INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR
DE TECNOLOGIA

First Phase Report

Computer Systems Engineering

Object-Oriented Programming

Rodrigo Ferreira Moura-27995

INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR
DE TECNOLOGIA

Introduction

Purpose of the Space System

This project was designed to simulate the operations of a modern space agency, providing a robust and flexible platform for exploring and understanding space-related missions. Developed in **C#**, the system leverages core concepts of **Object-Oriented Programming (OOP)** to model and manage various components and operations of a space agency.

The primary goal of the system is to create an educational and functional environment for simulating processes such as mission planning, rocket launches, space station management, and astronaut training. This project can serve as a tool for professionals modeling real-world scenarios or for students exploring programming concepts and space sciences.

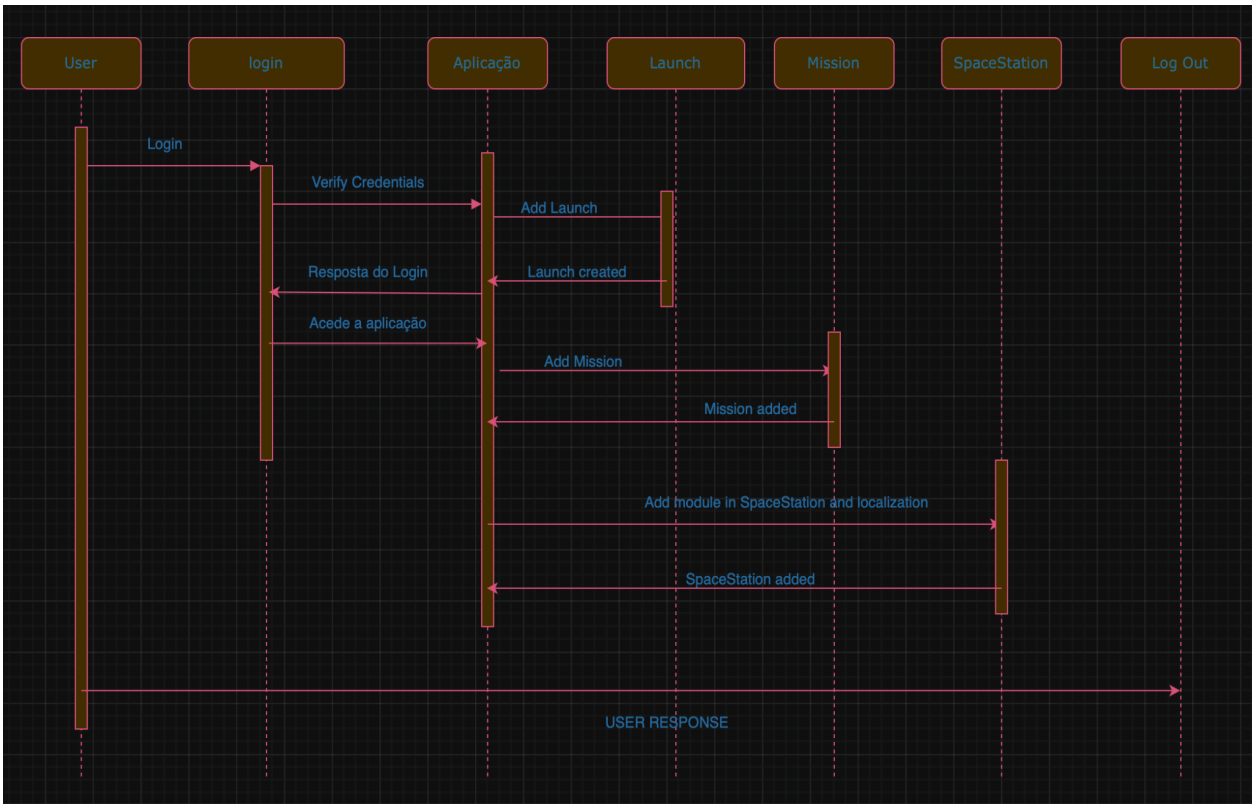
Components and Functionalities of the System

1. **Space Missions:**
   The system is designed to simulate various types of space missions. Each mission is represented by a base class (*Mission*), which defines generic attributes and methods, such as the mission's goal, destination, and duration. Specialized subclasses, like *MarsMission*, add unique characteristics, such as specific requirements for missions to Mars. This hierarchy allows the system to expand to include missions to the Moon, asteroid exploration, or even space tourism.
2. **Launch Processes:**
   One of the core features of the system is simulating rocket launches. The *Rocket* class models different types of launch vehicles, specifying parameters such as payload capacity, fuel types, and infrastructure requirements. Interfaces like *ILaunchable* allow other components, such as payloads (satellites or space modules), to be seamlessly integrated into the launch process.
3. **Space Stations:**
   The system simulates the construction and operation of space stations, modeled by

the *SpaceStation* class. Subclasses include *OrbitalSpaceStation* (designed for scientific research in orbit) and *TourismSpaceStation* (focused on space tourism experiences). These stations can host astronauts, conduct scientific experiments, and serve as intermediate points for longer missions.

