Software Modeling I

Season 2024-III Workshop No. 1 — Object-Oriented Programming

David Felipe García León - 20231020202

Universidad Distrital Francisco José de Caldas

REPORT

1. User stories:

- As a **buyer**, I want to choose the material for the machine (wood, aluminum, or carbon fiber), so that I can select a durable and stylish arcade machine.
- As a **gamer**, I want to view the list of available games, so that I can choose which ones I want to play on my machine.
- As a **gamer**, I want to add games to the machine using a code, so that I can expand my game library easily.
- As a **buyer**, I want to finalize the purchase and provide my delivery information, so that the machine can be delivered to my location.
- As a **buyer**, I want to choose the color of the arcade machine, so that it matches my personal style and home setup.
- As a **gamer**, I want an easy-to-use interface, so that I can quickly navigate through the available games and settings without confusion.
- As a **buyer**, I want to know the benefits of each material (wood, aluminum, or carbon fiber), so that I can make an informed decision about durability and aesthetics.
- As a **gamer**, I want the available games to include both classics and modern titles, so that I have a diverse range of options to choose from.
- As a **gamer**, I want to see descriptions of the games before adding them using a code, so that I can make sure they are worth purchasing.
- As a **buyer**, I want to have the option to review my order before finalizing the purchase, so that I can ensure everything is correct before paying.
- As a **buyer**, I want to have access to customer support during the purchasing process, so that I can resolve any issues before completing my order.
- As a **buyer**, I want to know the warranty options for the arcade machine, so that I can feel confident that my purchase is protected.
- As a **gamer**, I want the controls and setup to be intuitive, so that I can start playing without needing a complex manual.
- As a **gamer**, I want to be able to adjust the settings for sound and display, so that I can personalize my gaming experience according to my preferences.

- As a **gamer**, I want to easily switch between games without having to restart the system, so that I can enjoy a seamless gaming experience.

2. object-oriented principles análisis:
    1. **Abstraction:**
        - The project defines abstract concepts using abstract classes, such as Game, AbstractArcadeMachine, and User. These classes provide a high-level structure without implementing concrete details, allowing child classes to implement the specific behavior.
        - The use of the ABC module and abstractmethod decorators clearly enforces abstraction by ensuring that certain methods must be overridden in subclasses, like select_material, select_color, and add_game in AbstractArcadeMachine.
    2. **Encapsulation:**

        - The project encapsulates data within classes, such as Game and RetroArcadeMachine. Each object has its attributes (like name, code, material, and color) and methods (like select_material, add_game, etc.) that control how these attributes are accessed and modified.
        - By keeping data (attributes) and behavior (methods) within the same class, the project hides internal details and only exposes what's necessary. For example, games are added using the add_game method, which checks how many games are already in the arcade machine before allowing another one.
    3. **Inheritance:**
        - The inheritance relationship is clear between the AbstractArcadeMachine and RetroArcadeMachine classes. The RetroArcadeMachine inherits from the AbstractArcadeMachine class, thus gaining its attributes and abstract methods.
        - The RetroArcadeMachine class overrides the abstract methods (select_material, select_color, add_game) to provide specific behavior for a retro arcade machine, demonstrating the use of inheritance to specialize a more general class.
        - User also serves as a base class for future user types that could be derived.
    4. **Polymorphism:**
        - The project utilizes polymorphism through abstract methods. Since methods like select_material, select_color, and add_game are defined in the abstract AbstractArcadeMachine, multiple subclasses can implement these methods differently, allowing for various types of arcade machines.
        - This means a RetroArcadeMachine and any future types of arcade machines would share the same interface but could behave differently when these methods are called, supporting runtime polymorphism.

- For instance, if a new class ModernArcadeMachine is added in the future, it can still implement these methods in its own way while maintaining the interface.

3. CRC cards:

GAME	
RESPONSIBILITIES	COLLABORATORS
- Initialize a game with name and code.	- None

ABSTRACTARCADEMACHINE	
RESPONSIBILITIES	COLLABORATORS
- Initialize material, color, games. - Define abstract methods for material, color selection, and adding games. - Show available games.	- Game - RetroArcadeMachine

RETROARCADEMACHINE	
RESPONSIBILITIES	COLLABORATORS
- Concrete implementation of material, color selection. - Add games (max 5 games).	- AbstractArcadeMachine - Game

