Software-Modeling

Julian David Celis Giraldo – 20222020041

1. I have created a form in Microsoft Forms, with the answers I have recompiled the answers and with it I made the user stories
2. List User Stories:

    2.1 Clear game specifications: As a buyer, I want to see what games are included with the machine, to make sure I am interested in the titles.

    2.2 Joystick and button controls: As a buyer, I want the machine to have joysticks as well as buttons, for a more authentic gaming experience.

    2.3 Multi-console compatibility: As a buyer, I want to know which consoles the machine can emulate, so that I can play titles from different generations.

    2.4 Visual description of the design: As a buyer, I want the program to show me a description of the design of the machine, including color, material and lights when I finish setting it up, so I can imagine how it will look once it is installed in my space.

    2.5 Optimal machine condition:** As a buyer, I want to know that the machine is in good condition and has music and lights, for a more immersive experience.

    2.6 Strong but lightweight material: As a buyer, I want the machine to be made of strong but lightweight materials, so that it is easy to move and durable.

    2.7 Color customization: As a buyer, I want to be able to customize the colors of the controls and the housing of the machine to reflect my personal style.

    2.8 Sound customization options: As a buyer, I want to have the option of choosing different types of sound or music for the machine to suit my listening preferences.

    2.9 Modular design: As a buyer, I want the machine to have a modular design, so that I can easily upgrade or replace components.

    2.10 Support for software upgrades: As a buyer, I want the machine to allow software upgrades for new games or improvements, to keep it up to date over time.

2.11 Customizable lighting options: As a buyer, I want to be able to adjust the lights on the machine, both in color and intensity, to create different environments according to my preference.

2.12 Tactile feedback system: As a buyer, I want the machine to have a tactile feedback system on the controls for a more immersive and realistic gaming experience.

2.13 Add games by code: As a buyer, I want to be able to add games to the machine by code to customize my game selection.

Object-oriented principles analysis

Object Oriented Analysis

1. Classes and Relationships

1. Material (Enum)

- **Purpose**: Represents the different materials that can be used to build an arcade machine.
- **Attributes:** WOOD, ALUMINUM, CARBON_FIBER
- **Responsibility**: Provide a list of predefined materials for use in arcade machines.

2. **Color (Enum)**
   - **Purpose:** Defines the colors available for the arcade machine and its lights.
   - **Attributes:** BLACK, WHITE, RED, BLUE, GREEN, YELLOW, PURPLE, NONE
   - **Responsibility: Provide a list of colors that can be applied to arcade machines.**
3. **ArcadeMachine (Abstract Class)**
   - **Purpose:** Defines general behavior for arcade machines, which should be implemented by concrete classes.
   - **Attributes:**
     - _material (Material)
     - _color (Color)
     - _lights (Color)
     - _sound (str)
     - _games (games list)

- **Métodos:**
    - show_available_games(): Abstract method to display the games available on the machine.
    - is_game_valid(game): Abstract method to verify if a game is valid for this machine.
    - add_game(game): Method to add a game to the machine's game list.
    - show_info(): Method for displaying detailed information about the machine.

4. **ModernArcadeMachine (Concret Class)**
    - Purpose: It represents a modern arcade machine with specific features.
    - **Attributes:**
        - _controls (str): Defines modern controls.
    - **Methods:**
        - show_available_games(): Implementation of the abstract method for displaying modern games.
        - is_game_valid(game): Implementation of the abstract method to verify if a game is modern.
        - show_info(): Displays information about the modern machine including controls.
5. **RetroArcadeMachine (Concrete Class)**
    - **Purpose:** It represents a retro arcade machine with specific features.
    - **Attributes:**
        - _controls (str): Define the retro controls.
    - **Métodos:**
        - show_available_games(): Implementation of the abstract method to display retro games.
        - is_game_valid(game): Implementation of the abstract method to verify if a game is retro.
        - show_info(): Displays information about the retro machine including controls.
6. **Game (Class)**
    - Purpose: Represents a game in the arcade game catalog.

- **Atributos:**
  - available_games (list of available games).
  - title (str): Game title.
  - code (str): Unique code of the game.
  - type (str): Type of game (modern or retro)
- **Methods:**
  - show_available_games(machine_type): Displays the games available for a specific type of arcade machine.

7. **Customer (Class)**
   - **Purpose:** Represents a customer with his personal information.
   - **Attributes:**
     - name (str): Customer's name
     - address (str): Adress' Customer
     - phone (str): Customer's phone number
   - **Method:**
     - __str__(): Returns a string representation of the customer's information.
8. **ArcadeCatalog (Class)**
   - **Purpose:** Manages the arcade machine catalog and handles customer interaction.
   - **Attributes:**
     - _cart (ArcadeMachine): The arcade machine selected for purchase.
     - _customer (Customer): Customer information.
     - _machine_type (str): Type of machine (modern or retro).
   - **Methods:**
     - add_to_cart(machine_type): Customize and add an arcade machine to the cart.
     - choose_machine(machine_type): Select the type of arcade machine and start the customization.
     - customize_material(material_option): Customize the machine material.

