12/20/2013

Asteroid Exploration System

Design Document

Fanxing Meng
REVIEWER: JAYAPRAKASH GANTA

Table of Contents

Introduction

This document designs a modular system for managing a system of robotic spacecraft for the exploration and mining of asteroids. The range of tasks include discovering precious metals from the inner belt asteroids and comets to support the development of a moon base.

Overview

Discoveries show that asteroids are rich in mineral resources, and potentially water and oxygen for comets. Due to advancements in rocket engine, fuel, and guidance development, we are now capable of sending spacecraft to the asteroid belt for exploration and exploitation. The asteroid belt lies between Mars and Jupiter and contains asteroids as large as Ceres and Vesta. These large asteroids may have layers of ice and water over their rocky core, providing sufficient material and monetary resources themselves to support an entire extraterrestrial construction project. More importantly, their weak gravity field allows low fuel consumption for transporting out large quantities of cargo freight.

Requirements

The exploration system need to manage three major components: robotic spacecraft, asteroids, and mission resources. An additional authentication service will manage access permissions and user credentials to all three components. All methods will need authentication for proper access permissions.

A user interface will show the status of all three components. This component will support logging in and out, define missions, monitor and update mission/spacecraft/asteroid status, monitor mission resources and systems, monitor incoming spacecraft message.

Mission control center manages missions, which contains id, name, purpose, spacecraft id, destination asteroid, status, launch date and estimated time of arrival.

The center controls resources including fuel, spacecraft fleet, and operation budget, together with ground-based systems including communication link status and automated control system status. Corresponding resources will be updated upon creating a mission.

The spacecraft management system should keep track of all spacecraft information, add new spacecraft if resources permit, update spacecraft status, list existing spacecraft, and also look up spacecraft by id.

A spacecraft is dedicated to a mission, has an id, type (explorer or miner), location (AUs from sun), destination (target asteroid id), launch date, and its status including fuel, state, guidance systems and communication link systems.

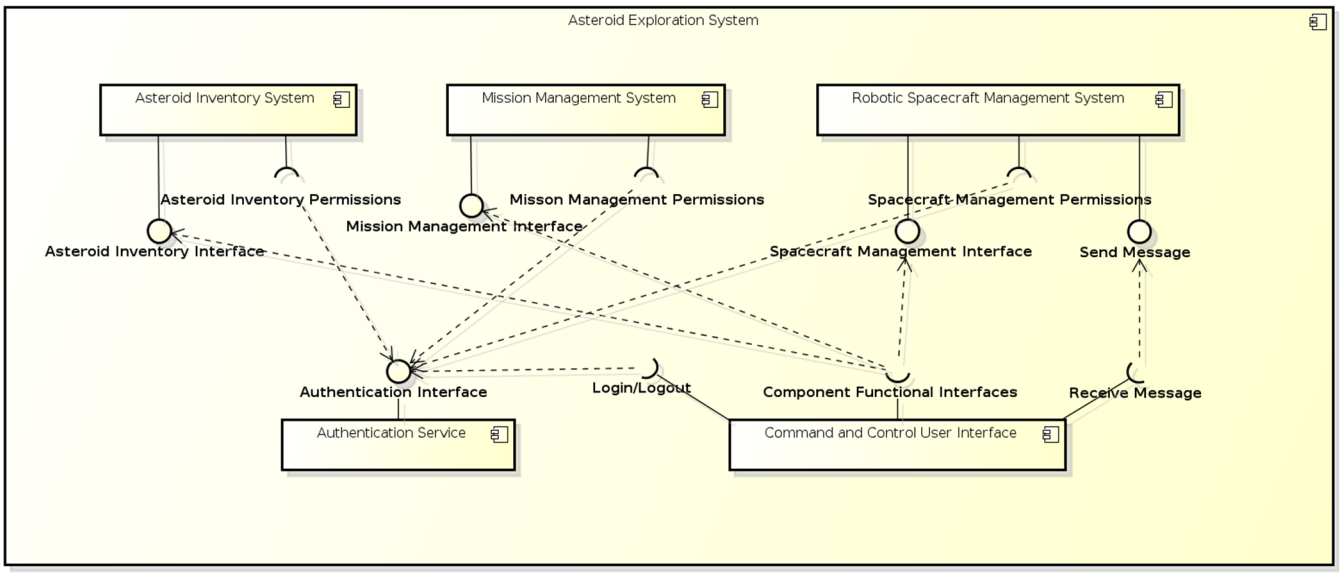
A spacecraft can send message back to the mission control center notifying either a status update, a failure alert, or a discovery message of either mineral, water, or life on an asteroid. The message is then used to update the spacecraft's status, mission state, or asteroid information.

The asteroid inventory keeps the following information for each asteroid: identifier, multiple notes, exploration status of mineral, water, and life, asteroid type (C, M, S, or comet), size, mass, surface gravity, aphelion and perihelion. The inventory will also provide access for listing asteroids, looking up asteroid by id, query for asteroids by note text, and create/update asteroids.

System Architecture

This section defines the System Architecture for the asteroid exploration system.

The following component diagram shows the primary system level components of the asteroid exploration system.



Asteroid Inventory System

The asteroid inventory system is responsible for managing the asteroids in file, the functions for creating asteroid, adding notes, adding discovery, list, lookup, and search asteroids all require the authentication service to check for a token's permission before performing these tasks. The system will also notify the command and control center if any of these information is updated. It also keeps connection to an asteroid schema in the database, which is updated in accordance with the system.

Robotic Spacecraft Management System

The spacecraft management system manages spacecraft and generates messages. Creating, updating, listing, and searching spacecraft, requires authentication service to check for valid permissions. It also notifies the control center and record to database any updates on spacecraft.

Mission Management System

The mission management system is capable of creating missions, listing missions, and look up mission by id. The system also provides methods for checking and setting mission resources and ground based systems. Any change to these data will result in checking for authorization and corresponding changes will be written in the database.

Command and Control User Interface

The command and control center provides user interface for logging in and out in conjunction with the authentication service's token management capability. After logging in, the system then provides another screen on which the user could choose to perform functions provided by

other systems and manually enter parameters in text fields or choose from drop-down lists. Upon clicking corresponding buttons, the interface will call the specified subsystem's function, passing in the user's current token, and completes the task. Display information is updated as soon as the subsystem's value is updated using observer pattern.

To perform the mediator's role better, the message processing utility is also placed in the command and control center to call update methods in corresponding subsystems upon receiving message concerning discoveries and updates.

Authentication Service

The authentication service manages users with credentials and tokens, services, and entitlements including roles and permissions. It provides interface to users and other subsystems for logging in, logging out, changing password, and checking token permissions.

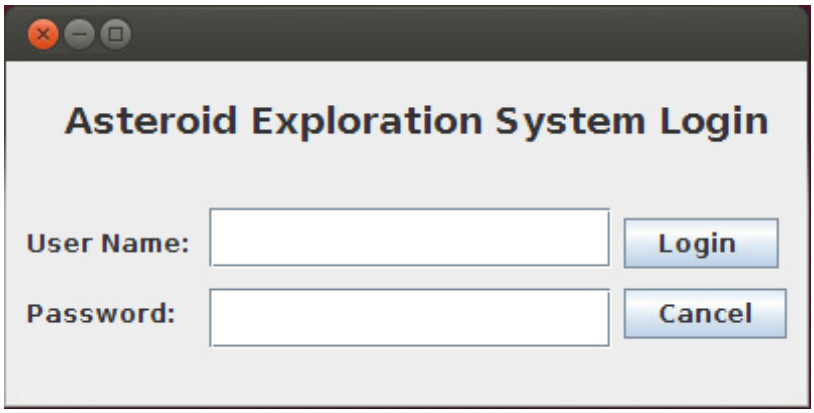
Persistence

Persistence of data may take various approaches. JDBC is the most common way to connect to remote databases, if provided with username and password. A yet simpler way is to have the subsystems implement serializable interface and write the aggregate object to local disk, and recover these data upon running the program.

User Interface

Login Screen

The login screen has two text fields for entering the username and password. If the username/password is incorrect, a warning message will appear at the bottom. If logging in is successful and a token is generated, the user will see the login screen vanish and the main screen will pop up.



Main Screen

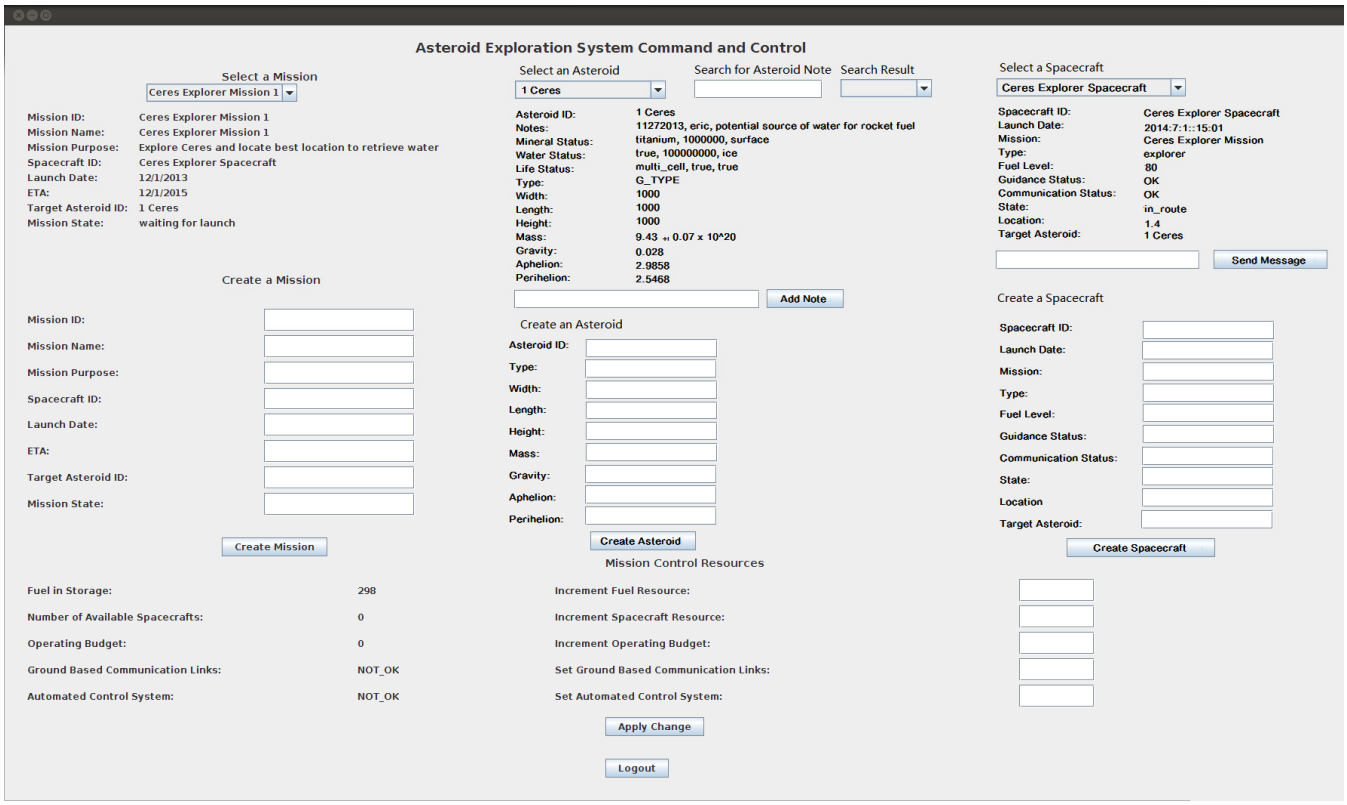
There will be four main panels on the main screen: the top three columns shows mission management system, asteroid inventory, and spacecraft management system.