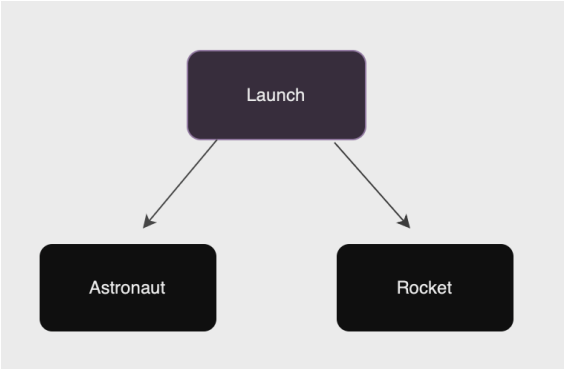
4. **Astronaut Management:**
   Astronauts are modeled as objects with attributes such as name, age, specializations, and mission experience. They can be assigned to various missions and space stations, adding customization and realism to the simulations.
5. **Planning and Management:**
   The system also incorporates strategic management elements, such as resource planning (fuel, modules, food) and mission schedules. While these aspects are still in their initial phase, they represent potential areas of expansion to incorporate economic and logistical simulation elements.

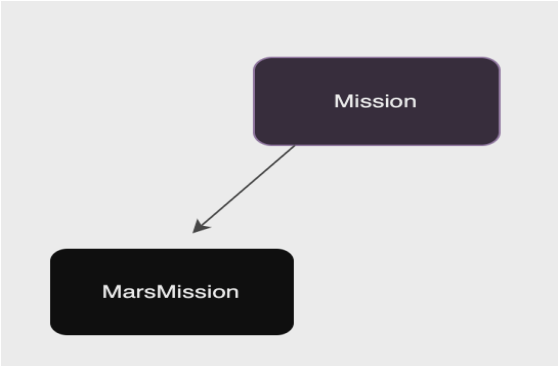
Flow Diagram

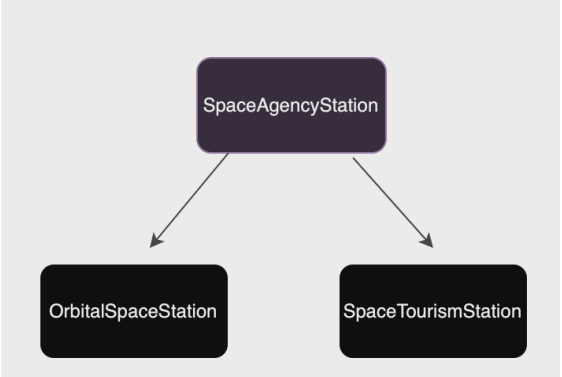


INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR
DE TECNOLOGIA

Class Structure Diagram







Implementations and Key Concepts Used in the Space Agency Project

This project uses key programming concepts to create a well-organized and expandable simulation of space agency operations. Here's an overview of the main ideas:

1. **Modular Design**:
   The project is divided into folders and classes like **`Launch`**, **`Mission`**, and **SpaceAgencyStation**. Each section has a clear purpose, making the code easier to understand and expand.

2. **Interfaces for Flexibility**:
   Interfaces, like `ILaunchable` and `ILaunchableMission`, define shared behaviors that different classes can use. For example, both rockets and missions can be "launchable" because they follow the same rules set by these interfaces. This makes it easy to add new launchable items in the future.

3. **Encapsulation(private and public classes)**:
   Each class (e.g., `Rocket`, `SpaceStation`) has its own data and functions. This keeps complex behaviors contained within each class, which makes it easier to troubleshoot and update.

4. **Inheritance**:
   Some classes are special versions of others. For instance, `MarsMission` inherits from `Mission`, adding Mars-specific details, and `OrbitalSpaceStation` is a type of `SpaceStation`. This allows us to create specific versions without rewriting everything.

5. **Organized File Structure**:
   Each part of the project has its own folder (e.g., `Launch`, `Mission`), which makes it easy to navigate and keep things organized.

6. **Easy to Expand**:
   The project's structure allows for new missions, types of stations, or other features to be added without major changes. For example, adding a new type of space station would be simple because of the existing `SpaceStation`class structure.

7. **Enums:**

   The `AstronautRole` enum represents different roles astronauts can take on during a mission, such as Commander, Pilot, or Scientist. Each role corresponds to specific tasks or expertise required for mission success. Using enums ensures consistency across the system and reduces errors by restricting values to a predefined list.

Future Projects

Looking ahead, we plan to make significant improvements in the next phase of our project. In Phase 2, we will introduce Windows Forms for a more intuitive user interface, along with the integration of a database to enhance data management and functionality. Additionally, we will refine the class structure and subclasses to improve the system's scalability and performance.

We will also focus on enhancing the project documentation, ensuring it is comprehensive and easier for future developers to understand. A universal programming language will be adopted to further streamline development and improve collaboration across teams.

These updates will ensure that the project not only meets current requirements but is also flexible and robust enough to support future growth.

Conclusion

This project demonstrated the power of object-oriented programming in designing and implementing a space system. By leveraging classes and modular design, we were able to create an organized framework for simulating and managing the various components of space missions.

The use of object-oriented principles allowed for a clear separation of responsibilities, making the system both scalable and easy to maintain. This approach also facilitated future enhancements and modifications, ensuring that the system remains flexible as new features or components are added.

In conclusion, the project showcased how a structured, class-based design can effectively model and manage complex systems, providing a simple yet powerful solution for simulating space mission dynamics.