Caitlin Gregory & Daniela Gutierrez

05/06/25

CS 3331 – Advanced Object-Oriented Programming – Spring 2025

Final Lab Report

1. Program Explanation

a) After completing Part 1, our main objective for Part 2 was to adapt the system to handle input files with non-standard column orders. To address this, we redesigned how the `FileLoader` class processes CSV files. Instead of relying on hardcoded column positions, we read the header row dynamically and stored the column names and their corresponding indexes in a `HashMap`. This mapping was then passed to the `DictionaryBuilder`, allowing us to access data fields by their header name rather than their fixed position. By doing this, we made the system more flexible and robust, especially when working with jumbled or reordered CSV files. We approached the task by breaking it down into smaller, manageable components—first extracting the headers, then mapping the indices, and finally modifying downstream classes to use this new dynamic structure.

b) Next, we focused on enhancing our system's modularity and maintainability by incorporating a design pattern and defining a reusable interface. We chose to implement the interface in the `DisplayMenu` class to formalize the behavior expected from different user role menus (e.g., Scientist, Space Agency Representative, Administrator). By doing so, we ensured that all user types adhered to a consistent structure when interacting with the menu system, making future expansions (like adding new roles) more seamless and scalable.

In addition, we applied the **Singleton design pattern** to the `Logger` class. This was essential because we needed a centralized and consistent logging mechanism across the entire system. By enforcing a single instance of the `Logger`, we avoided issues like duplicated logs or file access conflicts. The pattern allowed us to ensure that all parts of the system—regardless of user type—log their actions in a unified and timestamped manner, making debugging and auditing significantly easier.

Both of these implementations reflect core object-oriented principles: abstraction, encapsulation, and low coupling. They also helped us reduce redundancy, improve clarity, and prepare the system for future feature additions.

c) Next, we implemented functionality specific to the **Space Agency Representative** user role. This role is responsible for analyzing debris patterns and assessing long-term risks in LEO. To support this, we added two core features:
   1. **Analyze Long-Term Impact** – This feature filters and displays all space objects in Low Earth Orbit (LEO) that are more than 200 days old and have experienced at least one conjunction. These criteria help identify potentially hazardous objects that could interfere with current and future space missions.
   2. **Generate Density Reports** – This function allows the representative to enter a minimum and maximum longitude. The system then scans the dataset and prints all objects that fall within that range, including details like record ID, satellite name, country, orbit type, launch year, and object type.

   To build these features, we reused and extended the core logic within the `MissionControl` class. We ensured each function printed clear, organized output and included logging calls to maintain an audit trail. Exception handling was added to manage incorrect input gracefully (e.g., non-numeric longitude values), improving the system's overall user experience.

   These features were developed with reusability and scalability in mind, using helper methods and structured conditions to make future extensions—like exporting filtered data—easy to implement.

d) Finally, we added the functionality for the Administrator. The Administrator is responsible for the creation, managing, and deletion of all three types of users.

1. To create a new user, we first ask the user to enter a username and a password. Since the users are being stored in a dictionary, we can easily access them and see if a certain user is already being used. We are also using hashing and salting for the password, and we are also saving this information on the CSV file so the Administrator can log in the user back when they enter their password.
2. To manage a user, the menu first asks the Administrator for the user's username. Then, the user is prompted on whether they want to change the username, password, or both. Depending on the selection, the system then just asks the

Administrator to enter the new desired credentials. Finally, the corresponding information is updated properly both on the dictionary and the CSV file.

3. To delete a user, the menu asks the Administrator for the username of the user they want to delete. After this, the user is simply deleted from the CSV file and the dictionary.

The Administrator class is tightly related to the UserManager class. Administrator is basically calling the methods on userManager to do all the needed operations. We decided to add this other class specially because of the hashing/salting, we believe that integrating this class adds a layer of abstraction and makes the code more cohesive.

2. What did I learn?

As a result of this assignment, I learned how to implement interfaces and apply design patterns in a real-world, multi-class Java application. Specifically, I now understand the value of using an interface to enforce consistent behavior across different user roles and how the Singleton pattern can centralize functionality—in our case, logging user activity.

While our current solution is functional and robust, it could be improved by separating more responsibilities into smaller, dedicated classes. For example, the `DisplayMenu` class currently handles both logic and interaction—it could be refactored using the MVC (Model-View-Controller) pattern to cleanly separate user interface handling from business logic.

Another way to solve parts of this problem could be by using a GUI-based interface instead of a command-line one. This would improve usability and make the system more intuitive for non-technical users like policymakers. Additionally, integrating a small database instead of relying solely on CSV files could help improve performance and scalability as data grows.

Overall, this assignment took approximately the period that was allotted, including time spent debugging input parsing, designing the class structure, implementing user role logic, writing JUnit tests, and preparing the documentation and UML diagrams.