- customize_color(color_option): Customize the color of the machine.
- customize_light_color(lights_option): Customize the color of the machine lights.
- get_option(category, options): Displays options and allows you to select one.
- get_sound(): Allows you to select whether sound is desired on the machine.
- show_games(): Displays the games available for the selected machine.
- add_game_by_code(code): Adds a game to the selected machine by code.
- add_games(): Allows to add games to the machine by code.
- complete_purchase(name, address, phone): Completes the purchase by saving the customer's information and displaying the final details.

Relaciones y Diseño

- **- Inheritance: ModernArcadeMachine and RetroArcadeMachine inherit from ArcadeMachine, specializing and extending its functionality.**
- **Composition: ArcadeCatalog contains instances of ArcadeMachine and Customer to manage the purchase and customization of arcade machines.**
- **Enumerations:** The Material and Color classes are used to define attributes of arcade machines in a consistent manner.

CRC card (Class Responsibility Collaborator): The cars were made in Draw.io

Color	
Define colors for arcade machines and lights. Provide standardized list of color options.	ArcadeMachine ModernArcadeMachine RetroArcadeMachine

ArcadeMachine	
Represent common features and behaviors of arcade machines. Manage attributes like material, color, lights, sound, and games. Provide abstract methods for subclasses.	ModernArcadeMachine RetroArcadeMachine Game ArcadeCatalog

game	
Represent a game in the arcade catalog. Add games to the catalog. Display available games for specific machine types.	ArcadeMachine ModernArcadeMachine RetroArcadeMachine ArcadeCatalog

RetroArcadeMachine	
Represent retro arcade machines. Provide list of retro games. Validate games for retro machines.	ArcadeMachine Game ArcadeCatalog

Customer	
Store customer information. Display customer details.	ArcadeCatalog

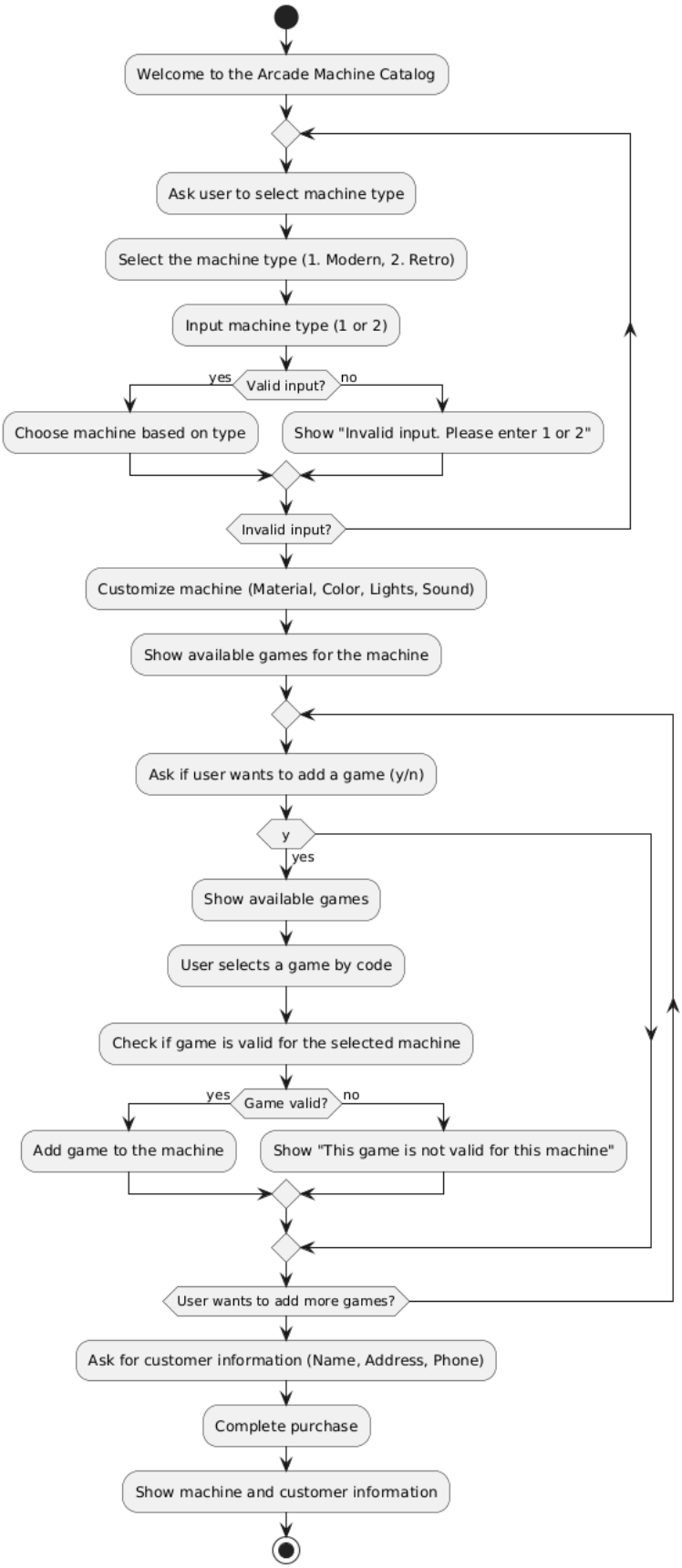
ModernArcadeMachine	
Represent modern arcade machines. Provide list of modern games. Validate games for modern machines.	ArcadeMachine Game ArcadeCatalog

