Miguel Padilla & Diego Acosta

April 22, 2025

CS 3331-Advanced Object-Oriented Programming-Spring 2025

Dr. Gurijala

Project Part 1

This work was done individually/as a team and completely on our own. we did not share, reproduce, or alter any part of this assignment for any purpose. I/we did not share code, upload this assignment online in any form, or view/received/modified code written from anyone else. All deliverables were produced entirely on my/our own. This assignment is part of an academic course at The University of Texas at El Paso and a grade will be assigned for the work we produced.

Program Explanation

This project had us implement a debris analysis system that simulates the function of space agency control center. It reads a csv file that contains the debris and satellites, and allows different users such as Scientist, Space Agency Rep, Policy maker, and Administrator. This also allows the Scientist to perform the following tasks, perform space-tracking tasks such as filtering objects, assessing orbit status, calculating orbital risk, and generating debris density reports.

To perform this assignment, we first created a rough draft of the UML Diagram, and Level 2 UML use case diagram. For the UML Diagram we decided to use the following classes, Debris, Filereader, Logger, Payload, RocketBody, RunSimulation, Satellite, SpaceObject, UI (Mission Control), and Unkown. This allowed use to break down the problem to smaller more manageable tasks. With the Use case Diagram we created, it allowed us to see a high-level overview of how the program can function. We also used inheritance and encapsulation to ensure that we can properly strengthen our program to run with minimal errors, this allowed us to further breakdown the problem to be smaller and allowed us to check each class and method along the way of creating the project.

What did I learn?

What we have learned is how to practically use UML use cases, and class diagrams and have it translated into the function of the program, we also learned how UML diagrams have allowed us to break down a problem to be smaller and allow us to transform a high-level overview problem to a low level overview of the proble. Some classes could have been one space object and not creating a lot of different class files for them, such as Unknown, Satellite, Rocketbody, etc., these objects could have been set in SpaceObject. This one of the ways we could have improved the code and reduce the classes and points of failure. We assumed that the problem could have been a bit to daunting given the time frame, but as we slowly broke down the problem into smaller parts and implemented and tested the problem along the way, we found it to be doable.

Solution Design

I used ArrayList<Debris> to store and filter objects because the data set is moderately sized and sequential filtering was sufficient. Risk levels are calculated based on drift, the difference between longitude and average longitude. The system revolves around an abstract base class SpaceObject, which defines shared attributes and behaviors for all orbital entities. Subclasses like Debris, Satellite, Payload, RocketBody, and Unknown inherit from SpaceObject and implement the abstract method displayInfo() for role-specific outputs. User interaction is managed by the UI class, which dynamically adapts based on the logged-in role (Scientist, Space Agency Representative, Policymaker). It logs all interactions through the Logger class, which writes timestamped entries to a persistent log file. The RunSimulation class acts as the program's entry point, demonstrating how components interact.

**Testing**

During testing, the system consistently produced correct results and cleanly formatted outputs across multiple scenarios. The console output confirmed that all stages of the pipeline executed successfully, including CSV loading, debris analysis, and report generation. For instance, after initializing the system with RunSimulation, it reported the correct number of loaded debris objects, executed the full orbit assessment logic, and printed confirmation of saved output files. We did run into the program breaking, so we figured out the issue and saw how we could further improve the program, we saw that when we first had the implementation of the readfile, it would not find the file, so we improved the code by better implementing the parser and reading of the file. We believe we tested the solution enough given the time, but we could improve our testing methods, we can have better implementation of test cases or have a separate test file to run our test.

**Test Results**

**Code Review**

The code does what it is supposed to do, but we do think it can be simplified, as we stated earlier in the document by having less classes, and consolidating them in the SpaceObject file, and creating a user file and having the Scientist, Policymaker, and the other actors to improve flexibility of code. The code is a mix of dynamic and hardcoded, but mostly dynamic, the test cases are hardcoded. In conclusion, the code is Maintainable.

We think the logic of the code is sound. We have tested and corrected most expected and intended issues. There may be test cases that it may fail, but we either have not accounted for it or found it.

We also found that the code is readable especially with the java doc comments aiding in the explanation of the function of methods, class, and the attributes. We also think the naming conventions improve the readability of the code. We can probably modify or adjust the comments explanation or possibly improve naming conventions in certain class files. Overall, the coding structure is sound and readable, and the document follow all java naming conventions and the comments allow for good documentation. The code is in O(n) or in linear complexity, the code review has allowed us to follow along and improve the code as we went along and implemented methods.