For each of these systems, only a creation button is provided for the administrator since status updates are solely conducted via the hidden message processor. For users, the drop-down list at the top, together with a search field for asteroids provide interface for listing and looking up objects in each subsystem. For asteroids, there is an Add Note button for adding notes to the asteroid being displayed. For spacecraft, there is a Send Message button where one can simulate a spacecraft and sends back to the control center a message with the correct format. (For testing purposes only)

All displays are refreshed if the certain subsystem receives an update, including creating of object or purely modifications to status.

On the bottom row is the mission resource panel, which shows remaining resources and system status. An administrator can add fuel, budget and spacecraft resources and set system status.

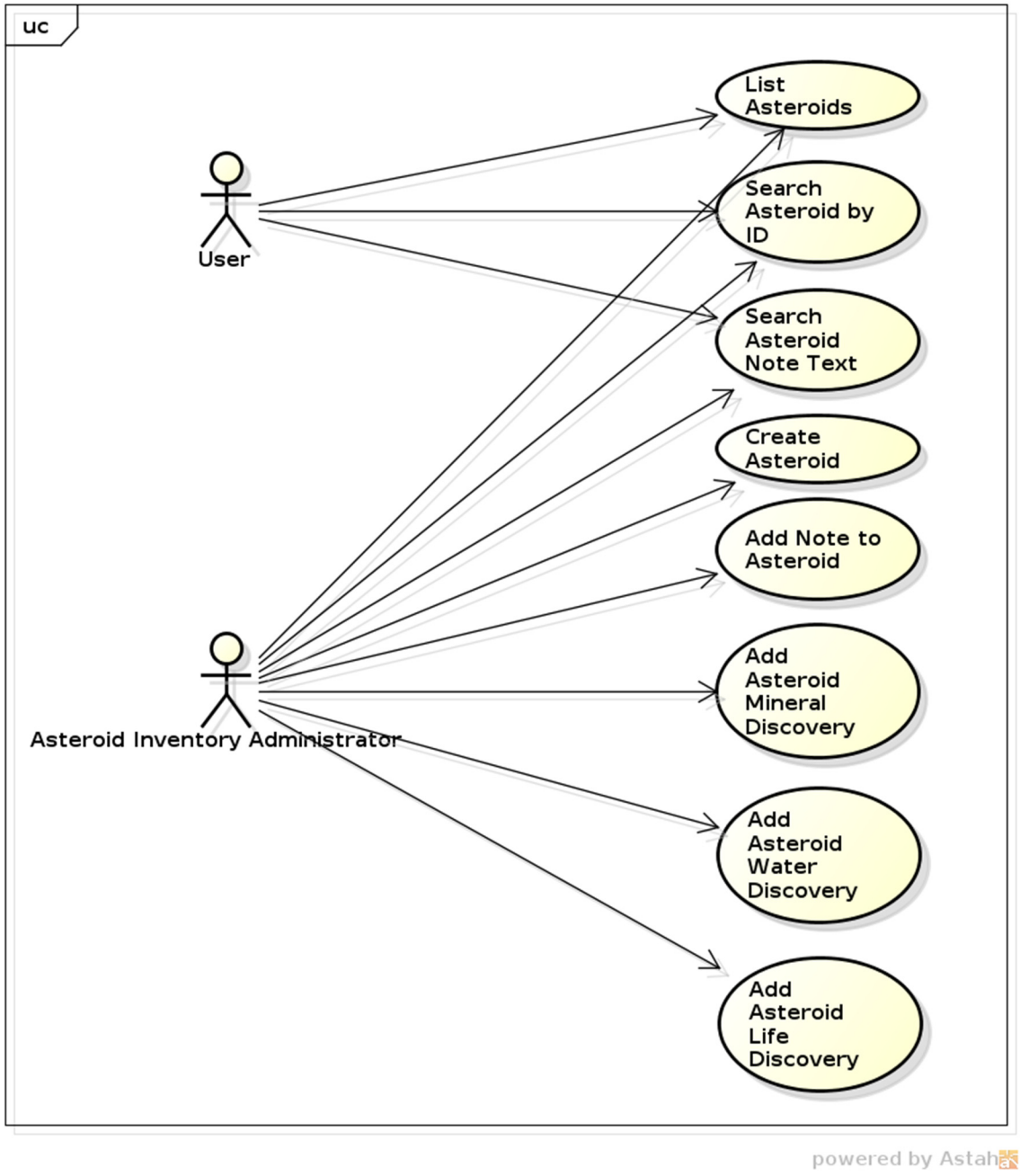
Finally there is a logout button. When a user clicks on that button, the interface calls authentication service to log the user off, closes the current window, and launch again the login screen.


Asteroid Exploration System Command and Control
Select a Mission
Ceres Explorer Mission 1
Mission ID: Ceres Explorer Mission 1
Mission Name: Ceres Explorer Mission 1
Mission Purpose: Explore Ceres and locate best location to retrieve water
Spacecraft ID: Ceres Explorer Spacecraft
Launch Date: 12/1/2013
ETA: 12/1/2015
Target Asteroid ID: 1 Ceres
Mission State: waiting for launch
Create a Mission
Mission ID:
Mission Name:
Mission Purpose:
Spacecraft ID:
Launch Date:
ETA:
Target Asteroid ID:
Mission State:
Create Mission
Select an Asteroid
1 Ceres
Search for Asteroid Note
Search Result
Asteroid ID: 1 Ceres
Notes: 11272013, eric, potential source of water for rocket fuel
Mineral Status: titanium, 1000000, surface
Water Status: true, 100000000, ice
Life Status: multi_cell, true, true
Type: G_TYPE
Width: 1000
Length: 1000
Height: 1000
Mass: 9.43 +/- 0.07 x 10^20
Gravity: 0.028
Aphelion: 2.9858
Perihelion: 2.5468
Add Note
Create an Asteroid
Asteroid ID:
Type:
Width:
Length:
Height:
Mass:
Gravity:
Aphelion:
Perihelion:
Create Asteroid
Mission Control Resources
Fuel in Storage: 298
Number of Available Spacecrafts: 0
Operating Budget: 0
Ground Based Communication Links: NOT_OK
Automated Control System: NOT_OK
Increment Fuel Resource:
Increment Spacecraft Resource:
Increment Operating Budget:
Set Ground Based Communication Links:
Set Automated Control System:
Apply Change
Logout
Select a Spacecraft
Ceres Explorer Spacecraft
Spacecraft ID: Ceres Explorer Spacecraft
Launch Date: 2014:7:1::15:01
Mission: Ceres Explorer Mission
Type: explorer
Fuel Level: 80
Guidance Status: OK
Communication Status: OK
State: in_route
Location: 1.4
Target Asteroid: 1 Ceres
Send Message
Create a Spacecraft
Spacecraft ID:
Launch Date:
Mission:
Type:
Fuel Level:
Guidance Status:
Communication Status:
State:
Location
Target Asteroid:
Create Spacecraft

Use Cases

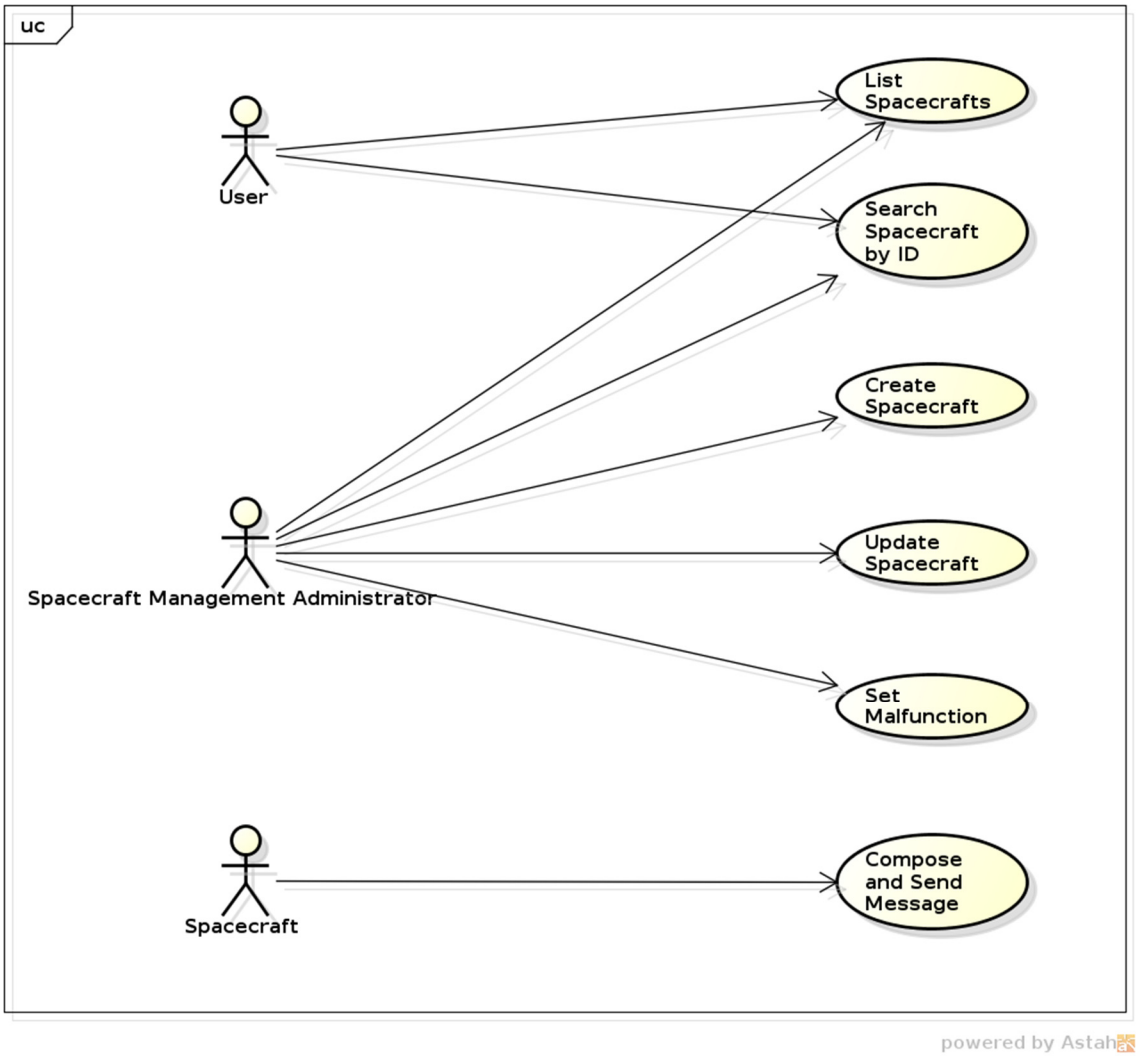
This design supports the following use cases:

Asteroid Inventory System



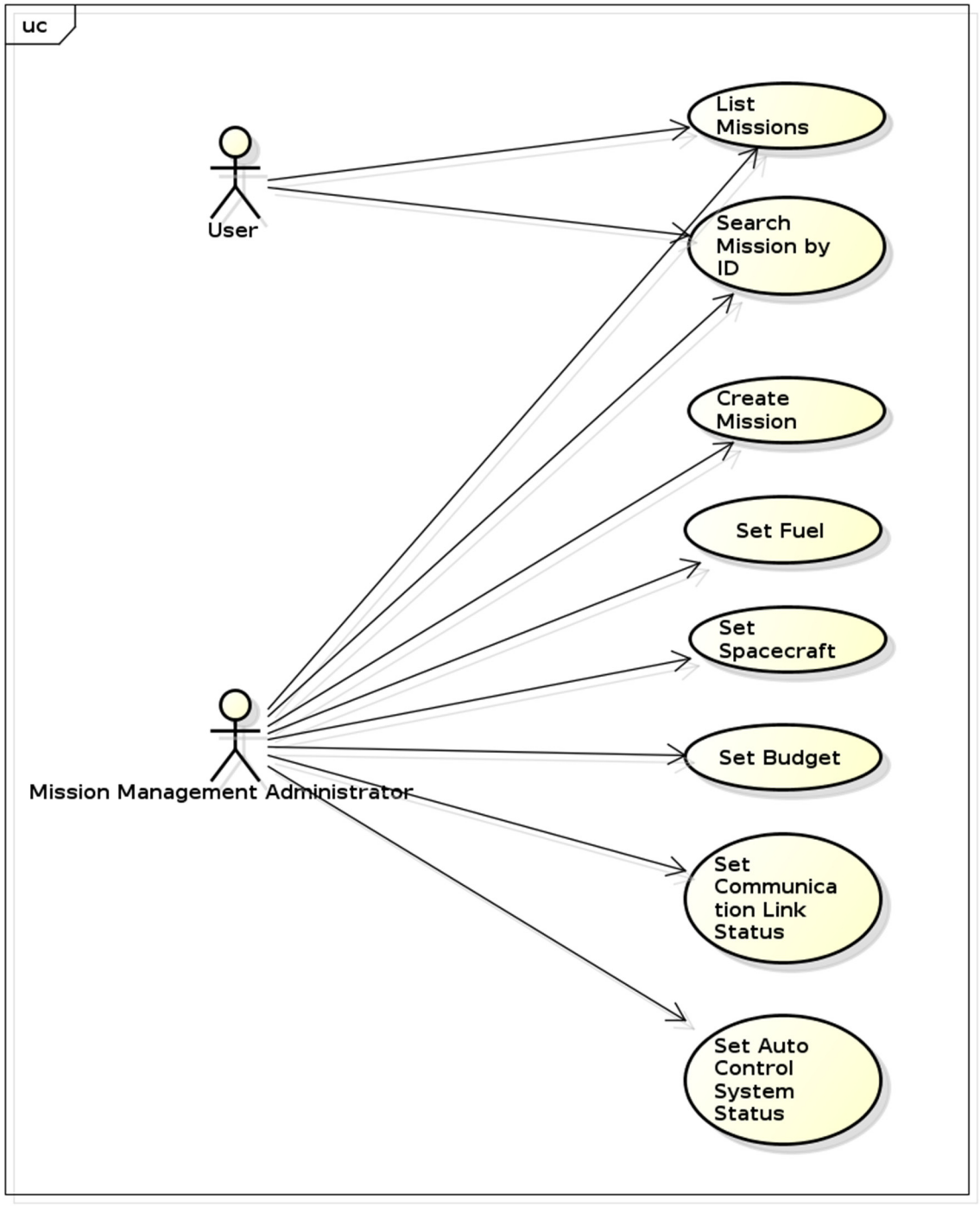
There are two classifications of users. An ordinary user may only browse and search for asteroids, while an administrator has all permissions ranging from creating asteroids to modifying its attributes. Their credentials will be checked upon performing the specified task.

Robotic Spacecraft Management System



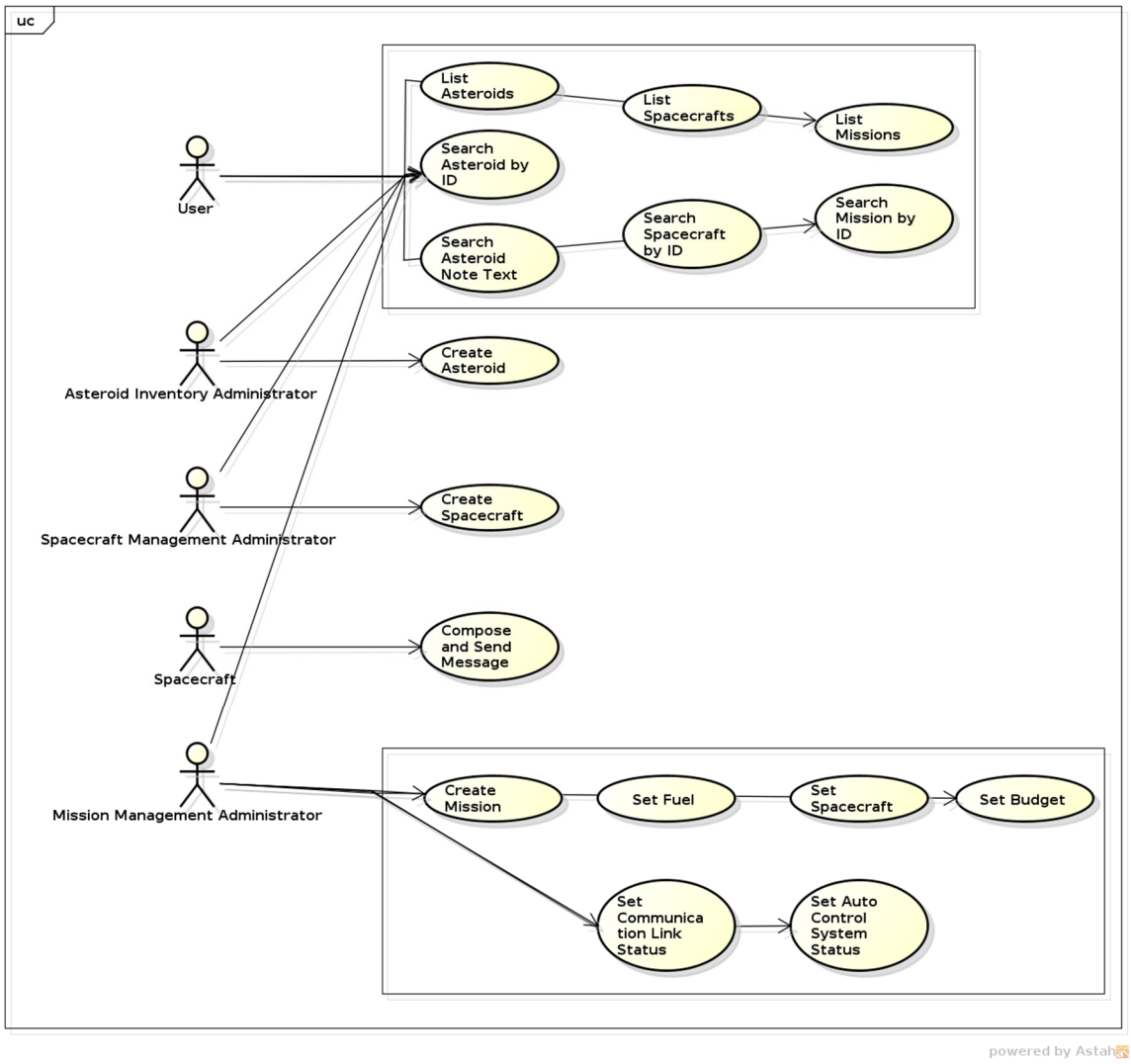
There are three types of users to the spacecraft management system: ordinary users can only list and look up spacecraft, while an administrator can create and manually update spacecraft status. The spacecraft itself will be responsible for converting its findings to a message in the form of a string and sends it to the command and control center for processing.