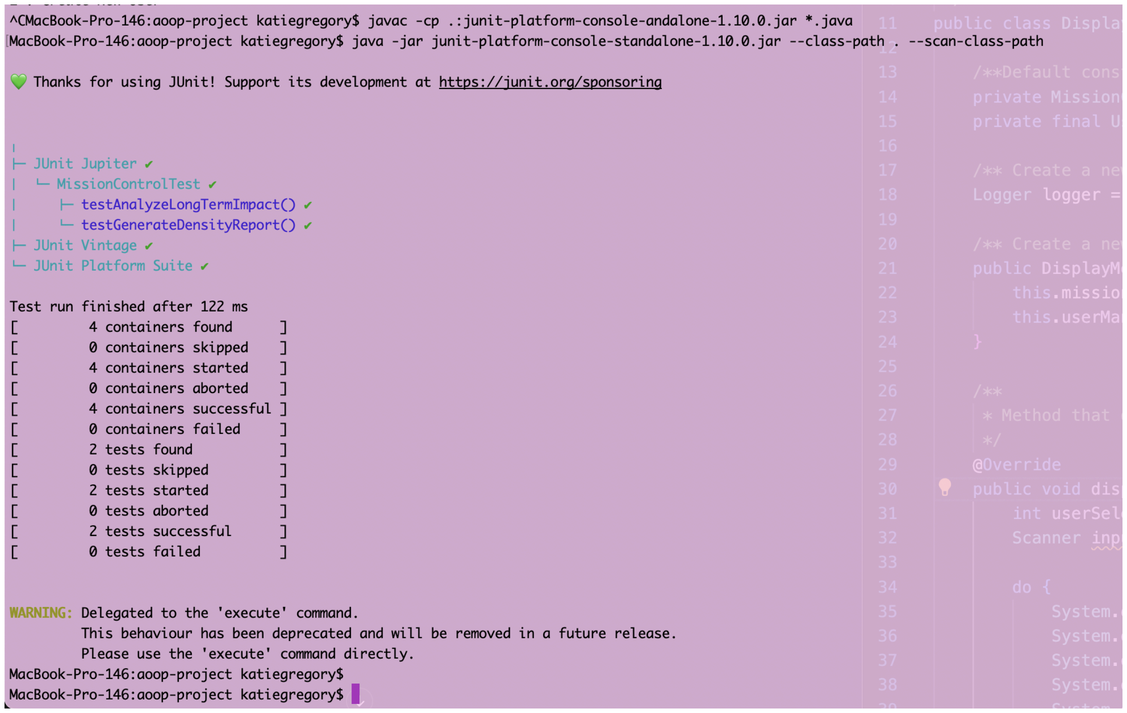
3. Solution Design

In this program, we developed an object-oriented system to track and analyze space debris in low Earth orbit (LEO), supporting multiple user roles including Scientists, Space Agency Representatives, and Administrators. The system allows users to track specific object types, assess whether debris is still in orbit, generate density reports based on longitude ranges, and analyze long-term orbital impact based on conjunction count and object age. To solve this, we followed a modular approach using inheritance, encapsulation, and design patterns like Singleton (for logging) and Factory (for creating specific space object types). We used HashMaps for efficient lookup of users and space objects by unique IDs, and ArrayLists to collect and export filtered data. Our assumptions included that CSV files would have all required headers, though not necessarily in a fixed order, and that string comparison would suffice for identifying orbit types like "LEO." The system also assumes that data is stored in memory during runtime, with updates written to CSV or TXT upon exiting. Overall, the solution is structured to be scalable, maintainable, and user-friendly through a menu-driven interface and layered design.

4. Testing

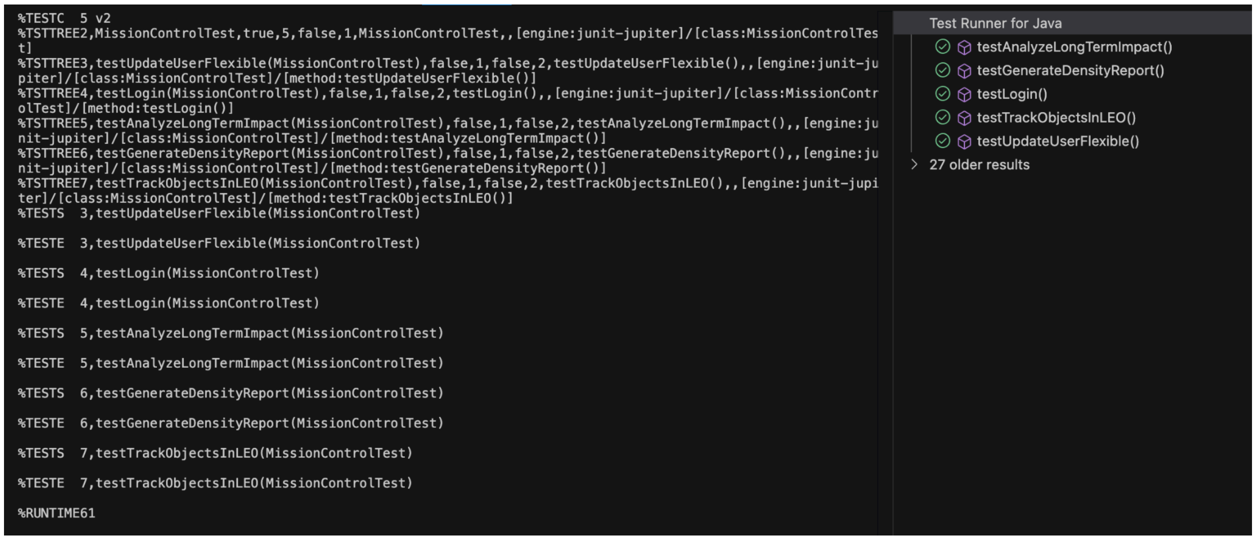
To test the program, we used both black-box and white-box testing approaches. Black-box testing was applied through the menu-driven interface to simulate real user interactions, ensuring that role-based actions like tracking debris, generating reports, and managing users functioned as expected without knowing the internal code. White-box testing was used through JUnit test cases in `MissionControlTest.java`, where we verified specific methods such as `analyzeLongTermImpact()` and `generateDensityReport()` by asserting the presence or absence of output based on known input data. These tests covered edge cases, such as objects with zero conjunctions, non-LEO orbits, and longitudes outside the specified range. While the current testing validates core functionalities, the coverage could be expanded to include exception handling, user authentication, and data parsing errors from malformed CSV files. We did break the program intentionally during development—for example, by feeding in incomplete data or incorrect inputs—to identify weaknesses in validation and improve robustness. Future improvements could include automating more UI-level tests, adding mock objects for file handling, and integrating test coverage tools to ensure all branches and conditions are verified.

