Initial Design Report

For

Gambler Agent

Group Name	**ErikSoft**
Taylan Işıkdemir	1560267
Alper Güngör	1560234
Volkan Çetin	1560663
İlkcan Keleş	1560382

Content

1. Introduction
    - 1.1. Problem Definition
    - 1.2. Purpose
    - 1.3. Scope
    - 1.4. Overview
    - 1.5. Definitions, Acronyms and Abbreviations
    - 1.6. References

2. System Overview

3. Design Constraints
    - 3.1. Design Assumptions, Dependencies and Constraints
    - 3.2. Design Goals and Guidelines

4. Data Design
    - 4.1. Data Description
    - 4.2. Data Dictionary

5. System Architecture
    - 5.1. Architectural Design
    - 5.2. Description of Components
        - 5.2.1. Game Playing Component
            - 5.2.1.1. Processing Narrative For Game Playing Comp.
            - 5.2.1.2. Game Playing Component Interface Description
            - 5.2.1.3. Game Playing Component Processing Detail
            - 5.2.1.4. Dynamic Behaviour of Game Playing Comp.
        - 5.2.2. Game Logging Component
            - 5.2.2.1. Processing Narrative For Game Logging Comp.
            - 5.2.2.2. Game Logging Component Interface Description
            - 5.2.2.3. Game Logging Component Processing Detail
            - 5.2.2.4. Dynamic Behaviour of Game Logging Comp.

5.2.3. Log Parser Component

5.2.3.1. Processing Narrative For Log Parser Comp.

5.2.3.2. Log Parser Component Interface Description

5.2.3.3. Log Parser Component Processing Detail

5.2.3.4. Dynamic Behaviour of Log Parser Comp.

5.2.4. Learning Component

5.2.4.1. Processing Narrative For Learning Comp.

5.2.4.2. Learning Component Interface Description

5.2.4.3. Learning Component Processing Detail

5.2.4.4. Dynamic Behaviour of Learning Comp.

5.3. Design Rationale

6. User Interface Design

6.1. Overview of User Interface

6.2. Screen Images

6.3. Screen Objects and Actions

7. Libraries and Tools

8. Time Planning (Gannt Chart)

8.1. Term 1-2 Gannt Chart

9. Conclusion

1. Introduction

1.1. Problem Definition

The problem is the lack of AI applications that can model human players for turn based games such as ‘King’. By mentioning ‘model’ , we intend to say that a computer agent learns to play like someone who the agent takes him as model. This problem first arised from the desire of humans to make computers impersonate humans in some way and to compete with these computers.

1.2. Purpose

There are several purposes of the Gambler Agent Project:

The first one is to prove that an AI agent can learn how to play the ‘king’ game by observing the game of a player who was the target of the agent to impersonate. This is like an experiment because it may be impossible to model someone for this game.In case of failure, the reasons will be explained. Otherwise, If the experiment, which has not been tried before, becomes successful, the results will be sent to some conferences.

The second purpose is to overcome the lack of online games with an intelligent learning agent. This is a desired purpose of a game by most of the players because people like to see computers behaving like themselves. There are some games with learning agents like chess, backgammon etc. However, there is no learning agent for the ‘king’ game which is too complex and extensive to be succesfull.

1.3. Scope

The intelligent system for turn based games are very common nowadays. Most turn based game developer companies design such a system. On the other hand there are not many instances of learning system for turn based games which are partially observable. Observable means all the game environment i.e moves of the opponents, evaluation of them is open to each player. Tile and card games are partially observable. Players only know the cards or tiles which are in their hands and thrown in previous turns.

King is an example of partionally observable games. King is played by four people and no teaming is allowed. At the end of the game players with positive points are considered as winners. There are six type of negative hands which are explained below:

No tricks – The aim is not to win tricks. The dealer plays any card and all the other players must follow that suit unless they do not hold any card of that suit. The winner of the trick is the highest card of the suit played at the start of the trick or the highest trump, if any was declared for that hand. The winner restarts play with any card and so on until all the cards have been played. Then, tricks of each player is counted. Each trick is worth 50 negative points and the total for the hand is 650 negative points.

No Hearts – The aim is not to win tricks with Hearts. A player must not start a trick with Hearts unless he holds no other suit. If a player cannot follow suit he can then play any card, including Hearts. Each Hearts card is worth 30 points and the total for the hand is 390 negative points.

No Queens – The aim is not to win tricks with Queens. Each Queen is worth 100 points and the total for the hand is 400 negative points.

No Kings or Jacks - The aim is not to win tricks with Kings or Jacks. Each King and Jack is worth 60 points and the total for the hand is 480 negative points.

No King of Hearts – The aim is not to get the King of Hearts. A player must not start a trick with Hearts unless he holds no other suit. Important: The King of Hearts must be played at the first legal opportunity, meaning when the holder cannot follow suit or at the first time Hearts is used to open a trick. The King of Hearts is worth 320 negative points.

No last 2 tricks - The aim is not to win the last 2 tricks. Each of those tricks is worth 180 negative points and the total for the hand is 360 negative points.