Mission Management System



There are two types of users in this scenario: ordinary user can list and search missions, while an administrator can create missions and update resources and subsystem status.

Command and Control User Interface

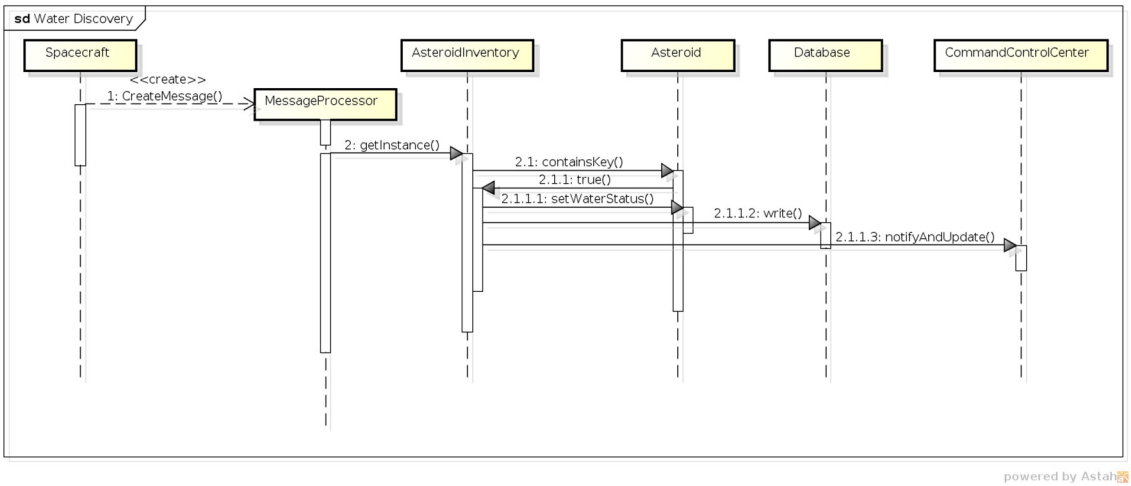


All types of users appears to access the command and control center. All human users may list and search existing facilities and states, while interfaces for updating asteroid and spacecraft information is now hiding from the users and will be accessible only by the system upon receiving spacecraft message, using façade pattern.

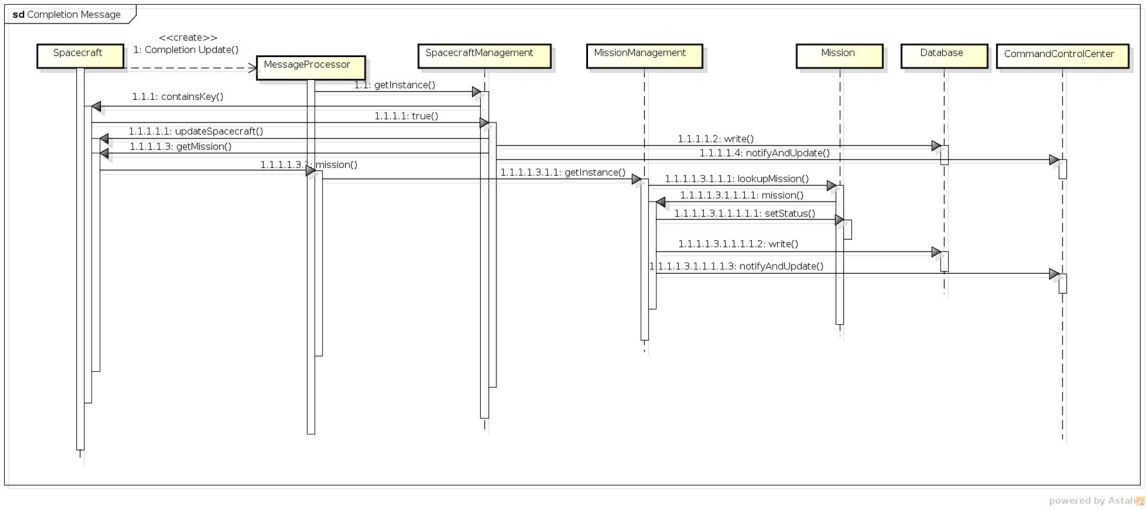
Sequence Diagram

The following sequence diagrams describes three sample workflow for the system upon receiving tasks originated by spacecraft and human users.

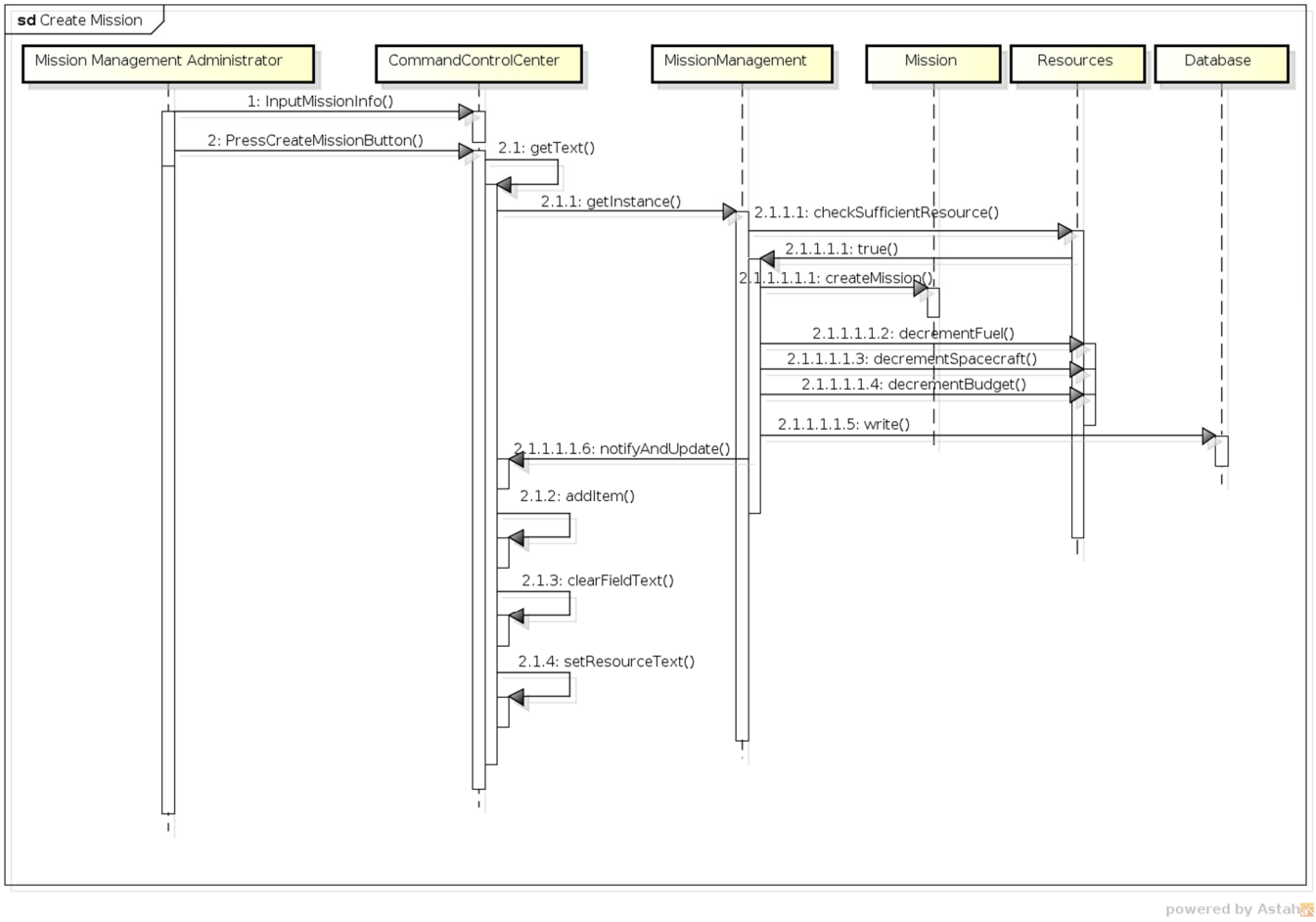
1. The message flow for receiving status message from a spacecraft where spacecraft has discovered water on target asteroid. The message is first decoded by the message processor, where the processor calls the asteroid inventory to check for the specified asteroid to be updated. If it is present, the inventory calls the setWaterStatus of the asteroid and writes the updated information to the database, and calls the command and control center to update the asteroid's data currently displaying.



2. The message flow for receiving a status message where the spacecraft has completed its mission. The message tells the processor to invoke the spacecraft management service where it retrieves the spacecraft that sent the message and updates its information, including the state. The processor then looks for the mission the spacecraft is dedicated for, and updates the status of the mission. Both management services writes their updates to the database after updating information, and notifies the user interface to update corresponding display.

