# ASTEROID EXPLORATION SYSTEM

Design Review

12/20/2013

Fanxing Meng
REVIEWER: JAYAPRAKASH GANTA

# Table of Contents

## Comments

1. Note of asteroids can be abstracted to a class, since it is already a compound;

2. Weight, length, height and mass can be set to type double;

3. No check for sufficient resources upon creating a mission

4. When a mission is created, a spacecraft id needs to be provided; similarly when a spacecraft is created, a mission id also needs to be provided. What is the right sequence of creating these? Either way? How to prevent inconsistency?

5. When a spacecraft's status is updated, the mission should also update its state. How many corresponding spacecraft status and mission states are there? Who should specify it?

## Changes

1. Changed note, mineral status, water status, and life status all into corresponding classes.

2. Since there is ± sign in mass, it could not be represented by double. Weight, length, height does not seem to have decimals, so left as int.

3. Added check for resources when creating mission.

4. Proposed a version of activity sequence in which a mission is first created without having a spacecraft id but with an asteroid id selected from a drop-down list, then when creating a spacecraft, a mission id need to be selected using a drop-down list but the destination asteroid is taken directly from the mission's data.

5. Added the capability of changing the mission's spacecraft if the spacecraft's status is updated with a new mission.

## Design Patterns

As mentioned in 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.

## Modular Approach

The modular approach forced a simplified inter-connection schema for the system. Without a central mediator, all subsystems will need to know other subsystems' interfaces, thus harming flexibility in future modifications.

## Update to Modules

I had to update two parts: one is putting the message processer to the command and control center package to avoid dependencies between spacecraft management package with the other two. Another change is putting resources into the mission management package to get one less dependency among resources, mission management, and command and control center.

## Design Review

The design review helped me recognize a potentially serious inconsistency problem and led me to think of using enforced sequence and drop-down lists to solve it. The review also helped me think of managing the message creation.

## Authentication Service

The authentication interface was simple enough to implement, but managing services and roles may require another window.

## Implement

Implementation is about 2/3 complete.