There is also one positive hand which is named trump. Trump cards win against any other suit but can only be played if the suit cannot be followed or if the trick started with the trump suit. Between two or more trump cards the highest one wins. Each trick taken by a player is worth 50 points and the total for the hand is 650 positive points.

1.4. Overview

The rest of this document contains an detailed description of the Intelligent and Learning System for Turn Based Games and specific constraints. There will be more specific details and information about the project content, system overview, design constraints, data design, system architecture and planning.

1.5. Definitions, Acronyms and Abbreviations

DB:	Database
AI:	Artificial Intelligence
GUI:	Graphical User Interface
ILSTBG:	Intelligent and Learning System for Turn Based Games
API:	Application programming interface
Http:	Hypertext Transfer Protocol
UML:	Unified Modelling Language
IEEE:	Institute of Electrical and Electronics Engineering
AAAI	Association for the Advancement of Artificial Intelligence

1.6. References

http://en.wikipedia.org/wiki/Artificial_neural_network

http://www.cs.cmu.edu/~tom/mlbook.html

http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html

http://aima.cs.berkeley.edu/

http://en.wikipedia.org/wiki/List_of_software_development_philosophies

http://www.sciencenews.org/sn_arc98/7_18_98/bob1.htm

http://en.wikipedia.org/wiki/Support_vector_machine

2. System Overview

The system will have three main stages. First one is for collecting training data. There will be some different types of agents and they will play against each other in diverse combinations to make training data richer. The second one is the most important one. It is the learning process. The training data obtained from first stage will be the source of learning system while defining state-action pairs. That means an agent is going to use the pairs determined in this part. The third one is the complete online 'king' game with all functionalities of usual online games and additionally some new functionalities for agent usage.

The training data collection part of the system is designed for creating the basis of the learning system. To create a well-behaving learning system, one needs too much training data which should be gathered from varying sources.

The learning system will be decomposed into parts too. Some techniques will be tried and when any of them becomes successfull, that one will be used for the project. To understand whether the learning is succesfull, test data will be collected from latest version of learning agent. Afterwards this test data will be compared to the training data from first stage. If they have same characteristics (percentage of same actions in same states), then it will be considered as succesfull. The candidate algorithms for the time being are reinforcement learning, neural network, support vector machines.

The complete 'king' game will be available online. It will look like other card games and easy to use. There can be lots of agents in any table in the game. The agents are going to communicate with their desicion function through a web service. The final state definition, one of the succesfull ones that had been tried in stage two, will be sent as input and the output is an action which agent is going to do.

The goal of the system is to show that learning is possible for the 'king' game.

3. Design Constraints

3.1. Design Assumptions, Dependencies and Constraints

Assumptions:

Learning phase for 'King' game may be unsuccesfull due to the properties of this game. Since there are seven different games in the 'King' game and the game is not fully observable, learning process in this limited time and processor number is a difficult task.

If this situation occurs,we will switch to the other game called 'Okey' which may be easier to implement a learning system for.

Since Gambler Agent application is a web-based Project, there will not be any system dependencies. It is enough to have the necessary plugins for the browser. We will make it available for all browsers.

Constraints:

Reports during development phase of the project will be prepared according to IEEEStd830. UML design and flow charts are going to take part in that phase. JavaDoc style comments will be used for documentation of the source code. In the implementation phase SVN is going to be used for accordance in development team. Any changes of DB schema during implementation phase will be recorded and initial design will be updated accordingly.

3.2. Design Goals and Guidelines

The principles which are going to be considered in software development are:

- You ain't gonna need it (YAGNI) : is the principle in extreme programming that programmers should not add functionality until it is necessary. There is a quote for this principle : "Always implement things when you actually need them, never when you just foresee that you need them."
- Abstraction principle : is a basic dictum that aims to reduce duplication of information in a program (usually with emphasis on code duplication) whenever practical by making use of abstractions provided by the programming language.
- Don't Make Me Think : this principle's premise is that a good program or web site should let users accomplish their intended tasks as easily and directly as possible.
- Team Software Process (TSP) : provides a defined operational process framework that is designed to help teams of managers and engineers organize and produce large-scale software projects of sizes beyond several thousand lines of code (KLOC). The TSP is intended to improve the levels of quality and productivity of a team's software development project, in order to help them better meet the cost and schedule commitments of developing a software system.
- Reusability : is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required.
- Maintainability: is the ease with which a product can be maintained in order to:

  - correct defects
  - meet new requirements
  - make future maintenance easier, or
  - cope with a changed environment

4. Data Design

4.1. Data Description

4.1.1. Game Playing

The game playing part of project will include Lobby, Table, Game, Player ,Card data objects.

Lobby object will be the topmost layer through other data objects. Lobby will include the logged players and created tables. When a player joins a table then it will be removed from Lobby object and transferred to Table object.

Table object will have a Game object and max four Player object. The players can be either type of human or agent.

Game object will do the main job. Game object will include Card object.
This object will deliver the cards to players when game is started. This object also will keep the game chart and point chart. The information of played game types are kept in the game chart.