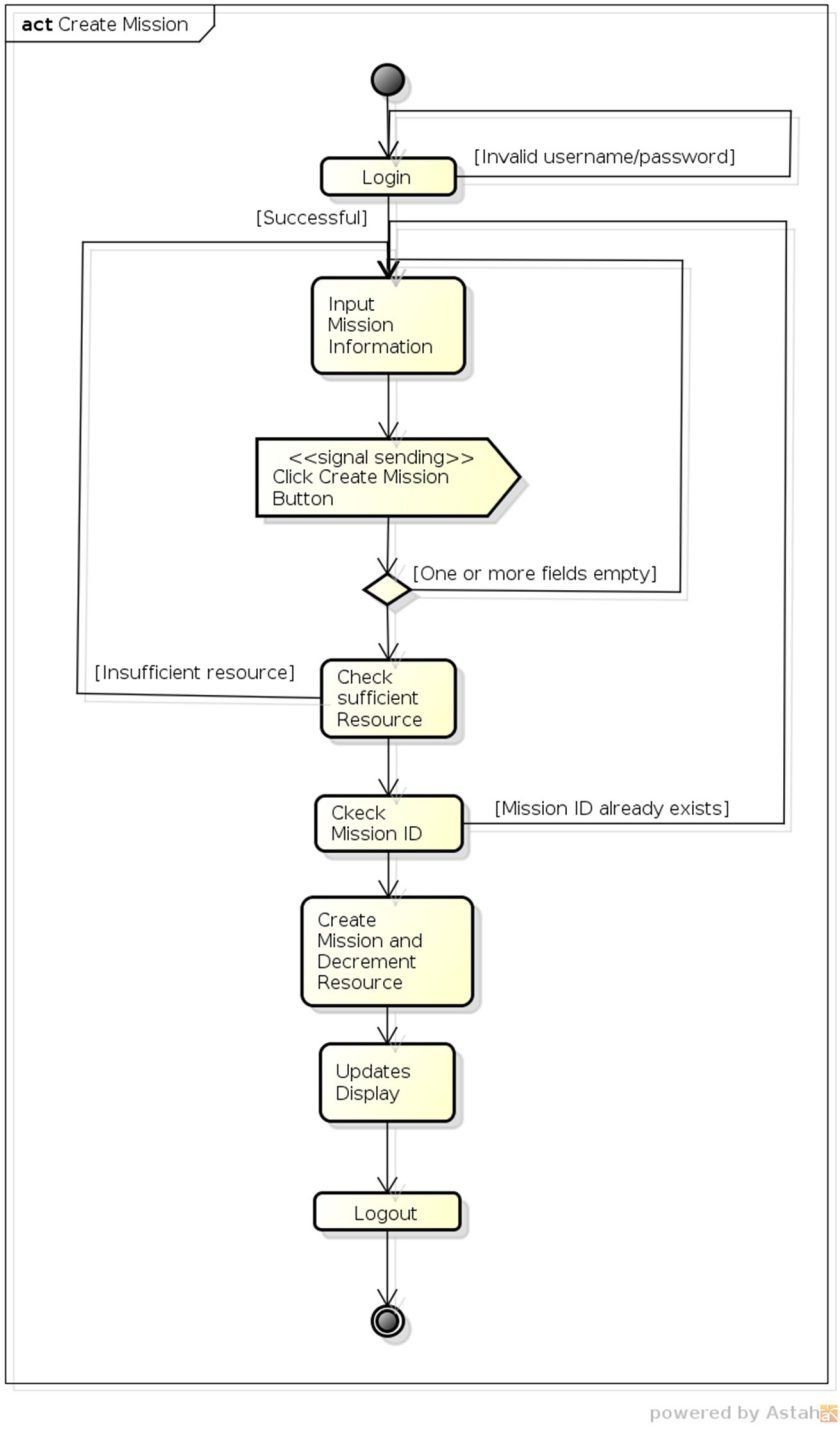


3. The message flow for provisioning a new mission. The administrator will user the user interface to enter necessary information for creating a mission, and click on a button to create it. The command and control center detects the click and get the input text in text fields. It invokes the mission management service and check for sufficient resources. If there are, the service creates a mission using the data the user provided, and decrements mission resources as specified by the mission. The subsystem then write all changes to missions and resources to the database and notifies the user interface to update its display, including adding the new mission to the list of existing missions, clears text fields and renew resource values.



Activity Diagram

The following diagram documents the provisioning of a new mission. An administrator first logs into the system where he can input mission data and click a button to submit. The interface checks for non-empty data field first, and let the mission management system check for sufficient resource and whether the id is existing. If all tests pass, the systems creates the mission and deducts resources. Finally the display will refresh to show the updated data.



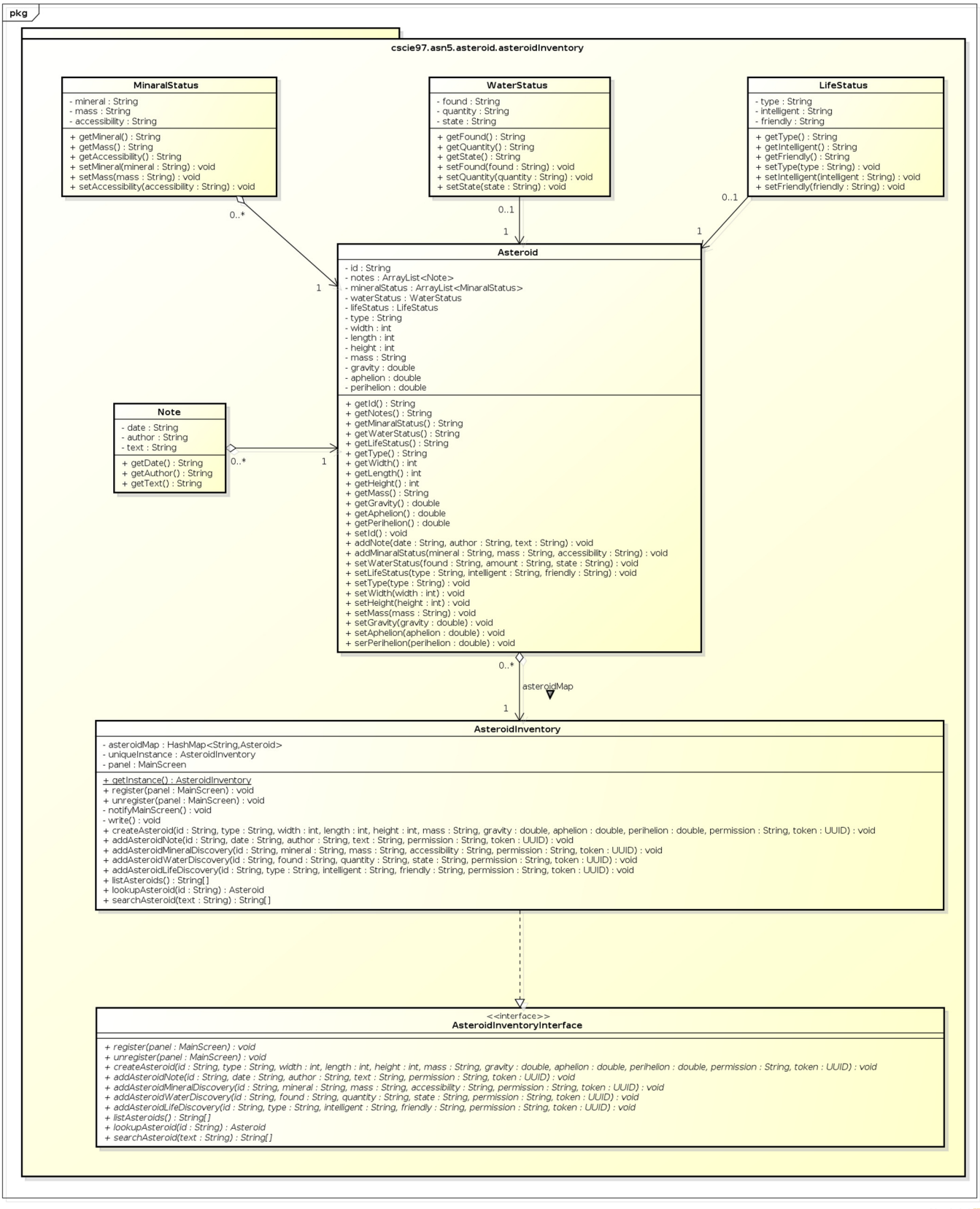
Implementation

This section of the document will describe the implementation details for all three subsystems and the user interface classes.

Class Diagram

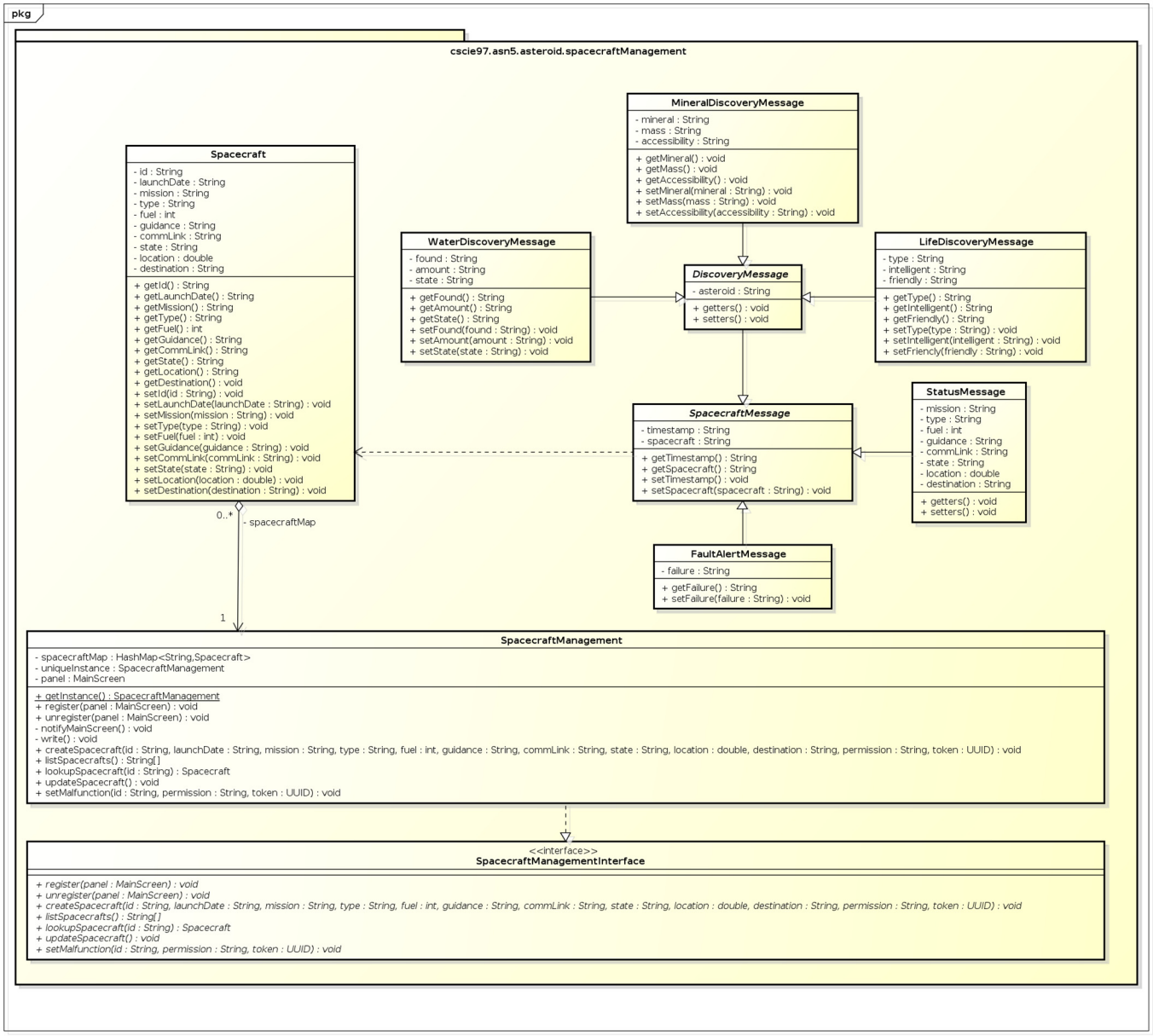
Asteroid Inventory System

The following class diagram defines the classes defined in the asteroid inventory system contained within package cscie97.asn5.asteroid.asteroidInventory.



Robotic Spacecraft Management System

The following class diagram defines the classes defined in the spacecraft management system contained within package cscie97.asn5.asteroid.spacecraftManagement.

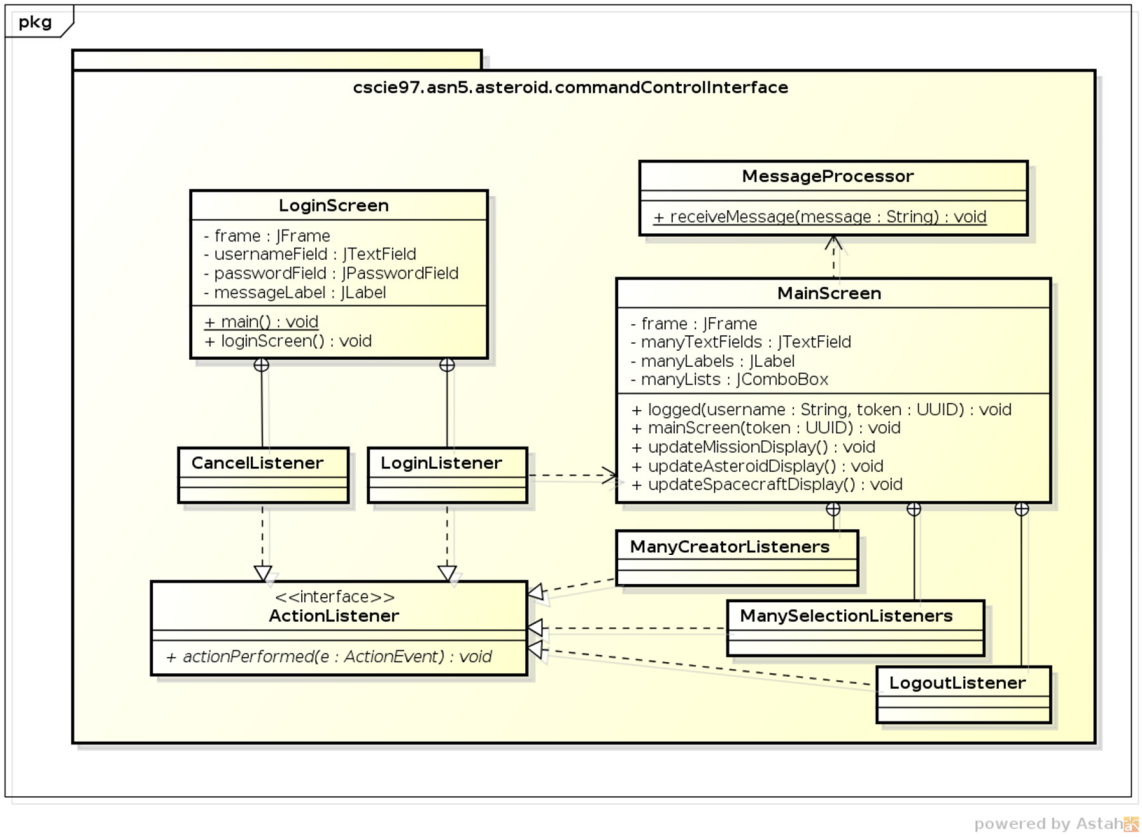


Mission Management System

The following class diagram defines the classes defined in the mission management system contained within package cscie97.asn5.asteroid.missionManagement.

Command and Control User Interface

The following class diagram defines the classes defined in the command and control user interface contained within package cscie97.asn5.asteroid.commandControlInterface.



Class Dictionary

This section specifies the class dictionaries for the class in all subsystems.

Asteroid Inventory System

This section specifies the class dictionary for the asteroid inventory system. The classes should be defined within the package cscie97.asn5.asteroid.asteroidInventory.

Asteroid

This class maintains properties of an asteroid, with getters and setters retrieving and updating an asteroid's information.

Methods

Method Name	Signature	Description
getId	() : String	Public method for getting the id
getNotes	() : String	Public method for getting all the notes in string format

getMinaralStatus	() : String	Public method for getting all the mineral status in string format
getWaterStatus	() : String	Public method for getting the water status in string format
getLifeStatus	() : String	Public method for getting the life status in string format
getType	() : String	Public method for getting the type of asteroid
getWidth	() : int	Public method for getting the width
getLength	() : int	Public method for getting the length
getHeight	() : int	Public method for getting the height
getMass	() : String	Public method for getting the mass
getGravity	() : double	Public method for getting the gravity
getAphelion	() : double	Public method for getting the aphelion
getPerihelion	() : double	Public method for getting the perihelion
setId	(id : String) : void	Public method for setting the id
addNote	(date : String, author : String, text : String) : void	Public method for adding a note
addMinaralStatus	(mineral : String, mass : String, accessibility : String) : void	Public method for adding a mineral status
setWaterStatus	(found : String, amount : String, state : String) : void	Public method for setting the water status
setLifeStatus	(type : String, intelligent : String, friendly : String) : void	Public method for setting the life status
setType	(type : String) : void	Public method for setting the type
setWidth	(width : int) : void	Public method for setting the width
setLength	(length : int) : void	Public method for setting the length

setHeight	(height : int) : void	Public method for setting the height
setMass	(mass : String) : void	Public method for setting the mass
setGravity	(gravity : double) : void	Public method for setting the gravity
setAphelion	(aphelion : double) : void	Public method for setting the aphelion
setPerihelion	(perihelion : double) : void	Public method for setting the perihelion

Properties

Property Name	Type	Description
id	String	id of the asteroid
mineralStatus	MineralStatus	mineral status of the asteroid
waterStatus	WaterStatus	water status of the asteroid
lifeStatus	LifeStatus	life status of the asteroid
type	String	type of the asteroid
width	int	width of the asteroid
length	int	length of the asteroid
height	int	height of the asteroid
mass	String	mass of the asteroid
gravity	double	gravity of the asteroid
aphelion	double	aphelion of the asteroid
perihelion	double	perihelion of the asteroid

Associations

Association Name	Type	Description
notes	ArrayList<Note>	notes of the asteroid

Note

This class provides encapsulation to note objects belonging to asteroids.

Methods

Method Name	Signature	Description

getDate	() : String	Public method for getting the date of note
getAuthor	() : String	Public method for getting the author of note
geText	() : String	Public method for getting the text of note

Properties

Property Name	Type	Description
date	String	the date of note
author	String	the author of note
text	String	text of note

MineralStatus

This class provides encapsulation to mineral status of asteroids.

Methods

Method Name	Signature	Description
getMineral	() : String	Public method for getting the mineral
getMass	() : String	Public method for getting the mass of mineral
getAccessibility	() : String	Public method for getting the accessibility of mineral
setMineral	(mineral : String) : void	Public method for setting the mineral
setMass	(mass : String) : void	Public method for setting the mass of mineral
setAccessibility	(accessibility : String) : void	Public method for setting the accessibility of mineral

Properties

Property Name	Type	Description
mineral	String	the mineral
mass	String	the mass of mineral
accessibility	String	the accessibility of mineral

WaterStatus

This class provides encapsulation to water status of asteroids.

Methods

Method Name	Signature	Description
getFound	() : String	Public method for getting found status of water
getQuantity	() : String	Public method for getting quantity of water
getState	() : String	Public method for getting state of water
setFound	(found : String) : void	Public method for setting found status of water
setQuantity	(quantity : String) : void	Public method for setting quantity of water
setState	(state : String) : void	Public method for setting state of water

Properties

Property Name	Type	Description
found	String	found status of water
quantity	String	quantity of water
state	String	state of water

LifeStatus

This class provides encapsulation to life status of asteroids.

Methods

Method Name	Signature	Description
getType	() : String	Public method for getting type of life
getIntelligent	() : String	Public method for getting whether life is intelligent
getFriendly	() : String	Public method for getting whether life is friendly
setType	(type : String) : void	Public method for setting type of life
setIntelligent	(intelligent : String) : void	Public method for setting whether life is intelligent

setFriendly	(friendly : String) : void	Public method for setting whether life is friendly

Properties

Property Name	Type	Description
type	String	type of life
intelligent	String	whether life is intelligent
friendly	String	whether life is friendly

AsteroidInventory

This singleton class manages all known asteroids using a map. It supports creating, listing, searching, retrieving, and updating asteroid information. When there is an update, it notifies the main screen observer to update its display.

