  - toString: returns the fleet data in string representation