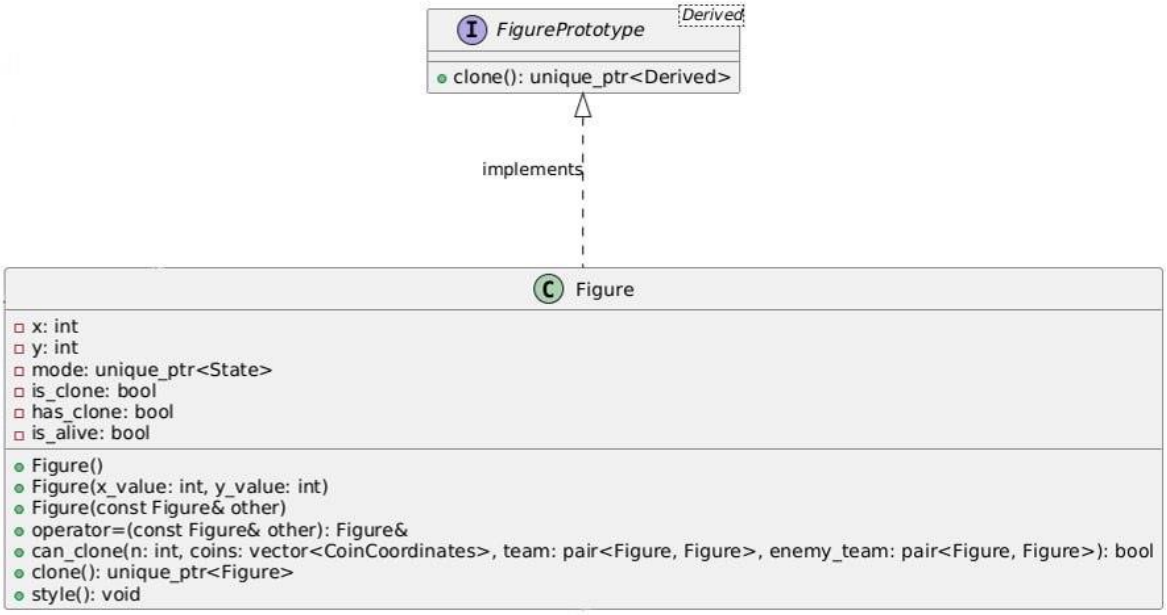
Implementation of UML Diagram

Assigment 3. Evdokimova Daria, CSE-01

To improve quality of code for this task I use 3 suitable design patterns from lectures.

1. Creational pattern – Prototype

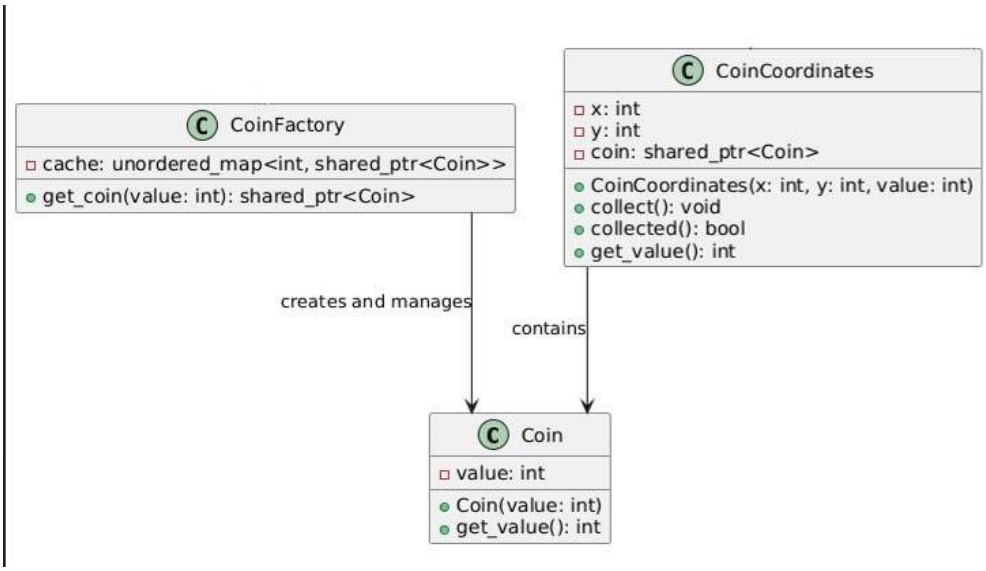
According to the terms of the assignment, we can clone the figures of the red and green teams. Accordingly, the Prototype pattern coped with this task best, because it allows you to flexibly copy existing objects, preserving their state and avoiding complex initialization logic, which is especially important in a dynamically changing game environment.



