Buying, auctioning and trading game moves are always accompanied by return of investment calculations in making the plans. These calculations represent adaptation of the more general planning associated with the cases in the CBR database. These adaptations are necessary due to the fact that the cases do not identically correspond to the situation on the table. In addition calculating the game position value of each node of the game tree is done by heuristic functions that incorporate economic calculations of net present value, cash, and strategic layout and so on. For example railroads in monopoly are known to be strategically effective because they bring constant income even though the income can be smaller than building on other properties.

6.2 Details on the CBR Implementation

The implementation of the CBR is by using the JColibri2 platform. JColibri2 is an object-oriented framework in Java for building CBR systems that is an evolution of previous work on knowledge intensive CBR [14].

For this implementation we need to look into three particular classes of the JColibri2 platform. The StandardCBRApplication, Connector, CBRQuery. For a JColibri2 implementation the StandardCBRApplication interface needs to be implemented.

The CBR cycle executed accepts an instance of CBRQuery. This class represents a CBR query to the CBR database. The description component (instance of CaseComponent) represents the description of the case that will be looked up in the database. All cases and case solutions are implementing the CaseComponent interface.

The JColibri2 platform connects to the CBR database via a Connector class. Each connector implements all the necessary methods for accessing the database, retrieval of cases, storing and deletion of cases. This implementation uses a custom XML structure for holding the CBR cases. Since the game will not update the CBR database only read it, a XML solution satisfies the needs. The XML file to a certain extent is similar to the XML representation of the board. We are interested in finding one CBRCase that is the most similar case to the situation in the game at the time of the search. This procedure is done in the cycle method of the CBRApplication. The JColibri2 CBR comparison is done by Nearest Neighbor (NN) search method.

JColibri2 offers implementations for NN search algorithms of simple attributes. These implementations are called local similarities. For complex attributes like in our case global customized similarity mechanisms need to be implemented.

The MonopolyDescription class [Figure 13] is basically a serialization of the GameState. It holds all the information about the state of the board, the players, their amount of cash etc.

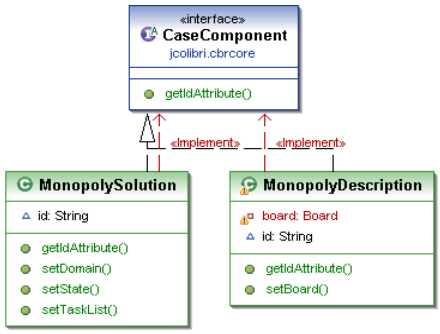

Figure 13: Class diagram of the Monopoly Case component models

On the other hand the MonopolySolution class holds the three particular attributes that are needed for the planning, the planning Domain, State and TaskList.

The game is implemented by using the Model-View-Controller software development pattern. The controller is responsible for implementing the game rules and handling all of the events in the game like roll of dice, input commands for trading, auctioning and etc from the players. The View layer is responsible for displaying the board and all of the input widgets on to the game screen, and the models are data structures representing the game state [Figure 14].

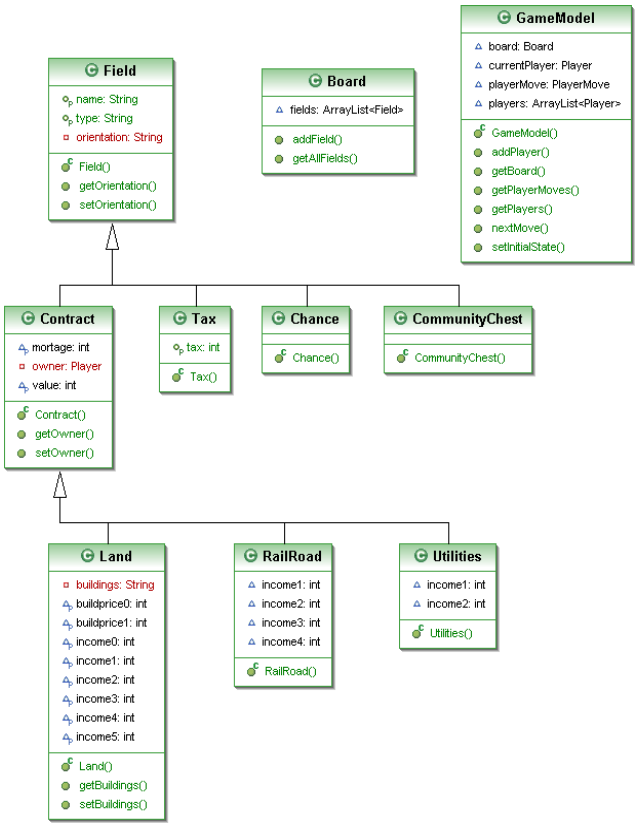

Figure 14: Class diagram of the Monopoly models

6.2.1 Complex Similarity representation in CBR

The similarity measurement part of the Nearest Neighbor algorithm JColibri2 is implemented by implementing the LocalSimiralrityFunction and the GlobalSimiralityFunction interface. A local similarity function is applied to simple attributes by the NN algorithm, and a global similarity function is applied to compound attributes. In the case of our implementation the attributes of the MonopolyDescription are compound attributes describing the state of the board, number of players, amount of cash for every player and etc. Since MonopolyDescription is a custom CaseComponent a global similarity function needs to be implemented to accurately find the distance between different CBR cases.

The similarity mechanism is inseparable core element of the CBR system. This mechanism represents how the CBR decides which strategy is best suited for the particular situation by