Archil Nadaraia

Big Assignment – Second part Documentation

4 Orion Expedition

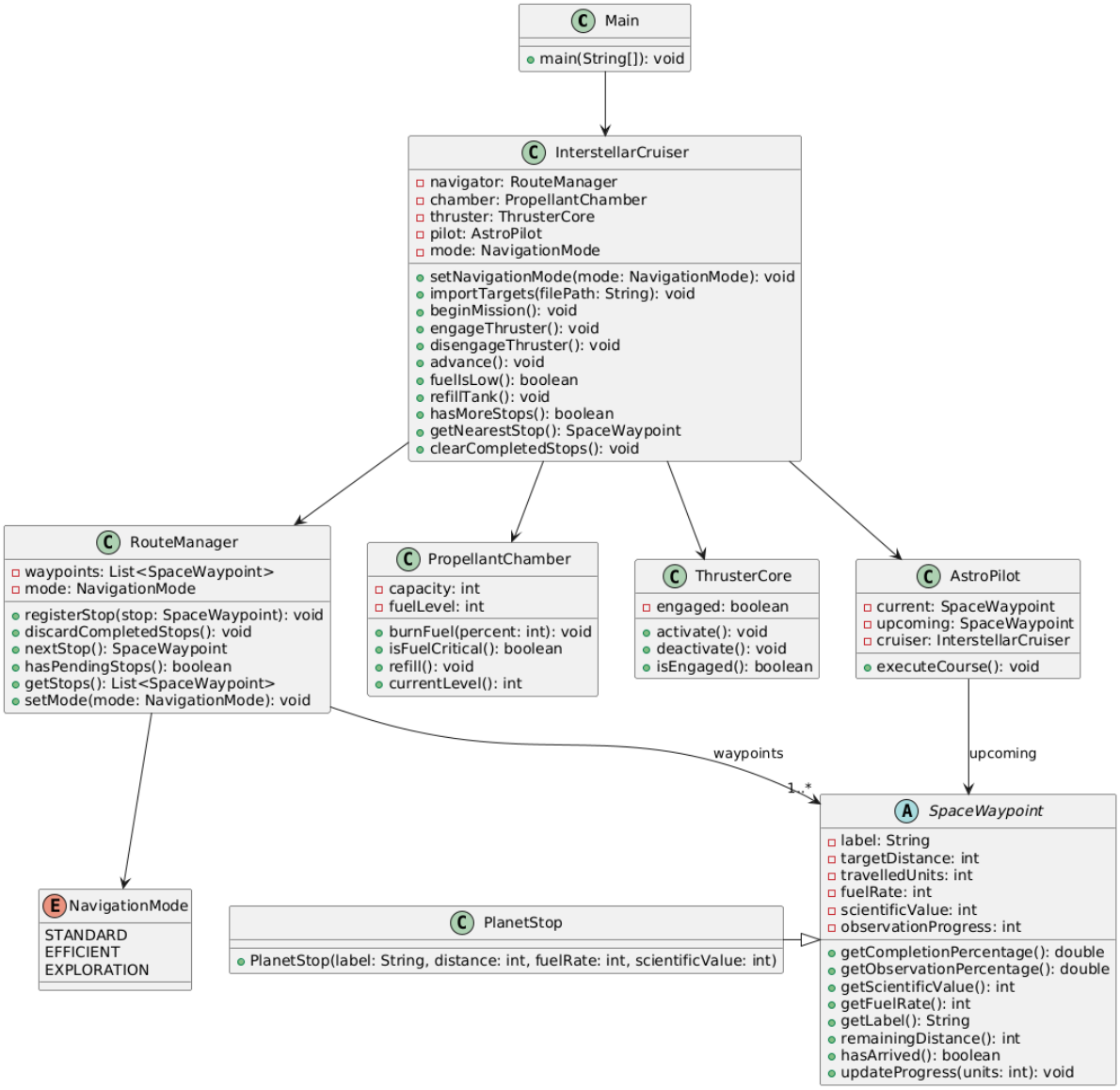
4.1 Additional functionality As the Orion space shuttle ventures deeper into the cosmos, NASA has developed advanced systems to enhance its exploration capabilities. Each celestial region now includes a scientific value metric, observation completion percentage, and specific fuel consumption rate. The navigation system now features a navigation mode with three distinct strategies to interstellar travel. Standard mode takes the direct approach found in the original design. Efficient mode factors fuel consumption into routing decisions by calculating a fuel-to-distance ratio for each destination. Exploration mode chooses destinations with the highest value, sometimes choosing longer paths deliberately to study cosmic phenomena of interest.

4.2 State diagram The shuttle's operational behavior is now governed by a state model that modifies its functionality. Initially, the shuttle is in the docked state, where it can refuel but cannot move as the engine remains inactive. When the engine starts, it transitions to cruising state, traveling at standard velocity by consuming fuel per move and increasing travel progress by 5%. Upon reaching a destination, the shuttle enters orbiting state, performing observations with reduced fuel consumption at 3%. If fuel levels drop critically low, the shuttle transitions to emergency state, where it stops the observation and attempts to refuel. If refueling succeeds, the shuttle returns to the cruising state; if unsuccessful, the engine stops, and shuttle returns to docked state.

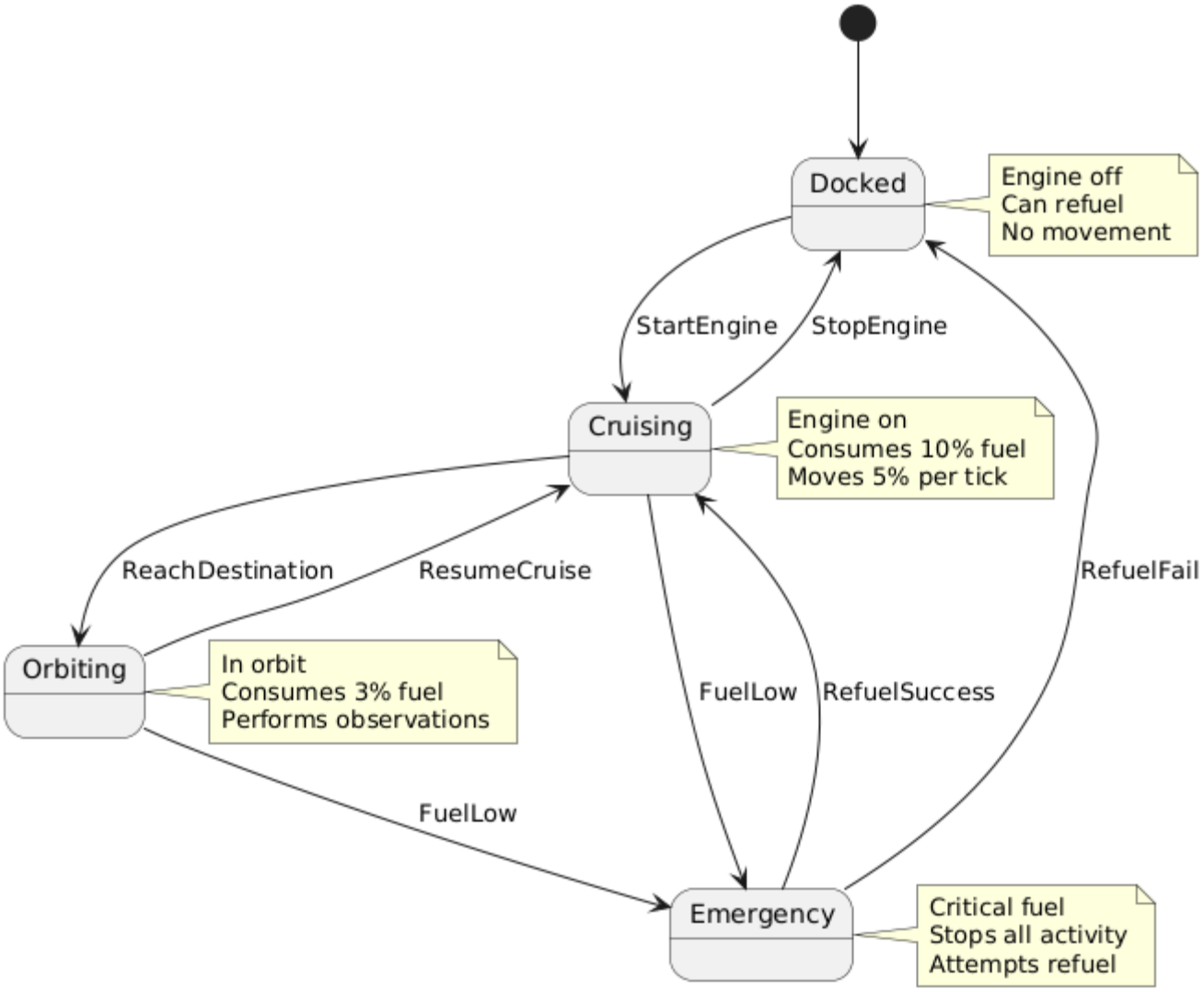
4.3 Scenarios to model with sequence diagrams 4.3.1 Scenario 1: Successful Planetary Visit When the shuttle goes to a new destination, the navigation course begins with the selected destination. The engine starts, and the shuttle enters cruising state. As the shuttle advances, its movement gradually increases the destination's travel progress until it reaches 100%. Once complete, the current destination is updated, the destination is removed from the navigation system, and the closest destination becomes the next target using the active navigation strategy.