I create *FigurePrototype* interface, which produce *clone()* method for cloning figure. Also I add boolean values such as *is_clone* and *has_clone* to check if we can't clone some figure.

2. Structural pattern - Flyweight

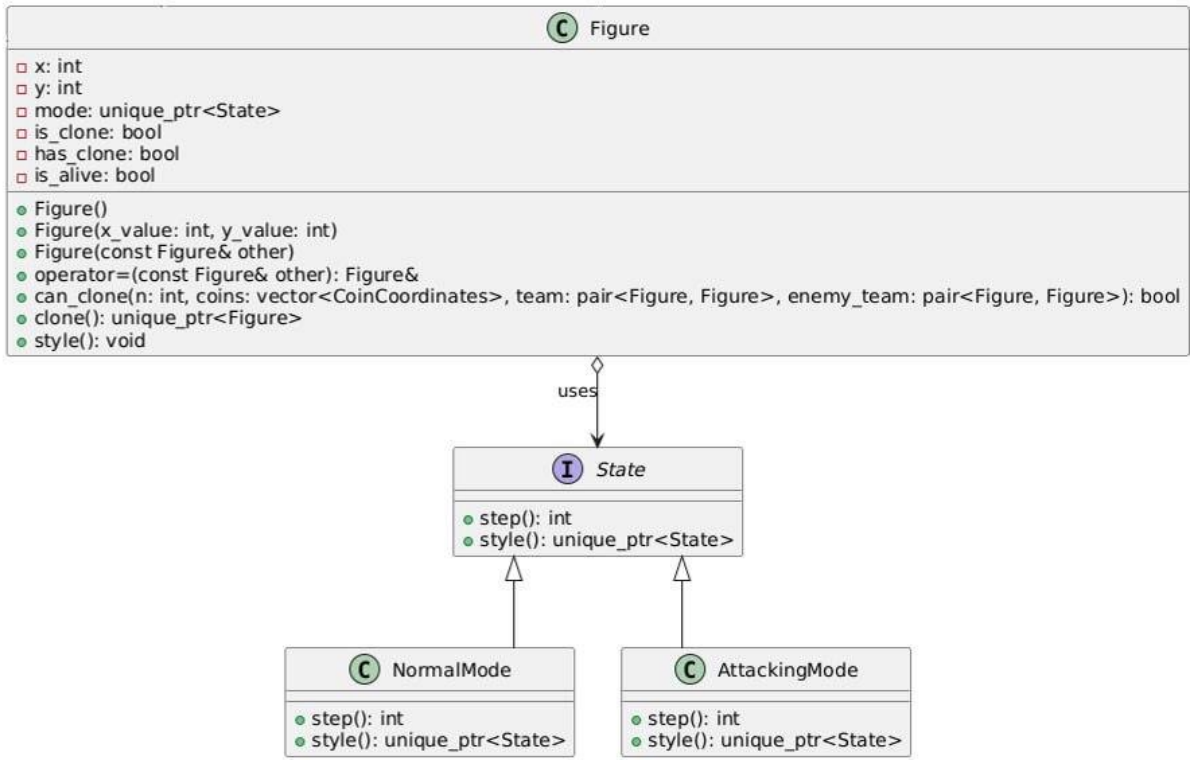
The Flyweight pattern is ideal for coins in context of the task, as it saves memory by reusing common objects (for example, coins of the same denomination) instead of creating multiple identical copies.



The shared pointers in the code is used to automatically manage the lifetime of coins and securely share them between *CoinCoordinates* objects without the risk of memory leaks or double deletion.

3. Behavioral pattern – State

Since the style of a piece can change several times per game, the state can be considered a suitable pattern. The figure dynamically changes its style during the game, that is, it changes its movement behavior depending on its state (*NormalMode* and *AttackMode*)



Full UML-Diagram: