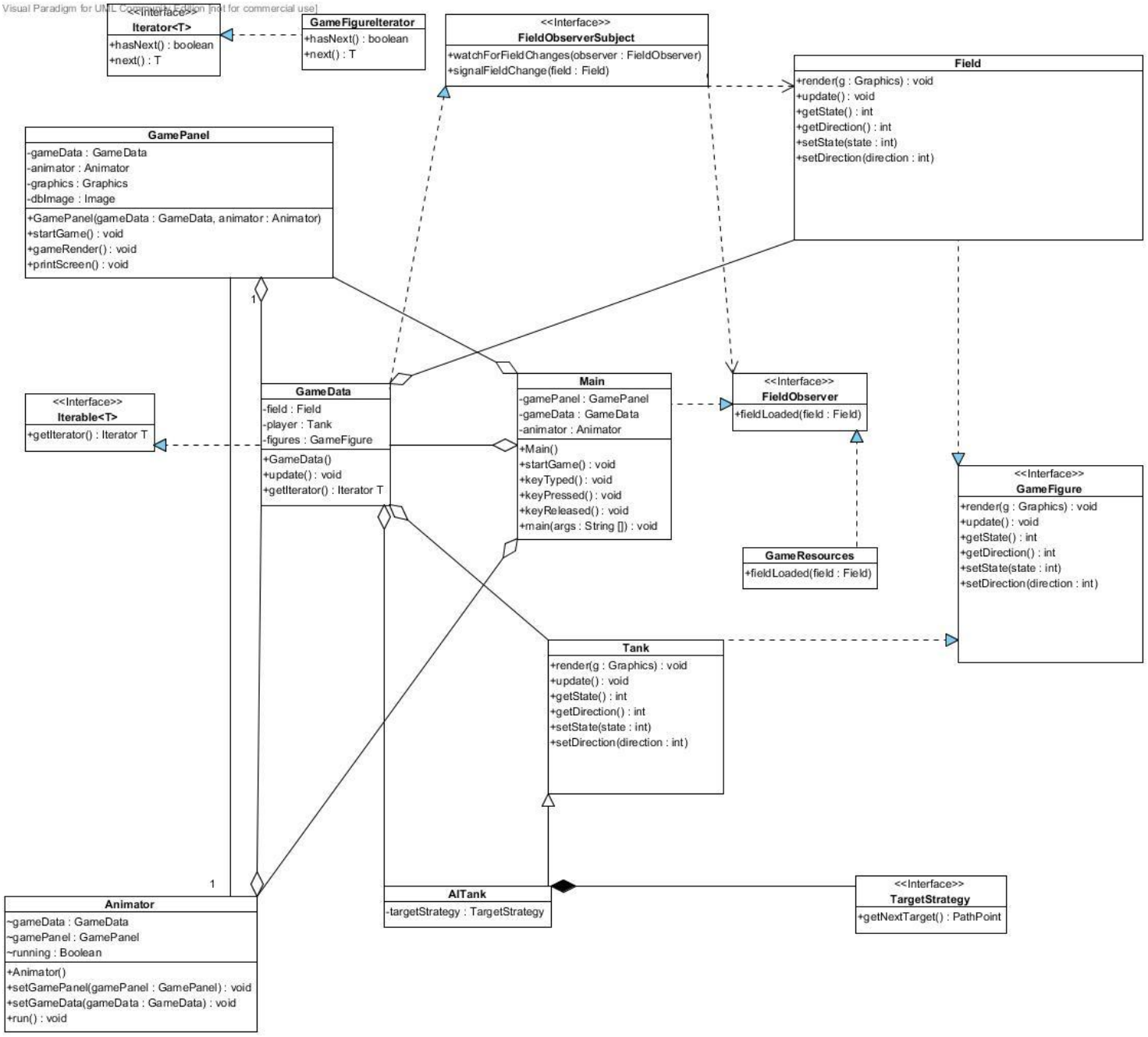
I. Game Description

Create a game, "Maze Wars" in which a player moves a tank piece around a game board with a maze layout. The object of the game is to eliminate the opposing tanks within the maze. If the player is destroyed, you lose.

II. Features

- Game levels that can be loaded from an XML file. For instance each level would be contained in a different file, as the game progresses, a new field/level is loaded to the screen. This allows the game to be very dynamic and new levels to be easily added.
- A human controlled tank whose input comes from the keyboard. Direction controlled by the arrow keys, and firing controlled by the space bar.
- AI tanks of the game which work to kill the player. They need to be able to navigate whatever field is loaded, and target the player (not each other) when they player is within range.
- Follow a basic game structure involving the following classes. GamePanel, which is in control of rendering the UI. GameData, which holds the instances of game objects to be used in the game. Finally the Animator class, which is responsible for game frame speed, looping through a pattern of calling GameData to update game objects and GamePanel to render them.
- Implement the Iterator and Observer design patterns.

III. UML Diagram



IV. Polymorphism Discussion

Maze Wars will implement the following design patterns: Iterator, and Observer. Beginning with Iterator, the Iterator<T> interface is the Iterator participant and the Iterable<T> interface is the Aggregate participant. The GameFigureIterator is the concrete Iterator, while the GameData class is the concrete iteration of the Aggregate participant.

To implement the Observer design pattern, the FieldObserver interface is the Observer participant, and the FieldObserverSubject is the Subject participant. The following classes are the concrete Observers: Main, and GameResources. The concrete implementation of the Subject is the GameData class.

The concrete implementation of the interfaces used the these design patterns is one example of how polymorphism is used. Another is the case of the AITank units that oppose the player in the game. The AITank is a child of the players Tank object. The AITank overrides the Update and GetNextCorner, also another example of polymorphism.