ArcadeCatalog	
Manage arcade machine catalog. Handle customer interactions. Customize and purchase machines. Add games to machines.	ArcadeMachine ModernArcadeMachine RetroArcadeMachine Game Customer

Material	
Define materials for arcade machines. Provide standardized list of material option	ArcadeMachine ModernArcadeMachine RetroArcadeMachine

Diagrams: The diagrams were developed in the tool planttext.com

Activity Diagram:


Welcome to the Arcade Machine Catalog
Ask user to select machine type
Select the machine type (1. Modern, 2. Retro)
Input machine type (1 or 2)
Valid input?
yes
no
Choose machine based on type
Show "Invalid input. Please enter 1 or 2"
Invalid input?
Customize machine (Material, Color, Lights, Sound)
Show available games for the machine
Ask if user wants to add a game (y/n)
y
yes
Show available games
User selects a game by code
Check if game is valid for the selected machine
Game valid?
yes
no
Add game to the machine
Show "This game is not valid for this machine"
User wants to add more games?
Ask for customer information (Name, Address, Phone)
Complete purchase
Show machine and customer information

Secuence Diagram:

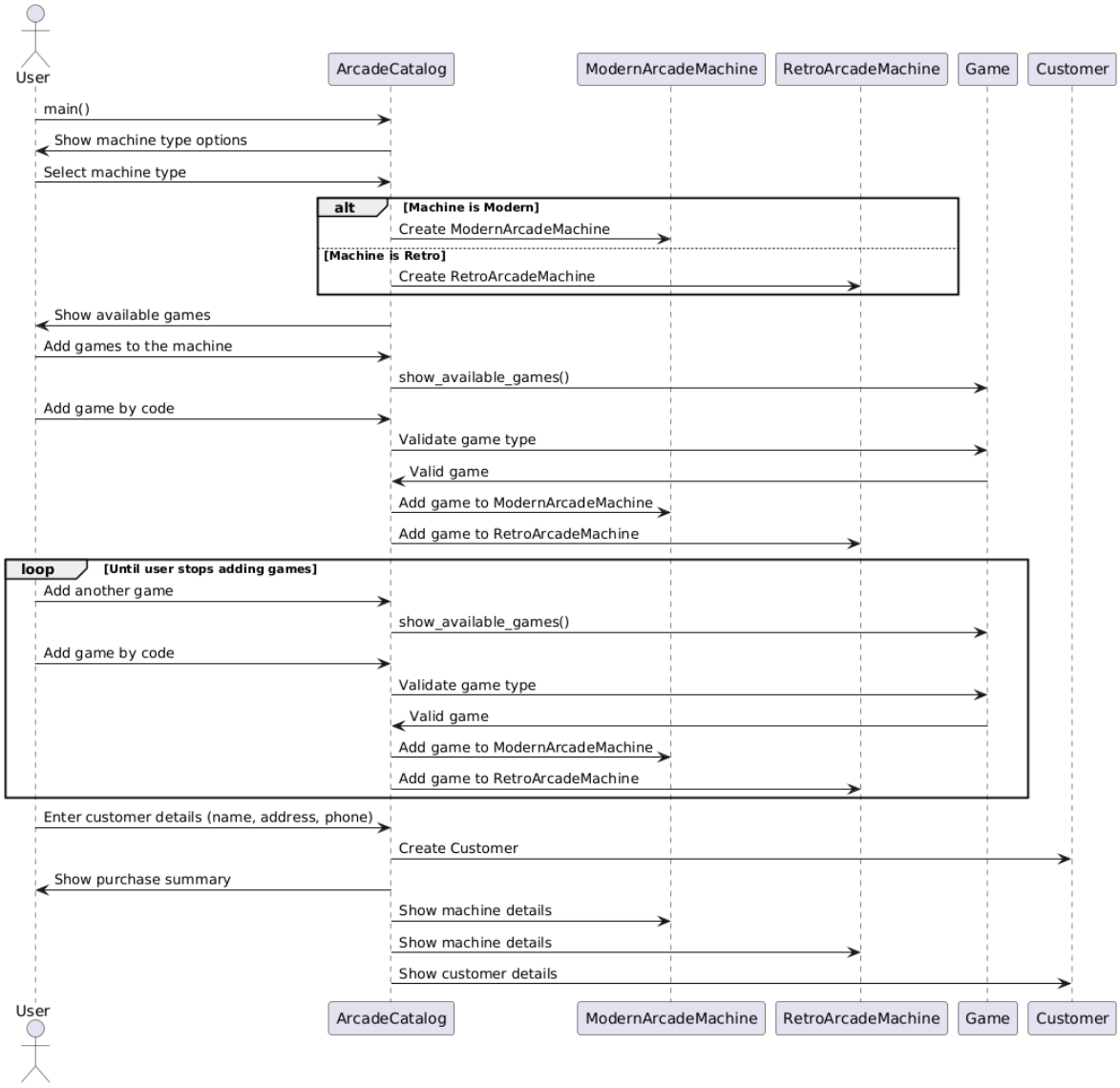
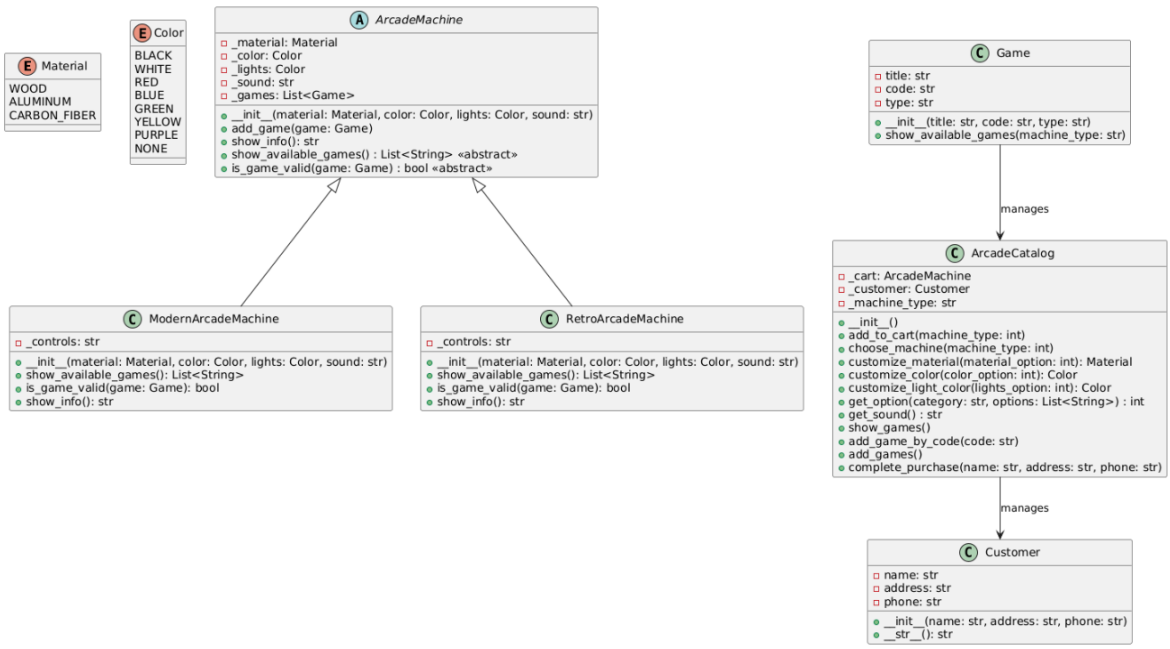


Diagrama Class



Enumerations (Material, Color): Define the materials and colors that can be used for arcade machines.

Abstract Class (ArcadeMachine): Basic attributes and methods for an arcade machine, including abstract methods that must be implemented by the concrete subclasses. Concrete Classes (ModernArcadeMachine, RetroArcadeMachine): Implement the abstract methods of ArcadeMachine and add attributes and behaviors specific to modern and retro machines, changing the types of controls depending on the version of the machine. Game Class: Represents a game in the catalog, with attributes such as title, code, and type.

Customer Class: Stores customer information to complete the purchase, such as name, address, and phone number.

ArcadeCatalog Class: Manages the arcade machine catalog and customer interaction, including machine customization and game handling.