Report on the Development of an Arcade Video Game Machine Application

Santiago Alejandro Guada Bohorquez

Software Modeling I

Carlos Andrés Sierra

Universidad Distrital Francisco José de Caldas

October 5 , 2024

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

1. Introduction

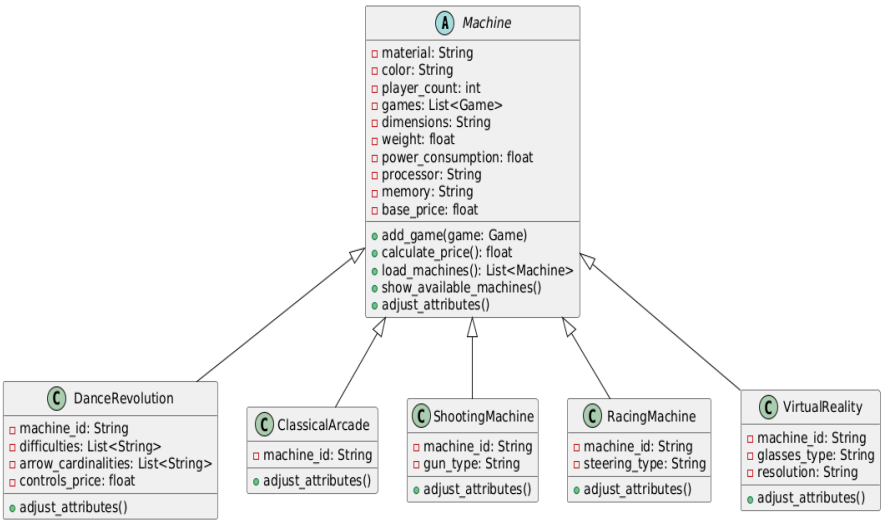
The purpose of this report is to provide a comprehensive overview of the architecture and design of an arcade machine management system. This system is designed to facilitate the operation and management of various types of arcade machines, allowing users to add and manage games, calculate pricing based on machine attributes, and ensure efficient data handling through JSON file integration.

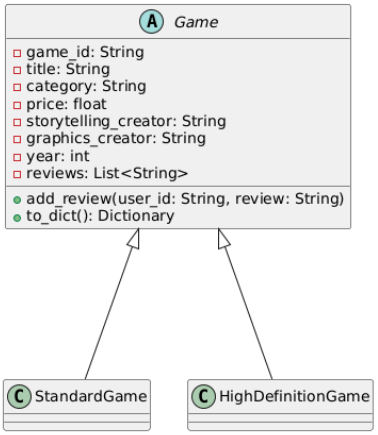
The arcade machine system is built around an object-oriented approach, utilizing abstract classes and concrete implementations to represent different types of machines and games. Key components include various machine classes such as DanceRevolution, ClassicalArcade, and ShootingMachine, each tailored to specific functionalities and user experiences. The system also incorporates a game management feature, enabling the addition of games with distinct characteristics, including pricing, storytelling, and graphics.

In this report, we will detail the class structure through a Diagrama de Clases, highlighting the relationships between components. Additionally, we will discuss the technical decisions made during the design process, including the application of design patterns and adherence to SOLID principles. These considerations are crucial for ensuring that the system is maintainable, scalable, and robust in meeting user needs.

By documenting these aspects, this report aims to serve as a foundational resource for understanding the architecture of the arcade machine system and guiding future enhancements or modifications.

2. Class Diagrams





3. Decisions on Design Patterns and SOLID Principles

In developing the arcade machine system, several design patterns and principles were considered to ensure a robust, maintainable, and scalable architecture.

Design Patterns Used

- Factory Pattern: The system employs a form of the Factory pattern through the abstract classes Machine and Game. This allows for the creation of various machine and game types without exposing the instantiation logic to the client. For example, subclasses like DanceRevolution and HighDefinitionGame can be instantiated based on specific requirements without modifying existing code.

SOLID Principles

- Single Responsibility Principle (SRP): Each class has a single responsibility. For instance, the Machine class is responsible for managing machine attributes and game lists, while the Game class handles game-related functionalities such as reviews and pricing.
- Open/Closed Principle (OCP): The design allows for extending functionality without modifying existing code. New machine types or game types can be added by creating new subclasses of Machine or Game, respectively, without altering the base classes.
- Liskov Substitution Principle (LSP): Subtypes can be substituted for their base types without affecting the correctness of the program. For example, any subclass of Machine can be treated as a Machine, ensuring that methods like add_game() will work seamlessly regardless of the specific machine type.
- Interface Segregation Principle (ISP): The system avoids forcing classes to implement interfaces they do not use by defining abstract methods in a way

that only relevant subclasses implement them. For instance, only game classes implement methods related to reviews.

- Dependency Inversion Principle (DIP): High-level modules (like those managing user interactions) depend on abstractions (Machine, Game) rather than concrete implementations. This reduces coupling between different parts of the system and enhances testability.

4. Conclusion

In conclusion, the development of the arcade management system represents a significant advancement in the management and operation of various types of arcade machines. By using an object-oriented approach, the system effectively encapsulates machine and game functionalities, allowing for efficient management and scalability.

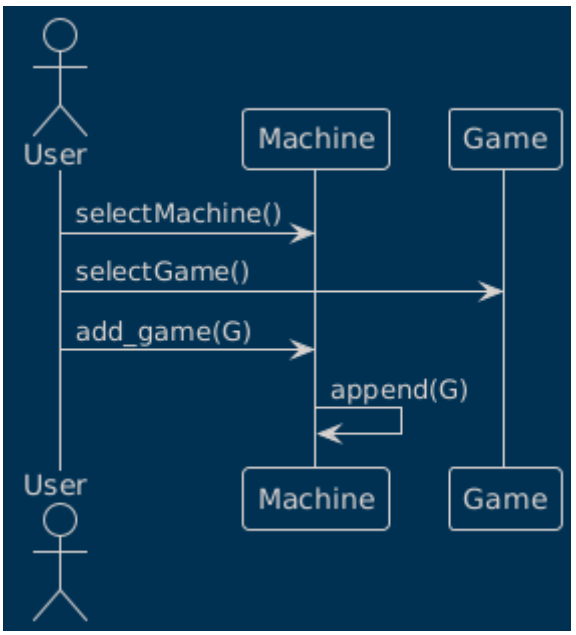
The architecture is based on key design patterns such as Factory, which facilitates the creation of various types of machines and games while maintaining flexibility in their behaviors. This modular design not only improves code reuse, but also simplifies future extensions or modifications.

Adherence to SOLID principles has been a guiding factor throughout the development process. Each class is designed with a single responsibility, ensuring that changes in one area do not negatively impact the others. The open/closed principle allows for easy addition of new features without altering existing code, while the Liskov substitution principle ensures that derived classes can seamlessly replace their base classes.

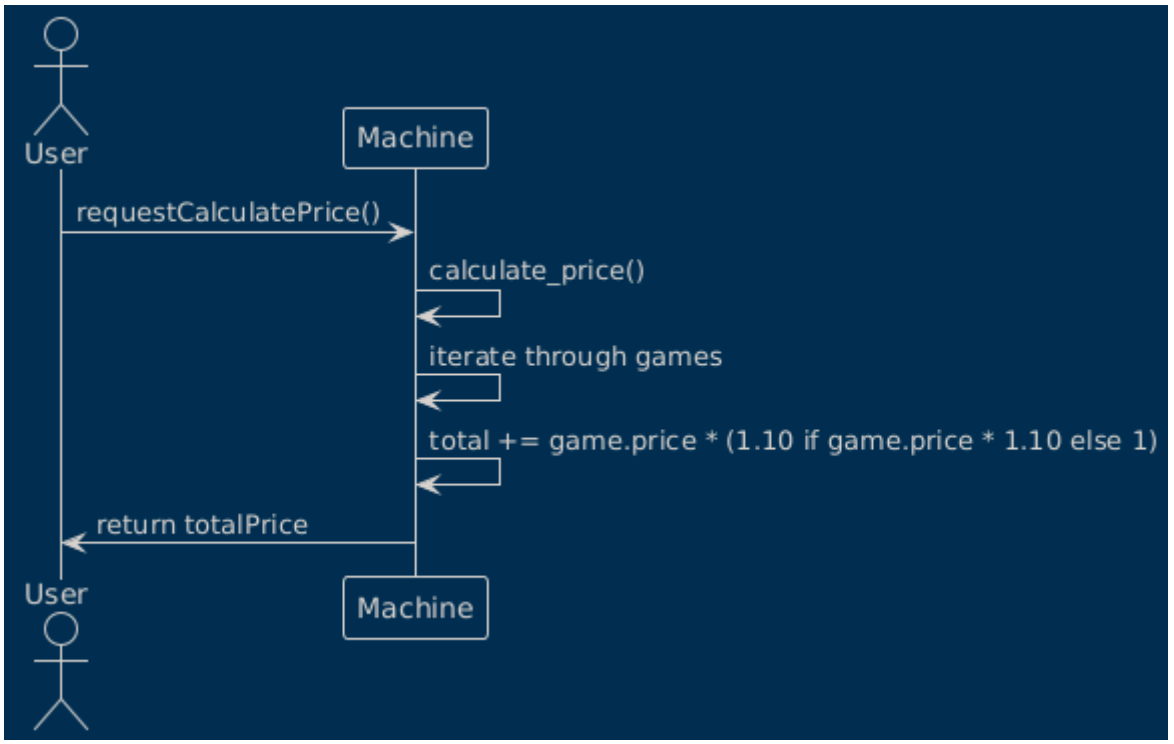
The implementation of this system demonstrates a commitment to high-quality software engineering practices. By leveraging established design patterns and principles, the AMS is positioned to meet current user needs while remaining adaptable for future enhancements. This report serves as a foundational document for understanding the system architecture and provides insight into potential areas for future development.

5.Appendices

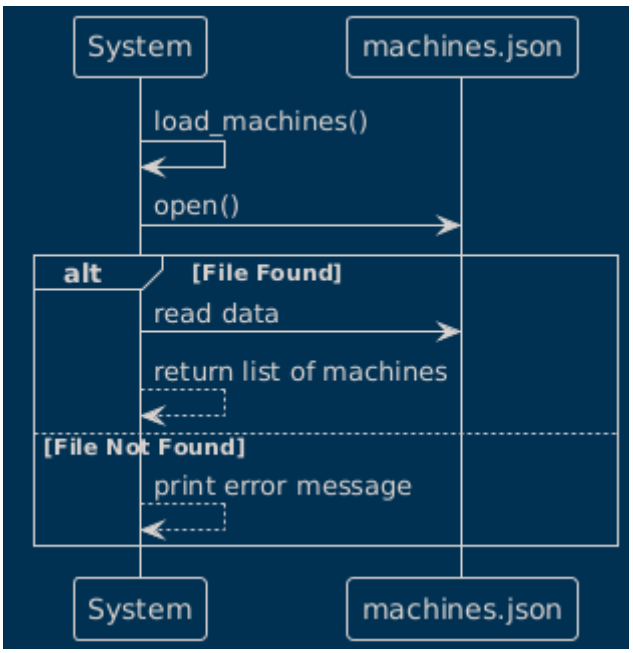
Adding a Game to a Machine



Calculating Total Price of a Machine



Loading Machines from JSON



Displaying Available Machines