5. Test results



```
^CMacBook-Pro-146:aoop-project katiegregory$ javac -cp .:junit-platform-console-andalone-1.10.0.jar *.java
MacBook-Pro-146:aoop-project katiegregory$ java -jar junit-platform-console-standalone-1.10.0.jar --class-path . --scan-class-path

💚 Thanks for using JUnit! Support its development at https://junit.org/sponsoring


╷
├─ JUnit Jupiter ✔
│  └─ MissionControlTest ✔
│     ├─ testAnalyzeLongTermImpact() ✔
│     └─ testGenerateDensityReport() ✔
├─ JUnit Vintage ✔
└─ JUnit Platform Suite ✔

Test run finished after 122 ms
[         4 containers found      ]
[         0 containers skipped    ]
[         4 containers started    ]
[         0 containers aborted    ]
[         4 containers successful ]
[         0 containers failed     ]
[         2 tests found           ]
[         0 tests skipped         ]
[         2 tests started         ]
[         0 tests aborted         ]
[         2 tests successful      ]
[         0 tests failed          ]


WARNING: Delegated to the 'execute' command.
         This behaviour has been deprecated and will be removed in a future release.
         Please use the 'execute' command directly.
MacBook-Pro-146:aoop-project katiegregory$
MacBook-Pro-146:aoop-project katiegregory$
```

```
public class Displa

    /**Default cons
    private Mission
    private final U

    /** Create a ne
    Logger logger =

    /** Create a ne
    public DisplayM
        this.missio
        this.userMa
    }

    /**
     * Method that
     */
    @Override
    public void dis
        int userSel
        Scanner inp

        do {
            System.
            System.
            System.
            System.
```



```
%TESTC  5 v2
%TSTTREE2,MissionControlTest,true,5,false,1,MissionControlTest,,[engine:junit-jupiter]/[class:MissionControlTes
t]
%TSTTREE3,testUpdateUserFlexible(MissionControlTest),false,1,false,2,testUpdateUserFlexible(),,[engine:junit-ju
piter]/[class:MissionControlTest]/[method:testUpdateUserFlexible()]
%TSTTREE4,testLogin(MissionControlTest),false,1,false,2,testLogin(),,[engine:junit-jupiter]/[class:MissionContr
olTest]/[method:testLogin()]
%TSTTREE5,testAnalyzeLongTermImpact(MissionControlTest),false,1,false,2,testAnalyzeLongTermImpact(),,[engine:ju
nit-jupiter]/[class:MissionControlTest]/[method:testAnalyzeLongTermImpact()]
%TSTTREE6,testGenerateDensityReport(MissionControlTest),false,1,false,2,testGenerateDensityReport(),,[engine:ju
nit-jupiter]/[class:MissionControlTest]/[method:testGenerateDensityReport()]
%TSTTREE7,testTrackObjectsInLEO(MissionControlTest),false,1,false,2,testTrackObjectsInLEO(),,[engine:junit-jupi
ter]/[class:MissionControlTest]/[method:testTrackObjectsInLEO()]
%TESTS  3,testUpdateUserFlexible(MissionControlTest)

%TESTE  3,testUpdateUserFlexible(MissionControlTest)

%TESTS  4,testLogin(MissionControlTest)

%TESTE  4,testLogin(MissionControlTest)

%TESTS  5,testAnalyzeLongTermImpact(MissionControlTest)

%TESTE  5,testAnalyzeLongTermImpact(MissionControlTest)

%TESTS  6,testGenerateDensityReport(MissionControlTest)

%TESTE  6,testGenerateDensityReport(MissionControlTest)

%TESTS  7,testTrackObjectsInLEO(MissionControlTest)

%TESTE  7,testTrackObjectsInLEO(MissionControlTest)

%RUNTIME61
```