Methods

Method Name	Signature	Description
getInstance	() : AsteroidInventory	Public method for getting the instance of the class
register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
notifyMainScreen	() : void	Private method for notifying the main screen for updates
write	() : void	Private method for writing the updated map of asteroids to database
createAsteroid	(id : String, type : String, width : int, length : int, height : int, mass : String, gravity : double, aphelion : double, perihelion : double, permission : String, token : UUID) : void	Public method for creating an asteroid
addAsteroidNote	(id : String, date : String, author : String, text : String, permission : String, token : UUID) : void	Public method for adding note to an asteroid

addAsteroidMineralDiscovery	(id : String, mineral : String, mass : String, accessibility : String, permission : String, token : UUID) : void	Public method for adding mineral discovery to an asteroid
addAsteroidWaterDiscovery	(id : String, found : String, quantity : String, state : String, permission : String, token : UUID) : void	Public method for setting water discovery to an asteroid
addAsteroidLifeDiscovery	(id : String, type : String, intelligent : String, friendly : String, permission : String, token : UUID) : void	Public method for setting life discovery to an asteroid
listAsteroids	() : String[]	Public method for getting all asteroids listed in String[] form for drop-down list display
lookupAsteroid	(id : String) : Asteroid	Public method for getting an asteroid object given its id
searchAsteroid	(text : String) : String[]	Public method for getting a list of asteroids with note matching a text query

Properties

Property Name	Type	Description
uniqueInstance	AsteroidInventory	The unique instance of the class
panel	MainScreen	The registered main screen

Associations

Association Name	Type	Description
asteroidMap	HashMap<String,Asteroid>	A map of the asteroids in database

AsteroidInventoryInterface

This interface provides function declarations to the command and control center for calling asteroid inventory methods.

Methods

Method Name	Signature	Description

register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
createAsteroid	(id : String, type : String, width : int, length : int, height : int, mass : String, gravity : double, aphelion : double, perihelion : double, permission : String, token : UUID) : void	Public method for creating an asteroid
addAsteroidNote	(id : String, date : String, author : String, text : String, permission : String, token : UUID) : void	Public method for adding note to an asteroid
addAsteroidMineralDiscovery	(id : String, mineral : String, mass : String, accessibility : String, permission : String, token : UUID) : void	Public method for adding mineral discovery to an asteroid
addAsteroidWaterDiscovery	(id : String, found : String, quantity : String, state : String, permission : String, token : UUID) : void	Public method for setting water discovery to an asteroid
addAsteroidLifeDiscovery	(id : String, type : String, intelligent : String, friendly : String, permission : String, token : UUID) : void	Public method for setting life discovery to an asteroid
listAsteroids	() : String[]	Public method for getting all asteroids listed in String[] form for drop-down list display
lookupAsteroid	(id : String) : Asteroid	Public method for getting an asteroid object given its id
searchAsteroid	(text : String) : String[]	Public method for getting a list of asteroids with note matching a text query

Robotic Spacecraft Management System

This section specifies the class dictionary for the spacecraft management system. The classes should be defined within the package cscie97.asn5.asteroid.spacecraftManagement.

Spacecraft

This class maintains properties of a spacecraft, with getters and setters retrieving and updating a spacecraft's information.

Methods

Method Name	Signature	Description
getId	() : String	Public method for getting the id
getLaunchDate	() : String	Public method for getting the launch date
getMission	() : String	Public method for getting the mission id
getType	() : String	Public method for getting the type
getFuel	() : int	Public method for getting the fuel remaining
getGuidance	() : String	Public method for getting the guidance status
getCommLink	() : String	Public method for getting the communication link status
getState	() : String	Public method for getting the state
getLocation	() : String	Public method for getting the location
getDestination	() : void	Public method for getting the destination
setId	(id : String) : void	Public method for setting the id
setLaunchDate	(launchDate : String) : void	Public method for setting the launch date
setMission	(mission : String) : void	Public method for setting the mission id
setType	(type : String) : void	Public method for setting the type
setFuel	(fuel : int) : void	Public method for setting the fuel remaining
setGuidance	(guidance : String) : void	Public method for setting the guidance status
setCommLink	(commLink : String) : void	Public method for setting the communication link status

setState	(state : String) : void	Public method for setting the state
setLocation	(location : double) : void	Public method for setting the location
setDestination	(destination : String) : void	Public method for setting the destination

Properties

Property Name	Type	Description
id	String	the id
launchDate	String	the launch date
mission	String	the mission id
type	String	the type
fuel	int	the fuel remaining
guidance	String	the guidance status
commLink	String	the communication link status
status	String	the status
location	double	the location
destination	String	the destination

SpacecraftManagement

This singleton class manages all known spacecraft using a map. It supports creating, listing, retrieving, and updating spacecraft information. When there is an update, it notifies the main screen observer to update its display.

Methods

Method Name	Signature	Description
getInstance	() : SpacecraftManagement	Public method for getting the instance of the class
register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
notifyMainScreen	() : void	Private method for notifying the main screen for updates

write	() : void	Private method for writing the updated map of asteroids to database
createSpacecraft	(id : String, launchDate : String, mission : String, type : String, fuel : int, guidance : String, commLink : String, state : String, location : double, destination : String, permission : String, token : UUID) : void	Public method for creating a spacecraft
listSpacecrafts	() : String[]	Public method for getting all spacecraft listed in String[] form for drop-down list display
lookupSpacecraft	(id : String) : Spacecraft	Public method for getting a spacecraft object given its id
updateSpacecraft	(id : String, launchDate : String, mission : String, type : String, fuel : int, guidance : String, commLink : String, state : String, location : double, destination : String, permission : String, token : UUID) : void	Public method for updating a spacecraft
setMalfunction	(id : String, permission : String, token : UUID) : void	Public method for setting a spacecraft as mulfunctioning

Properties

Property Name	Type	Description
uniqueInstance	SpacecraftManagement	The unique instance of the class
panel	MainScreen	The registered main screen

Associations

Association Name	Type	Description
spacecraftMap	HashMap<String,Spacecraft>	The map of all spacecraft in database

SpacecraftManagementInterface

This interface provides function declarations to the command and control center for calling spacecraft management methods.

Methods

Method Name	Signature	Description
register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
createSpacecraft	(id : String, launchDate : String, mission : String, type : String, fuel : int, guidance : String, commLink : String, state : String, location : double, destination : String, permission : String, token : UUID) : void	Public method for creating a spacecraft
listSpacecrafts	() : String[]	Public method for getting all spacecraft listed in String[] form for drop-down list display
lookupSpacecraft	(id : String) : Spacecraft	Public method for getting a spacecraft object given its id
updateSpacecraft	(id : String, launchDate : String, mission : String, type : String, fuel : int, guidance : String, commLink : String, state : String, location : double, destination : String, permission : String, token : UUID) : void	Public method for updating a spacecraft
setMalfunction	(id : String, permission : String, token : UUID) : void	Public method for setting a spacecraft as mulfunctioning

SpacecraftMessage

This abstract class maintains properties of a most general spacecraft message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description

getTimestamp	() : String	Public method for getting the timestamp
getS	() : String	Public method for getting the spacecraft that sent the message
setTimestamp	() : void	Public method for setting the timestamp
setSpacecraft	(spacecraft : String) : void	Public method for setting the spacecraft that sent the message

Properties

Property Name	Type	Description
timestamp	String	the timestamp
spacecraft	String	the spacecraft that sent the message

FaultAlertMessage

This class maintains properties of a spacecraft fault alert message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description
getFailure	() : String	Public method for getting the failure information
setFailure	(failure : String) : void	Public method for setting the failure information

Properties

Property Name	Type	Description
failure	String	the failure information

StatusMessage

This class maintains properties of a spacecraft's status message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description

getId	() : String	Public method for getting the id
getLaunchDate	() : String	Public method for getting the launch date
getMission	() : String	Public method for getting the mission id
getType	() : String	Public method for getting the type
getFuel	() : int	Public method for getting the fuel remaining
getGuidance	() : String	Public method for getting the guidance status
getCommLink	() : String	Public method for getting the communication link status
getState	() : String	Public method for getting the state
getLocation	() : String	Public method for getting the location
getDestination	() : void	Public method for getting the destination
setId	(id : String) : void	Public method for setting the id
setLaunchDate	(launchDate : String) : void	Public method for setting the launch date
setMission	(mission : String) : void	Public method for setting the mission id
setType	(type : String) : void	Public method for setting the type
setFuel	(fuel : int) : void	Public method for setting the fuel remaining
setGuidance	(guidance : String) : void	Public method for setting the guidance status
setCommLink	(commLink : String) : void	Public method for setting the communication link status
setState	(state : String) : void	Public method for setting the state
setLocation	(location : double) : void	Public method for setting the location
setDestination	(destination : String) : void	Public method for setting the destination

Properties

Property Name	Type	Description
mission	String	the mission id
type	String	the type
fuel	int	the fuel remaining
guidance	String	the guidance status
commLink	String	the communication link status
status	String	the status
location	double	the location
destination	String	the destination

DiscoveryMessage

This abstract class maintains properties of a general discovery message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description
getAsteroid	() : String	Public method for getting the asteroid being discovered
setAsteroid	(asteroid : String) : void	Public method for setting the asteroid being discovered

Properties

Property Name	Type	Description
asteroid	String	the asteroid being discovered

MineralDiscoveryMessage

This class maintains properties of a mineral discovery message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description
getMineral	() : String	Public method for getting the mineral
getMass	() : String	Public method for getting the mass of mineral

getAccessibility	() : String	Public method for getting the accessibility of mineral
setMineral	(mineral : String) : void	Public method for setting the mineral
setMass	(mass : String) : void	Public method for setting the mass of mineral
setAccessibility	(accessibility : String) : void	Public method for setting the accessibility of mineral

Properties

Property Name	Type	Description
mineral	String	the mineral
mass	String	the mass of mineral
accessibility	String	the accessibility of mineral

WaterDiscoveryMessage

This class maintains properties of a water discovery message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description
getFound	() : String	Public method for getting found status of water
getQuantity	() : String	Public method for getting quantity of water
getState	() : String	Public method for getting state of water
setFound	(found : String) : void	Public method for setting found status of water
setQuantity	(quantity : String) : void	Public method for setting quantity of water
setState	(state : String) : void	Public method for setting state of water

Properties

Property Name	Type	Description
found	String	found status of water
quantity	String	quantity of water

state	String	state of water

LifeDiscoveryMessage

This class maintains properties of a life discovery message, with getters and setters retrieving and updating the message.