4.3.2 Scenario 2: Fuel Emergency Management When the fuel drops below 15% of capacity, the shuttle enters Emergency state. The system attempts refueling with a 50% success chance. If successful, the fuel is restored to maximum, and the mission continues. If unsuccessful, the engine stops, halting progress until the situation is resolved. The shuttle can implement an enhanced emergency protocol that enables solar energy collection in deep space, modifying the random refueling chance based on proximity to stars, or calculating a minimum-energy return trajectory to the nearest known refueling point by selecting the closest destination with refueling capabilities.

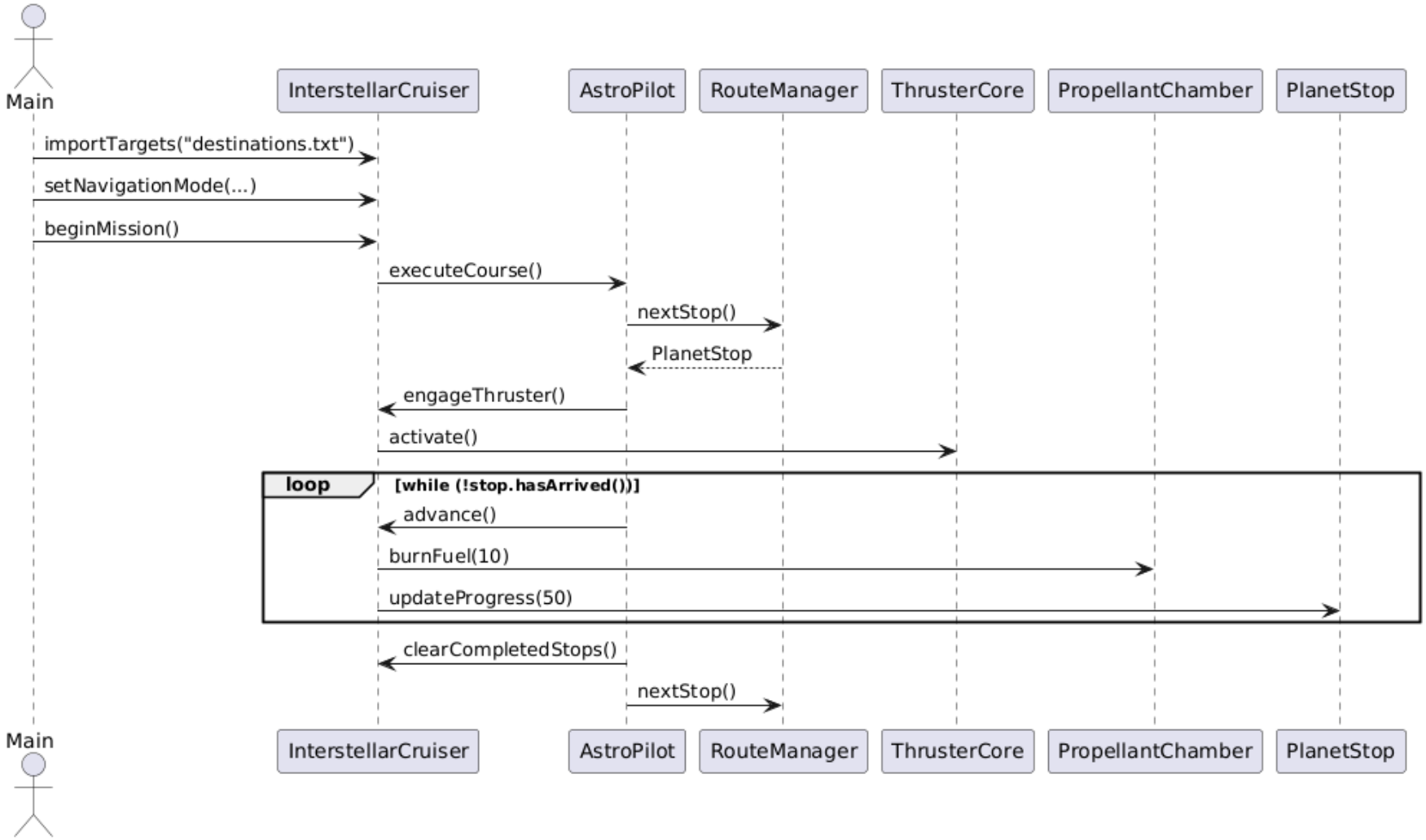
UML Diagram:


Main
+ main(String[]): void
InterstellarCruiser
- navigator: RouteManager
- chamber: PropellantChamber
- thruster: ThrusterCore
- pilot: AstroPilot
- mode: NavigationMode
+ setNavigationMode(mode: NavigationMode): void
+ importTargets(filePath: String): void
+ beginMission(): void
+ engageThruster(): void
+ disengageThruster(): void
+ advance(): void
+ fuelIsLow(): boolean
+ refillTank(): void
+ hasMoreStops(): boolean
+ getNearestStop(): SpaceWaypoint
+ clearCompletedStops(): void
RouteManager
- waypoints: List<SpaceWaypoint>
- mode: NavigationMode
+ registerStop(stop: SpaceWaypoint): void
+ discardCompletedStops(): void
+ nextStop(): SpaceWaypoint
+ hasPendingStops(): boolean
+ getStops(): List<SpaceWaypoint>
+ setMode(mode: NavigationMode): void
PropellantChamber
- capacity: int
- fuelLevel: int
+ burnFuel(percent: int): void
+ isFuelCritical(): boolean
+ refill(): void
+ currentLevel(): int
ThrusterCore
- engaged: boolean
+ activate(): void
+ deactivate(): void
+ isEngaged(): boolean
AstroPilot
- current: SpaceWaypoint
- upcoming: SpaceWaypoint
- cruiser: InterstellarCruiser
+ executeCourse(): void
waypoints
1..*
upcoming
NavigationMode
STANDARD
EFFICIENT
EXPLORATION
PlanetStop
+ PlanetStop(label: String, distance: int, fuelRate: int, scientificValue: int)
SpaceWaypoint
- label: String
- targetDistance: int
- travelledUnits: int
- fuelRate: int
- scientificValue: int
- observationProgress: int
+ getCompletionPercentage(): double
+ getObservationPercentage(): double
+ getScientificValue(): int
+ getFuelRate(): int
+ getLabel(): String
+ remainingDistance(): int
+ hasArrived(): boolean
+ updateProgress(units: int): void

State Diagram:



Sequence Diagram for Scenario 1: Successful Planetary Visit



Sequence Diagram for Scenario 2: Fuel Emergency Management