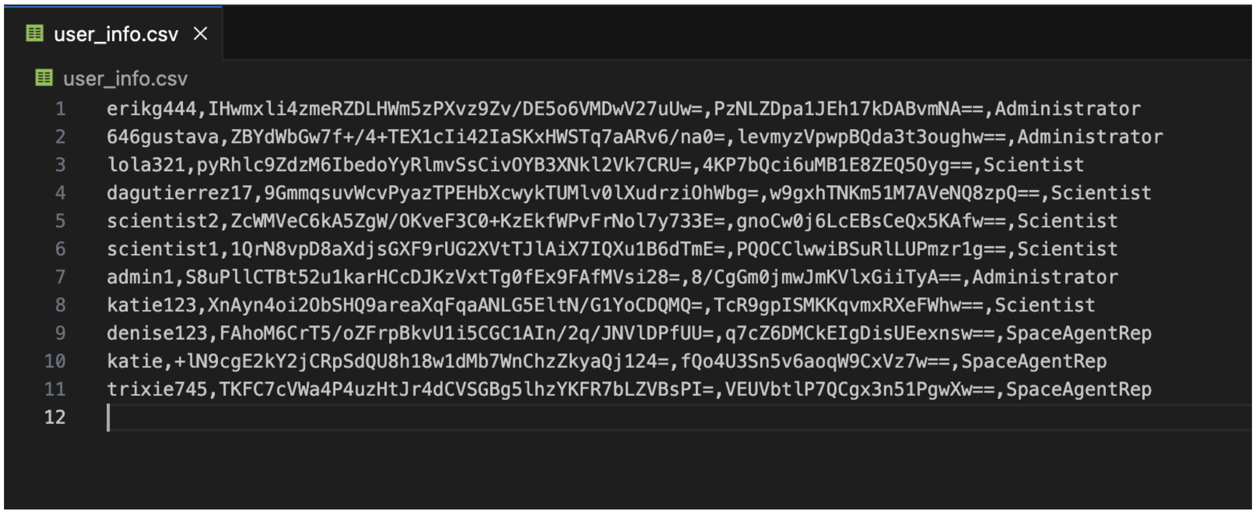
```
Test Runner for Java
  ✓ testAnalyzeLongTermImpact()
  ✓ testGenerateDensityReport()
  ✓ testLogin()
  ✓ testTrackObjectsInLEO()
  ✓ testUpdateUserFlexible()
> 27 older results
```

We created tests for 5 methods in our program:

1. Analyze long term impact
2. Generate density report
3. Track objects in LEO
4. Log In
5. Manage (update) users

All tests passed successfully, confirming that the key features behave as expected under both valid and invalid input scenarios.

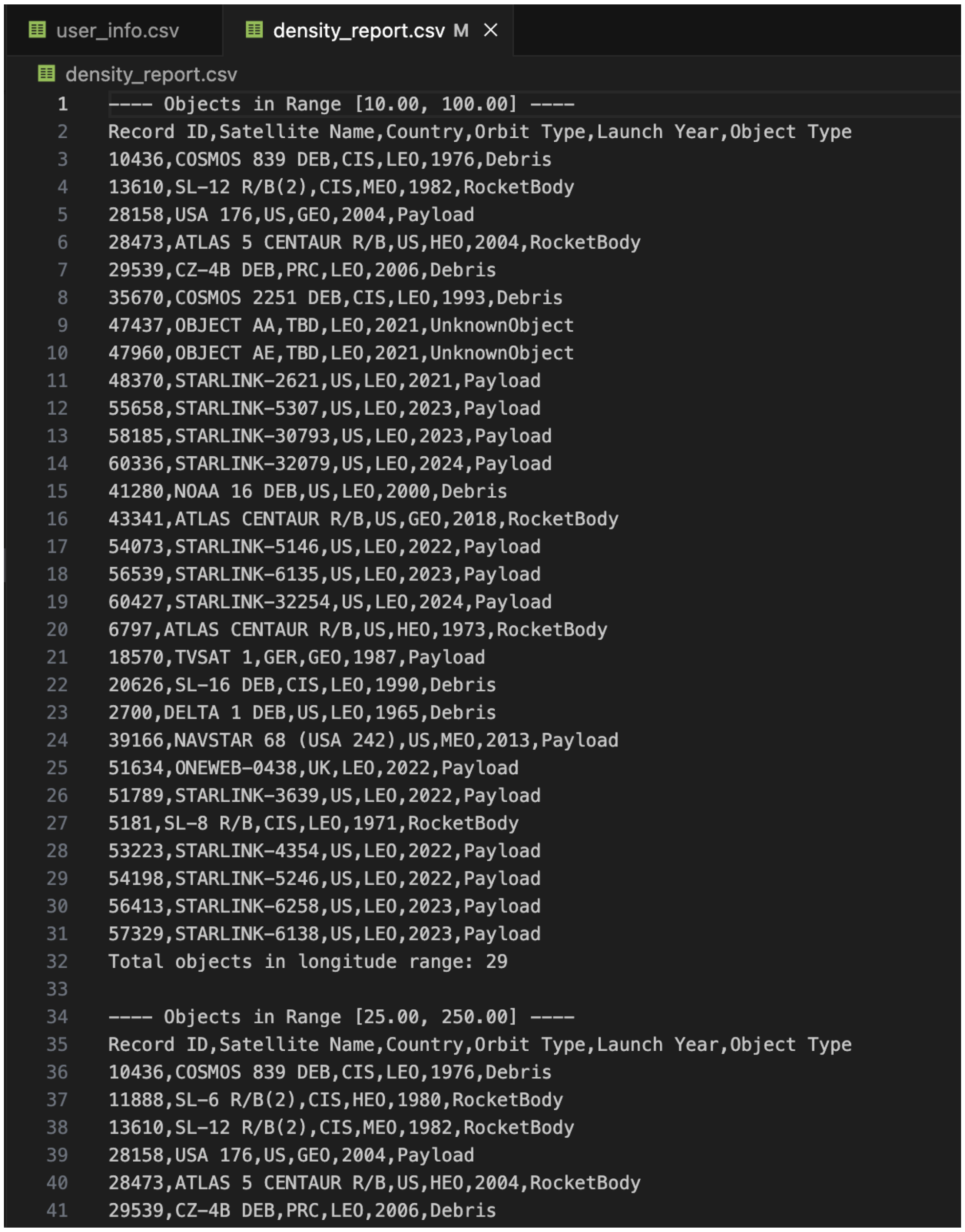
CSV File that stores the users that are created:



user_info.csv

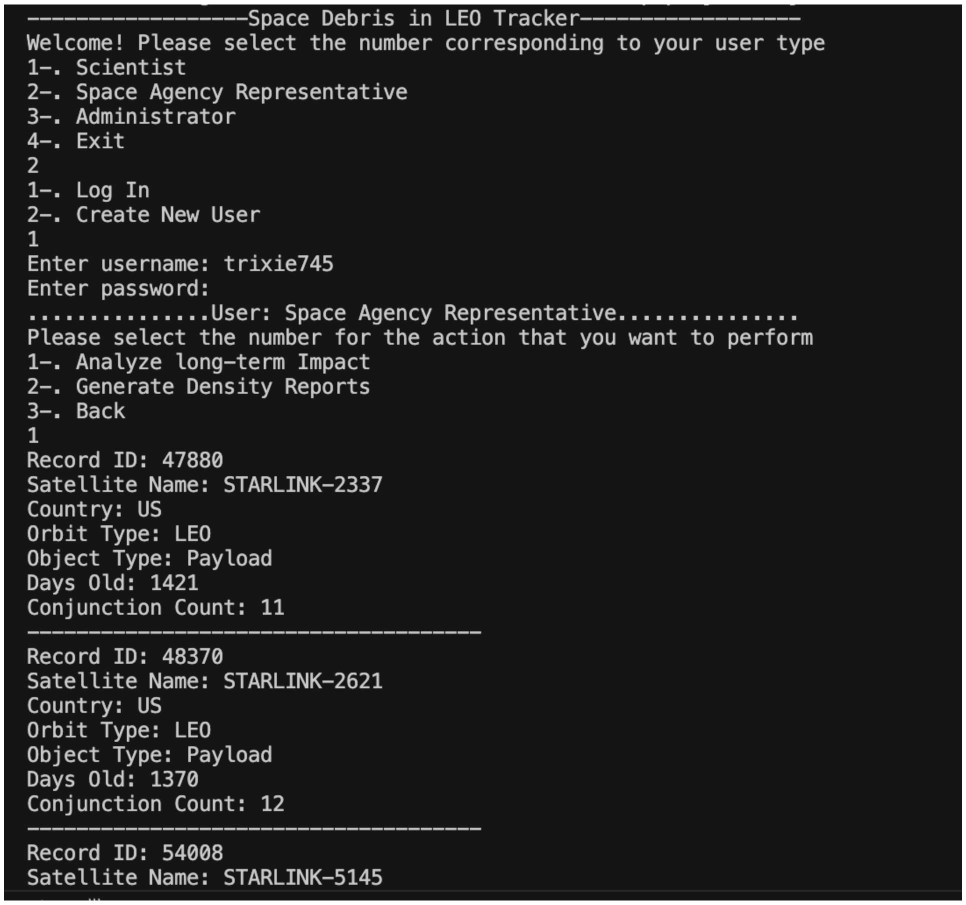
```
erikg444,IHwmxli4zmeRZDLHWm5zPXvz9Zv/DE5o6VMDwV27uUw=,PzNLZDpa1JEh17kDABvmNA==,Administrator
646gustava,ZBYdWbGw7f+/4+TEX1cIi42IaSKxHWSTq7aARv6/na0=,levmyzVpwpBQda3t3oughw==,Administrator
lola321,pyRhlc9ZdzM6IbedoYyRlmvSsCivOYB3XNkl2Vk7CRU=,4KP7bQci6uMB1E8ZEQ5Oyg==,Scientist
dagutierrez17,9GmmqsuvWcvPyazTPEHbXcwykTUMlv0lXudrziOhWbg=,w9gxhTNKm51M7AVeNQ8zpQ==,Scientist
scientist2,ZcWMVeC6kA5ZgW/OKveF3C0+KzEkfWPvFrNol7y733E=,gnoCw0j6LcEBsCeQx5KAfw==,Scientist
scientist1,1QrN8vpD8aXdjsGXF9rUG2XVtTJlAiX7IQXu1B6dTmE=,PQOCClwwiBSuRlLUPmzr1g==,Scientist
admin1,S8uPllCTBt52u1karHCcDJKzVxtTg0fEx9FAfMVsi28=,8/CgGm0jmwJmKVlxGiiTyA==,Administrator
katie123,XnAyn4oi2ObSHQ9areaXqFqaANLG5EltN/G1YoCDQMQ=,TcR9gpISMKKqvmxRXeFWhw==,Scientist
denise123,FAhoM6CrT5/oZFrpBkvU1i5CGC1AIn/2q/JNVlDPfUU=,q7cZ6DMCkEIgDisUEexnsw==,SpaceAgentRep
katie,+lN9cgE2kY2jCRpSdQU8h18w1dMb7WnChzZkyaQj124=,fQo4U3Sn5v6aoqW9CxVz7w==,SpaceAgentRep
trixie745,TKFC7cVWa4P4uzHtJr4dCVSGBg5lhzYKFR7bLZVBsPI=,VEUVbtlP7QCgx3n51PgwXw==,SpaceAgentRep
```

Density Report CSV file based on user specified longitude ranges:



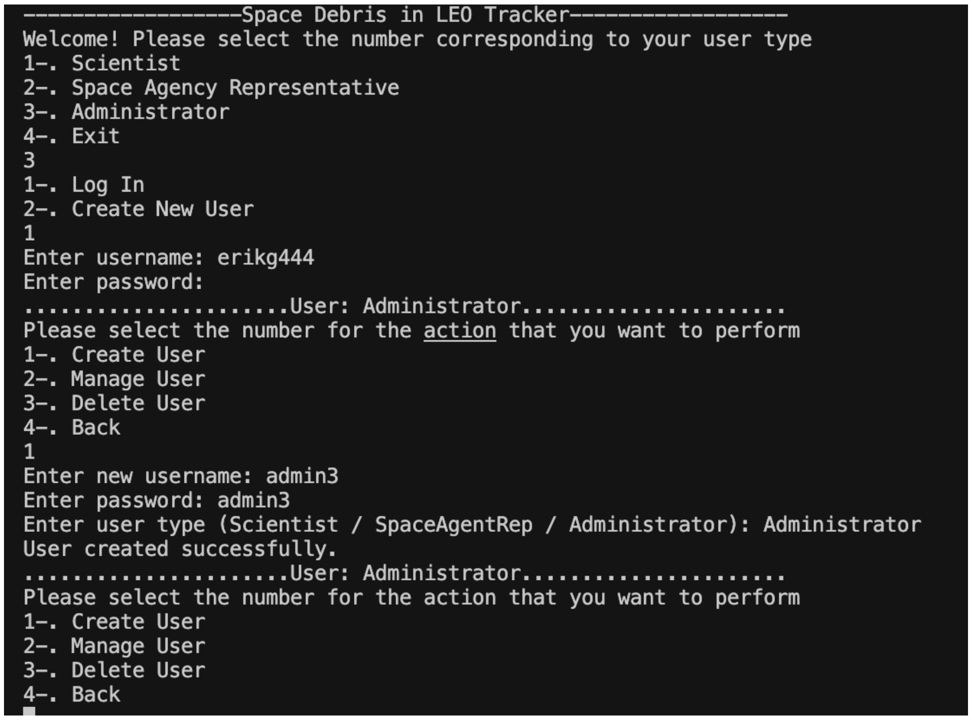
```
---- Objects in Range [10.00, 100.00] ----
Record ID,Satellite Name,Country,Orbit Type,Launch Year,Object Type
10436,COSMOS 839 DEB,CIS,LEO,1976,Debris
13610,SL-12 R/B(2),CIS,MEO,1982,RocketBody
28158,USA 176,US,GEO,2004,Payload
28473,ATLAS 5 CENTAUR R/B,US,HEO,2004,RocketBody
29539,CZ-4B DEB,PRC,LEO,2006,Debris
35670,COSMOS 2251 DEB,CIS,LEO,1993,Debris
47437,OBJECT AA,TBD,LEO,2021,UnknownObject
47960,OBJECT AE,TBD,LEO,2021,UnknownObject
48370,STARLINK-2621,US,LEO,2021,Payload
55658,STARLINK-5307,US,LEO,2023,Payload
58185,STARLINK-30793,US,LEO,2023,Payload
60336,STARLINK-32079,US,LEO,2024,Payload
41280,NOAA 16 DEB,US,LEO,2000,Debris
43341,ATLAS CENTAUR R/B,US,GEO,2018,RocketBody
54073,STARLINK-5146,US,LEO,2022,Payload
56539,STARLINK-6135,US,LEO,2023,Payload
60427,STARLINK-32254,US,LEO,2024,Payload
6797,ATLAS CENTAUR R/B,US,HEO,1973,RocketBody
18570,TVSAT 1,GER,GEO,1987,Payload
20626,SL-16 DEB,CIS,LEO,1990,Debris
2700,DELTA 1 DEB,US,LEO,1965,Debris
39166,NAVSTAR 68 (USA 242),US,MEO,2013,Payload
51634,ONEWEB-0438,UK,LEO,2022,Payload
51789,STARLINK-3639,US,LEO,2022,Payload
5181,SL-8 R/B,CIS,LEO,1971,RocketBody
53223,STARLINK-4354,US,LEO,2022,Payload
54198,STARLINK-5246,US,LEO,2022,Payload
56413,STARLINK-6258,US,LEO,2023,Payload
57329,STARLINK-6138,US,LEO,2023,Payload
Total objects in longitude range: 29

---- Objects in Range [25.00, 250.00] ----
Record ID,Satellite Name,Country,Orbit Type,Launch Year,Object Type
10436,COSMOS 839 DEB,CIS,LEO,1976,Debris
11888,SL-6 R/B(2),CIS,HEO,1980,RocketBody
13610,SL-12 R/B(2),CIS,MEO,1982,RocketBody
28158,USA 176,US,GEO,2004,Payload
28473,ATLAS 5 CENTAUR R/B,US,HEO,2004,RocketBody
29539,CZ-4B DEB,PRC,LEO,2006,Debris
```

Space Agency Rep Analyze Long Term Impact:



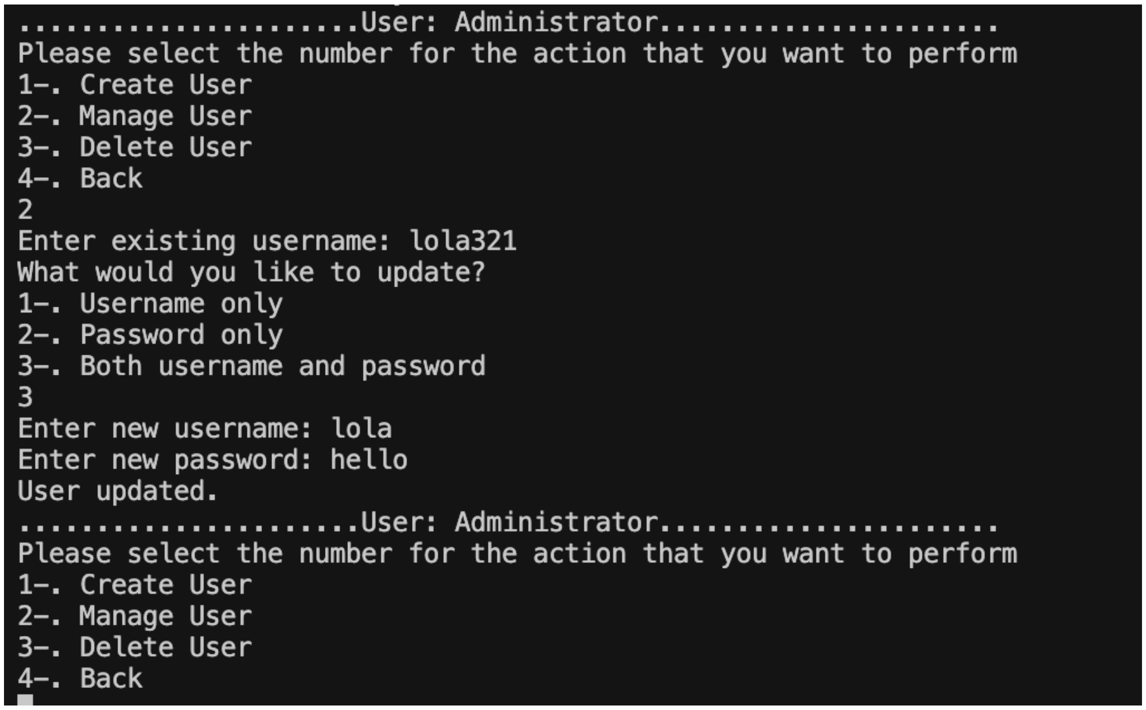
```
------------------Space Debris in LEO Tracker------------------
Welcome! Please select the number corresponding to your user type
1-. Scientist
2-. Space Agency Representative
3-. Administrator
4-. Exit
2
1-. Log In
2-. Create New User
1
Enter username: trixie745
Enter password:
...............User: Space Agency Representative...............
Please select the number for the action that you want to perform
1-. Analyze long-term Impact
2-. Generate Density Reports
3-. Back
1
Record ID: 47880
Satellite Name: STARLINK-2337
Country: US
Orbit Type: LEO
Object Type: Payload
Days Old: 1421
Conjunction Count: 11
-------------------------------------
Record ID: 48370
Satellite Name: STARLINK-2621
Country: US
Orbit Type: LEO
Object Type: Payload
Days Old: 1370
Conjunction Count: 12
-------------------------------------
Record ID: 54008
Satellite Name: STARLINK-5145
```

Administrator: Create a new user



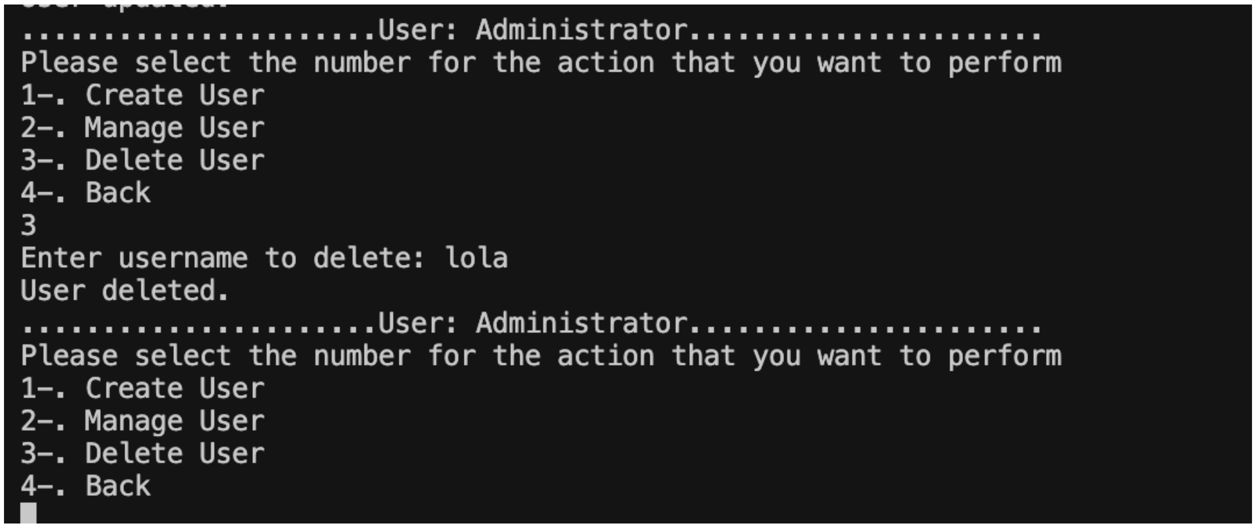
```
------------------Space Debris in LEO Tracker------------------
Welcome! Please select the number corresponding to your user type
1-. Scientist
2-. Space Agency Representative
3-. Administrator
4-. Exit
3
1-. Log In
2-. Create New User
1
Enter username: erikg444
Enter password:
......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
1
Enter new username: admin3
Enter password: admin3
Enter user type (Scientist / SpaceAgentRep / Administrator): Administrator
User created successfully.
......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
```

Update user:



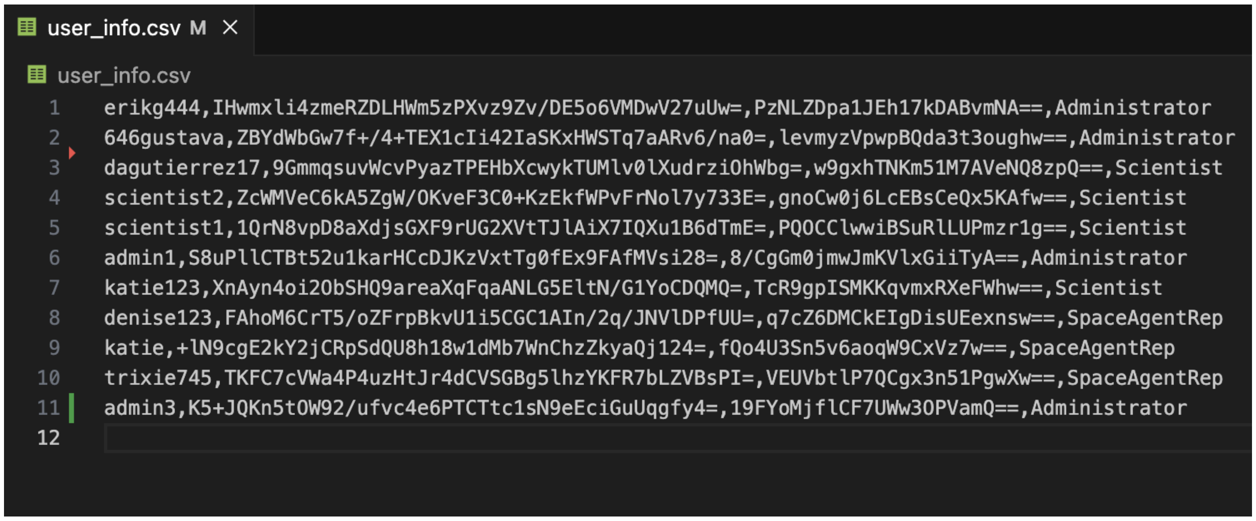
```
.......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
2
Enter existing username: lola321
What would you like to update?
1-. Username only
2-. Password only
3-. Both username and password
3
Enter new username: lola
Enter new password: hello
User updated.
.......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
```

Delete user:



```
.......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
3
Enter username to delete: lola
User deleted.
.......................User: Administrator......................
Please select the number for the action that you want to perform
1-. Create User
2-. Manage User
3-. Delete User
4-. Back
```

User_info.csv gets updated every time:



user_info.csv M

user_info.csv

```
erikg444,IHwmxli4zmeRZDLHWm5zPXvz9Zv/DE5o6VMDwV27uUw=,PzNLZDpa1JEh17kDABvmNA==,Administrator
646gustava,ZBYdWbGw7f+/4+TEX1cIi42IaSKxHWSTq7aARv6/na0=,levmyzVpwpBQda3t3oughw==,Administrator
dagutierrez17,9GmmqsuvWcvPyazTPEHbXcwykTUMlv0lXudrziOhWbg=,w9gxhTNKm51M7AVeNQ8zpQ==,Scientist
scientist2,ZcWMVeC6kA5ZgW/OKveF3C0+KzEkfWPvFrNol7y733E=,gnoCw0j6LcEBsCeQx5KAfw==,Scientist
scientist1,1QrN8vpD8aXdjsGXF9rUG2XVtTJlAiX7IQXu1B6dTmE=,PQOCClwwiBSuRlLUPmzr1g==,Scientist
admin1,S8uPllCTBt52u1karHCcDJKzVxtTg0fEx9FAfMVsi28=,8/CgGm0jmwJmKVlxGiiTyA==,Administrator
katie123,XnAyn4oi2ObSHQ9areaXqFqaANLG5EltN/G1YoCDQMQ=,TcR9gpISMKKqvmxRXeFWhw==,Scientist
denise123,FAhoM6CrT5/oZFrpBkvU1i5CGC1AIn/2q/JNVlDPfUU=,q7cZ6DMCkEIgDisUEexnsw==,SpaceAgentRep
katie,+lN9cgE2kY2jCRpSdQU8h18w1dMb7WnChzZkyaQj124=,fQo4U3Sn5v6aoqW9CxVz7w==,SpaceAgentRep
trixie745,TKFC7cVWa4P4uzHtJr4dCVSGBg5lhzYKFR7bLZVBsPI=,VEUVbtlP7QCgx3n51PgwXw==,SpaceAgentRep
admin3,K5+JQKn5tOW92/ufvc4e6PTCTtc1sN9eEciGuUqgfy4=,19FYoMjflCF7UWw3OPVamQ==,Administrator
```

6. Code Review

To ensure the completeness and reliability of our implementation, each team member performed an individual code review using the provided checklist as a guide. The review focused on several key areas:

- **Code Structure and Style**: We confirmed that the code followed consistent formatting, naming conventions, and was well-organized into modular components.

- **Object-Oriented Design**: We verified proper use of inheritance, encapsulation, and abstraction across relevant classes.

- **Documentation**: We ensured all public classes and methods included Javadoc comments, and that inline comments were used effectively for clarity.

- **Input Validation and Error Handling**: We applied exception handling to all user input scenarios and ensured robust try-catch blocks were in place for file operations.

- **Functionality**: Each method was manually tested to confirm that it performed as intended, and matched the specified behavior.

In addition to the manual code review, results from our JUnit tests helped us identify and fix unexpected behaviors in our logic. These tests were especially useful in exposing edge cases we hadn't initially accounted for. We also used this opportunity to refactor repetitive code—particularly within the menu system—and enhance the overall robustness of input validation.