Methods

Method Name	Signature	Description
getType	() : String	Public method for getting type of life
getIntelligent	() : String	Public method for getting whether life is intelligent
getFriendly	() : String	Public method for getting whether life is friendly
setType	(type : String) : void	Public method for setting type of life
setIntelligent	(intelligent : String) : void	Public method for setting whether life is intelligent
setFriendly	(friendly : String) : void	Public method for setting whether life is friendly

Properties

Property Name	Type	Description
type	String	type of life
intelligent	String	whether life is intelligent
friendly	String	whether life is friendly

Mission Management System

This section specifies the class dictionary for the mission management system. The classes should be defined within the package cscie97.asn5.asteroid.missionManagement.

Mission

This class maintains properties of a mission, with getters and setters retrieving and updating a mission information.

Methods

Method Name	Signature	Description
getId	() : String	Public method for getting the id

getName	() : String	Public method for getting the name
getPurpose	() : String	Public method for getting the purpose
getSpacecraft	() : String	Public method for getting the spacecraft used
getLaunchDate	() : String	Public method for getting the launch date
getETA	() : String	Public method for getting the estimated time of arrival
getDestination	() : String	Public method for getting the destination asteroid
getStatus	() : String	Public method for getting the mission status
setId	(id : String) : void	Public method for setting the id
setName	(name : String) : void	Public method for setting the name
setPurpose	(purpose: String) : void	Public method for setting the purpose
setSpacecraft	(spacecraft: String) : void	Public method for setting the spacecraft used
setLaunchDate	(launchDate: String) : void	Public method for setting the launch date
setETA	(eta: String) : void	Public method for setting the estimated time of arrival
setDestination	(destination: String) : void	Public method for setting the destination asteroid
setStatus	(status: String) : void	Public method for setting the mission

Properties

Property Name	Type	Description
id	String	the id
name	String	the name
purpose	String	the purpose
spacecraft	String	the spacecraft used
launchDate	String	the launch date
eta	String	the estimated time of arrival

destination	String	the destination asteroid
status	String	the mission status

MissionManagement

This singleton class manages all known missions using a map. It supports creating, listing, retrieving, and updating mission information. When there is an update, it notifies the main screen observer to update its display.

Methods

Method Name	Signature	Description
getInstance	() : MissionManagement	Public method for getting the instance of the class
register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
notifyMainScreen	() : void	Private method for notifying the main screen for updates
write	() : void	Private method for writing the updated map of asteroids to database
createMission	(id : String, name : String, purpose : String, spacecraft : String, launchDate : String, eta : String, destination : String, status : String, permission : String, token : UUID) : void	Public method for creating a mission
listMissions	() : String[]	Public method for getting all missions listed in String[] form for drop-down list display

lookupMission	(id : String) : Mission	Public method for getting a missions object given its id
getFuel	() : int	Public method for getting the remaining fuel
getSpacecraft	() : int	Public method for getting the remaining spacecraft
getBudget	() : int	Public method for getting the remaining budget
getCommLink	() : String	Public method for getting the communication link status
getAutoControlSystem	() : String	Public method for getting the auto control system status
setFuel	() : void	Public method for setting the remaining fuel
setSpacecraft	(amount : int) : void	Public method for setting the remaining spacecraft
setBudget	(amount : int) : void	Public method for setting the remaining budget
setCommLink	(commLink : String) : void	Public method for setting the communication link status
setAutoControlSystem	(autoControlSystem : String) : void	Public method for setting the auto control system status

Properties

Property Name	Type	Description

uniqueInstance	MissionManagement	The unique instance of the class
panel	MainScreen	The registered main screen

Associations

Association Name	Type	Description
missionMap	HashMap<String,Mission>	The map of all missions in database

MissionManagementInterface

This interface provides function declarations to the command and control center for calling mission management methods.

Methods

Method Name	Signature	Description
register	(panel : MainScreen) : void	Public method for registering the main screen for notification
unregister	(panel : MainScreen) : void	Public method for unregistering the main screen for notification
createMission	(id : String, name : String, purpose : String, spacecraft : String, launchDate : String, eta : String, destination : String, status : String, permission : String, token : UUID) : void	Public method for creating a mission
listMissions	() : String[]	Public method for getting all missions listed in String[] form for drop-down list display
lookupMission	(id : String) : Mission	Public method for getting a missions object given its id

getFuel	() : int	Public method for getting the remaining fuel
getSpacecraft	() : int	Public method for getting the remaining spacecraft
getBudget	() : int	Public method for getting the remaining budget
getCommLink	() : String	Public method for getting the communication link status
getAutoControlSystem	() : String	Public method for getting the auto control system status
setFuel	() : void	Public method for setting the remaining fuel
setSpacecraft	(amount : int) : void	Public method for setting the remaining spacecraft
setBudget	(amount : int) : void	Public method for setting the remaining budget
setCommLink	(commLink : String) : void	Public method for setting the communication link status
setAutoControlSystem	(autoControlSystem : String) : void	Public method for setting the auto control system status

Resources

This singleton class manages current mission resources. It supports getting and setting any of the resource entries.

Methods

Method Name	Signature	Description

getFuel	() : int	Public method for getting the remaining fuel
getSpacecraft	() : int	Public method for getting the remaining spacecraft
getBudget	() : int	Public method for getting the remaining budget
getCommLink	() : String	Public method for getting the communication link status
getAutoControlSystem	() : String	Public method for getting the auto control system status
setFuel	() : void	Public method for setting the remaining fuel
setSpacecraft	(amount : int) : void	Public method for setting the remaining spacecraft
setBudget	(amount : int) : void	Public method for setting the remaining budget
setCommLink	(commLink : String) : void	Public method for setting the communication link status
setAutoControlSystem	(autoControlSystem : String) : void	Public method for setting the auto control system status

Properties

Property Name	Type	Description
uniqueInstance	Resources	The unique instance of the class
fuel	int	the remaining fuel

spacecraft	int	the remaining spacecraft
budget	int	the remaining budget
commLink	String	the communication link status
autoControlSystem	String	the auto control system status

Command and Control User Interface

This section specifies the class dictionary for the command and control center. The classes should be defined within the package cscie97.asn5.asteroid.commandControlInterface.

LoginScreen

This class supports a window frame for user to log in to the system.

Methods

Method Name	Signature	Description
main	() : void	Public static method that initiates the GUI
loginScreen	() : void	Public method for displaying the login screen

Properties

Property Name	Type	Description
frame	JFrame	the main frame
usernameField	JTextField	the username text field
passwordField	JPasswordField	the password field
messageLabel	JLabel	the message label notifying incorrect username/password

MainScreen

This class supports a window frame for user to conduct list, search, create, and manage tasks via the graphical interface provided.

Methods

Method Name	Signature	Description
logged	(username : String, token : UUID) : void	Public method invoked after successful login, it then calls

		mainScreen and pass in the token it received
mainScreen	(token : UUID) : void	Public method for displaying the main screen
updateMissionDisplay	() : void	Public method for updating mission display
updateAsteroidDisplay	() : void	Public method for updating astroid display
updateSpacecraftDisplay	() : void	Public method for updating spacecraft display

Properties

Property Name	Type	Description
frame	JFrame	the main frame
manyTextFields	JTextField	text fields for data entering
manyLabels	JLabel	labels for displaying data
manyLists	JComboBox	drop-down lists for selection

MessageProcessor

This class receives messages from spacecraft, processes them, and calls update methods in other subsystems in accordance with the message received.

Methods

Method Name	Signature	Description
receiveMessage	(message : String) : void	Public method for receiving and processing the message from spacecraft. It calls other services to update in accordance with the message.

Implementation Details

This system used four design patterns: observer, mediator, façade, and singleton.

The observer pattern describes how updates of subsystems notify the main screen to invoke its update display methods when new incidents happen.

The mediator pattern describes the usage of the command and control center and its dependency with the other three subsystems. Having the main screen and the message processor all in this package eliminates the need for inter-subsystem communication. The main

screen acts as an interface between the human user and the subsystems, while the message processor takes action to asteroid status and mission status when spacecraft wants to update their status.

Façade pattern describes the usage of the main screen. Although each of the subsystems provide many functions in their interface, only the creating and searching functions are directly exposed to the user in forms of text fields and buttons (except updating mission resources and adding notes to asteroids). All listings are hidden in the drop-down list, while updates are taken care of by the message processor alone.

Singleton pattern shows in the asteroid inventory, mission management, resources, and spacecraft management classes. They should only have a single instance to keep one accurate copy of asteroids, missions, resources, and spacecraft.

There are some other details worth mentioning:

All listing and searching functions in the subsystems have return type of String[] to avoid another step of conversion for the drop-down lists.

For the two graphical windows, many listeners need to be implemented for each button and drop-down list so that the window can respond immediately.

Resources class lies in the mission management package to eliminate inter-subsystem communication, as the resource values are checked and updated when creating missions. In this way, functions accessing resource data will appear in the mission management interface.

The last issue concerns with creating spacecraft message. In the current implementation, only human users have the ability to make up meaningful messages (either by entering the complete message or using a couple of drop-down lists to shrink the range of the message type and parameters to enter). There does not seem to be a way for the spacecraft objects to create valid message on their own, except one way: the system designer provides a whole range of possible mineral options, accessibility types, parts of the spacecraft that may go wrong… and let a spacecraft randomly choose from the pool and compose messages accordingly.

Testing and Risks

For testing the implemented design, I first tested the login screen's compatibility with the authentication service by providing both correct and incorrect login information.

To test the main screen's functionality, I created sample asteroids and found them immediately in the drop-down list of all asteroids; clicking it reveals the information that I just entered. After adding note to the asteroid, the text search functionality can also retrieve the asteroid matching the text.

For the message processor, I can create a set of asteroid, mission, and spacecraft. Then having the spacecraft selected, I can simulate it sending an update message or a discovery message and see the corresponding fields of the spacecraft and asteroid updated immediately.

To verify the mission resources part, I can simply create a mission and see the resources decremented by some value (set to a random number when creating a mission). The Apply Change button works if any text field in this zone is non-empty and contains valid value.

There is one place where risks may happen: as mission id, spacecraft id, and asteroid id are interleaved and the first two are dependent, input mistake on such identifier will lead to looking for invalid identifiers, which will return null. A proposed solution is to enforce a sequence in which asteroid, spacecraft, and mission is created. When an entity being created requires the value of an identifier of another class, we should use a drop-down list to select from the database of existing entities to avoid such mismatch problems.