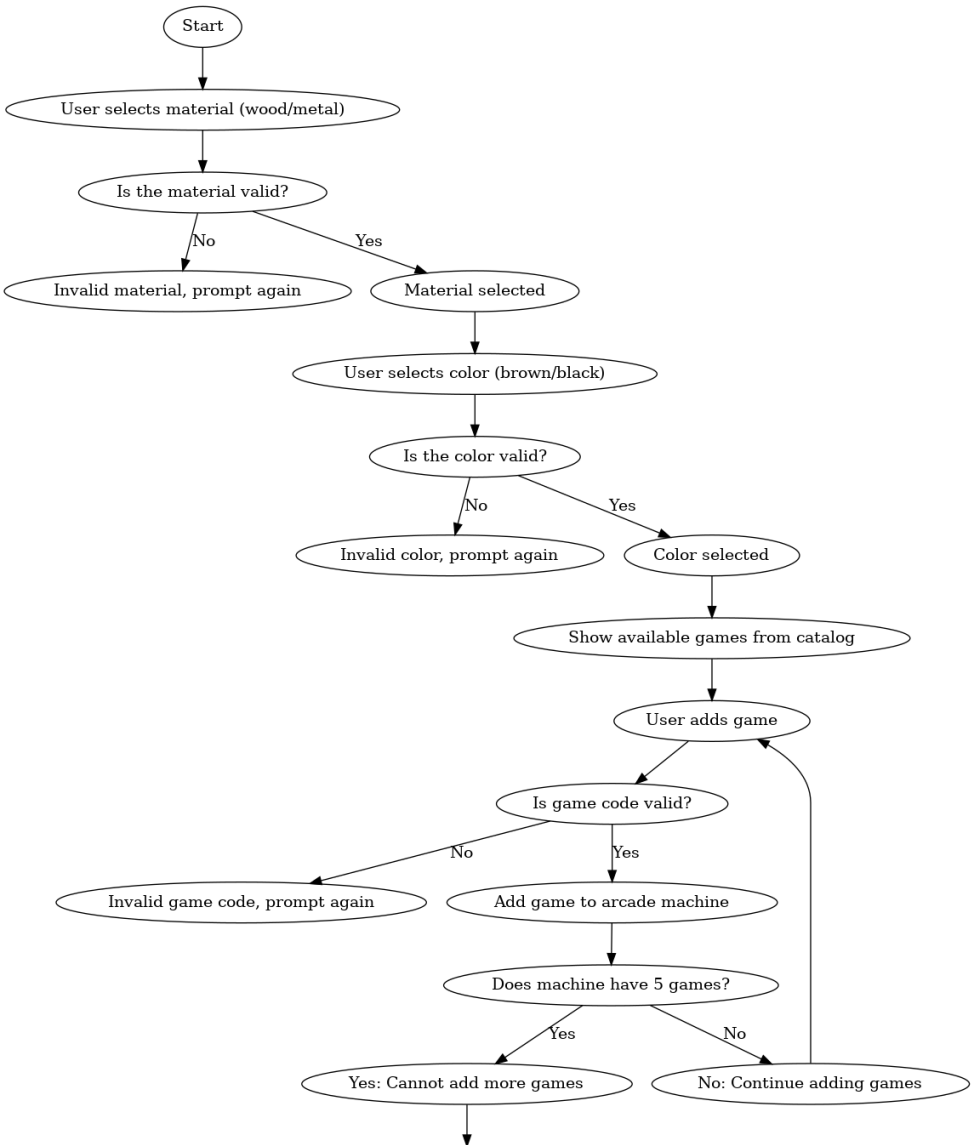
USER	
RESPONSIBILITIES	COLLABORATORS

- Initialize user with name and address.	- None

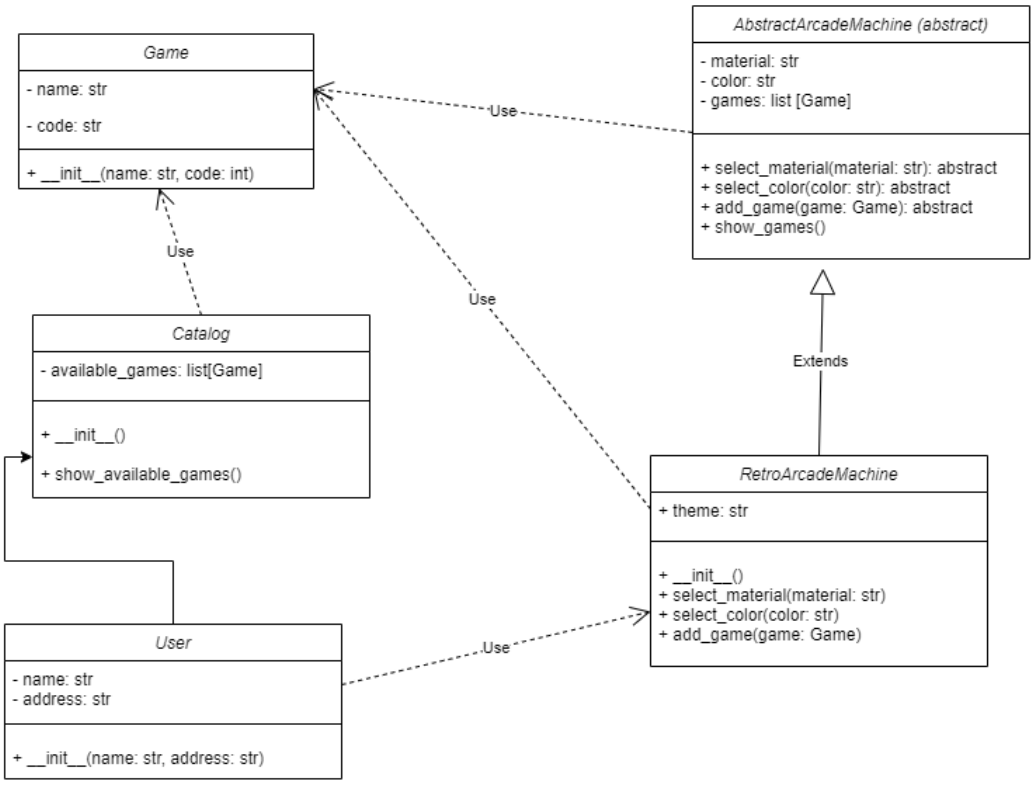
CATALOG	
RESPONSIBILITIES	COLLABORATORS
- Initialize with a list of games. - Show available games in the catalog.	- Game

4. Diagrams.

Activity Diagram



Class Diagram:



Sequence Diagram