Workshop 2

Jhomar Armando Bojaca Landinez

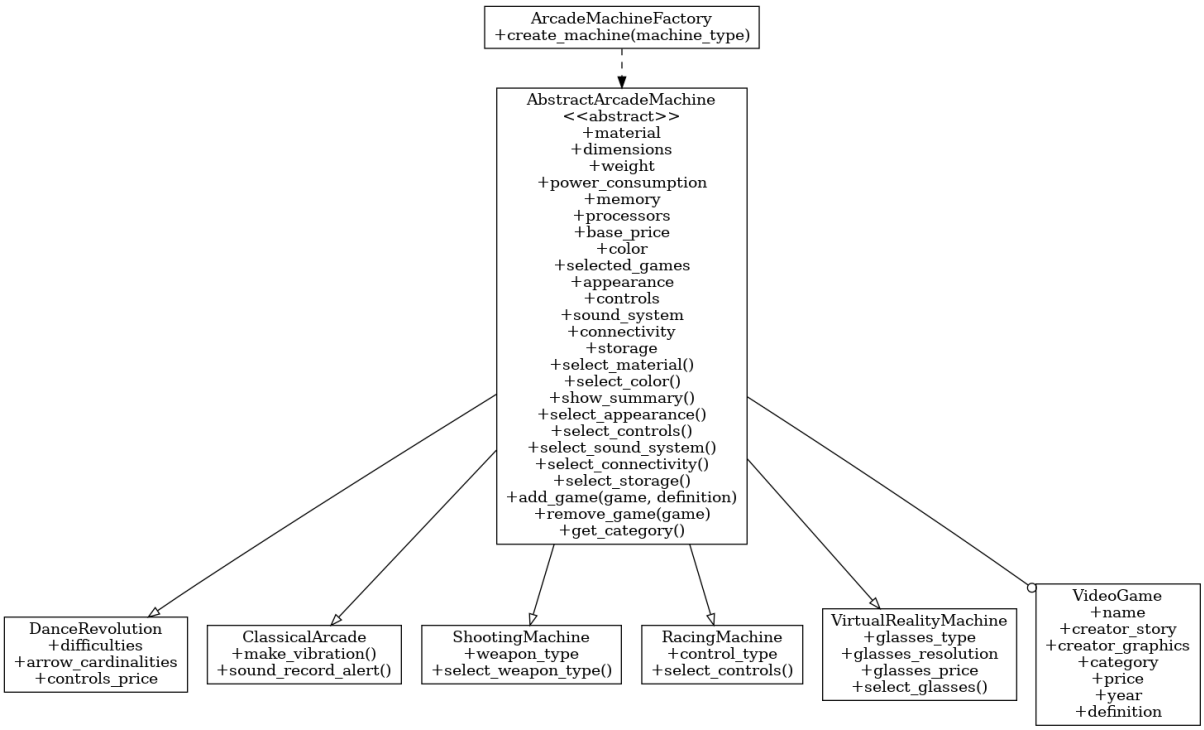
Carlos Andrés Sierra

Universidad Distrital Francisco José de Caldas

October 5 , 2024

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

1. Class Diagram



2. Design Pattern Analysis

2.1 Factory Method

The Factory Method pattern is implemented through the ArcadeMachineFactory class. This pattern enables the creation of different types of arcade machines without specifying the exact class of the object to be created.

Advantages:

Flexibility to add new machine types without modifying existing code.

Encapsulates object creation logic.

2.2 Template Method

The Template Method pattern is implemented in the abstract class AbstractArcadeMachine. It defines the skeleton of an algorithm in the superclass but allows subclasses to override specific steps without changing the algorithm's structure.

Advantages:

Code reuse.

Flexibility for subclasses to implement specific behaviors.

3. SOLID Principles Analysis

3.1 Single Responsibility Principle (SRP)
Compliance: Partial
Each machine class has specific responsibilities, but the AbstractArcadeMachine class handles multiple responsibilities.
Proposed improvement: Separate game management into a different class.

3.2 Open/Closed Principle (OCP)
Compliance: Good
The architecture allows new machine types to be added without modifying existing code.
3.3 Liskov Substitution Principle (LSP)
Compliance: Good
Subclasses of AbstractArcadeMachine can be used in place of the base class without affecting the program's behavior.
3.4 Interface Segregation Principle (ISP)
Compliance: Partial
The AbstractArcadeMachine interface is quite large and could be split into more specific interfaces.
Proposed improvement: Create separate interfaces for game management, hardware configuration, etc.

3.5 Dependency Inversion Principle (DIP)
Compliance: Good
Using AbstractArcadeMachine as a base type ensures that high-level modules do not depend on low-level modules, but rather on abstractions.