Hub & Transport System — UML Class Diagram

BasicContainer
- ID { get; } : int
- weight { get; } : double

+ consumption(): double // 2.5 * weight

<<Abstract>>
Container
- ID { get; } : int
- weight { get; } : double

+ Container(ID, weight)
+ consumption(): double
+ equals(other: Container): bool

<<interface>>
ITransport<THub> : IHub
- ID { get; } : int
- fuel { get; set; } : double
- weight { get; } : double
- maxRangeKm { get; } : double

+ moveTo(IHub dest) : bool

<<interface>>
IHub
- ID { get; } : int
- latitude { get; } : double
- longitude { get; } : double

+ distanceTo(IHub other) : double

<<interface>>
IAirport
- maxGates { get; } : int
- current { get; } List~IPlane~

+ incomingPlane(IPlane) : void
+ outgoingPlane(IPlane) : void

HeavyContainer
- ID { get; } : int
- weight { get; } : double

+ heavyContainer(int id, double weight) : —
+ consumption(): double // 3.0 * weight

<<interface>>
IShip
- currentContainers { get; } : List~Container~

+ weightTotalCalc(List<Container> currentContainers) : double
+ ReFuel(double newFuel) : void
+ load(cont: Container) : bool
+ unload(cont: Container) : bool

<<interface>>
IPlane
+ weightTotalCalc(List<Container> currentContainers) : double
+ ReFuel(double newFuel) : void
+ load(cont: Container) : bool
+ unload(cont: Container) : bool

RefrigeratedContainer
- ID { get; } : int
- weight { get; } : double

+ refrigeratedContainer(int id, double weight) : —
+ consumption(): double // 5.0 * weight

<<interface>>
IPort
- current { get; } List~IShip~
- containers { get; } List~Container~

+incomingShip(IShip) : void
+outgoingShip(IShip) : void

Port
- ID { get; } : int
- latitude { get; } : double
- longitude { get; } : double
- containers { get; } : List~Container~
- history { get; set; } : List~IShip~
- current { get; set; } : List~IShip~

+ Port(int id, double lat, double lon, List<Container> containers) : —
+ incomingShip(IShip) : void
+ outgoingShip(IShip) : void
+ distanceTo(IHub other) : double

LiquidContainer
- ID { get; } : int
- weight { get; } : double

+ liquidContainer(int id, double weight) : —
+ consumption(): double // 4.0 * weight

Airport
- ID { get; } : int
- latitude { get; } : double
- longitude { get; } : double
- maxGates { get; } : int
- containers { get; } : List~Container~
- history { get; } : List<IPlane>
- current { get; } : List<IPlane>

+ Airport(int id, double lat, double lon, int maxGates, List<Container> containers) : —
+ incomingPlane(IPlane) : void
+ outgoingPlane(IPlane) : void
+ hasFreeGate() : bool
+ distanceTo(IHub other) : double

Plane
- ID { get; } : int
- fuel { get; set; } : double
- currentAirport { get; set; }: IAirport?
- maxRangeKm { get; } : double
- weight { get; } : double
- maxNumberOfAllContainers { get; } : int
- maxNumberOfHeavyContainers { get; } : int
- maxNumberOfBasicContainers { get; } : int
- maxNumberOfRefrigeratedContainers { get; } : int
- maxNumberOfLiquidContainers { get; } : int
- fuelConsumptionPerKM { get; } : double
- currentContainers { get; } : List~Container~

+ Plane(int id, double weight, double initialFuel, int maxAll, int maxHeavy, int maxBasic, int maxRefrigerated, int maxLiquid, double maxRangeKm, IAirport? currentAirport) : —
+ reFuel(double newFuel) : void
+ weightTotalCalc(List<Container> currentContainers) : double
+ checkingForPermQuantityContainers(Container cont, List<Container> currentContainers) : bool
+ load(Container cargo) : bool
+ unload(Container cargo) : bool
+ moveTo(IAirport dest) : bool

Ship
- ID { get; } : int
- fuel { get; set; } : double
- currentPort { get; set; }: IPort?
- weight { get; } : double
- maxRangeKm { get; } : double
- maxNumberOfAllContainers { get; } : int
- maxNumberOfHeavyContainers { get; } : int
- maxNumberOfRefrigeratedContainers { get; } : int
- maxNumberOfLiquidContainers { get; } : int
- fuelConsumptionPerKM { get; } : double
- currentContainers { get; } : List~Container~

+ Ship(int id, double initialFuel, int weight, int maxAll, int maxHeavy, int maxRefrigerated, int maxLiquid, double fuelConsumptionPerKm, IPort? currentPort, double maxRangeKm) : —
+ getCurrentContainers(): List~Container~
+ checkingForPermQuantityContainers(Container cont, List<Container> currentContainers) : bool
+ reFuel(newFuel: double)
+ load(cont: Container): bool
+ unLoad(cont: Container): bool
+ weightTotalCalc(List<Container> currentContainers) : double
+ moveTo(IPort dest) : bool

Program
- ports : Dictionary<int, Port>
- ships : Dictionary<int, Ship>
- cons : Dictionary<int, Container>
- shipMaxWeight : Dictionary<int, int>
- nextPortId : int
- nextConId : int
- nextShipId : int

+ Main(string[] args) : void
+ createPort(JsonObject o) : void
+ createCon(JsonObject o) : void
+ createShip(JsonObject o) : void
+ nextShipId() : int
+ load(JsonObject o) : void
+ unload(JsonObject o) : void
+ sailTo(JsonObject o) : void
+ refuel(JsonObject o) : void
+ buildOut() : string
+ toJsonArray(IEnumerable<int> values) : JsonArray

0..1   1   *   0..1   1   1