Player object will keep the array of delivered cards and his/her available game type chart. In this object also players thrown cards and points wil be kept.

4.1.2. Keeping game logs

The data objects for this part of the project are Trainer, TrainingTable and the following objects from the game playing part: Card, GameType, Player.

Trainer object behaves like a Main class, it is the start point and manages other objects in the environment.

TrainingTable is a special kind of Table mentioned in the game playing part. This is only for fixed game types and only agent players are playing the game, no human player. After each game this objects saves the game log to a file in appropriate directory in the file hierarchy which is described below.

Keeping game logs part of the project will supply game log files for further parts of the project, from the games which agents play against each other. At the top of file hierachy there will be seven (number of game types) directories all of which includes three more directories which corresponds to agent types namely random-like agent, rule-based agent and hybrid agent. These directories will be populated by the files from the games which are played by agents. Since the games will be played on NAR machine, this file hierarchy must be held in disc of NAR machine.

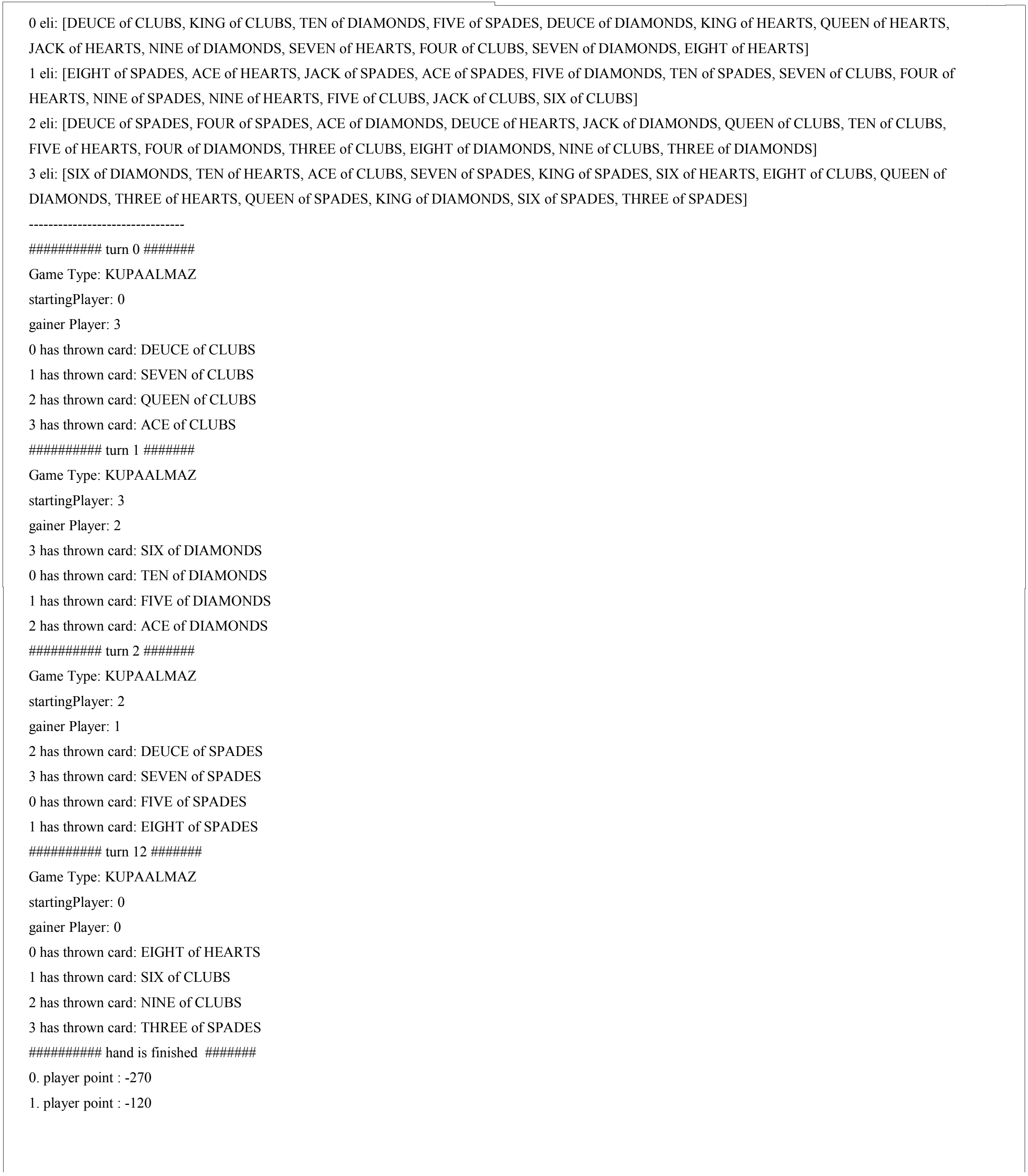
Each game log file will be a binary data file because of size constraints. The format of this file is like the following:

"Cards" in each players' hand

"Thrown cards" and "score" update if any change

Repeat the last action 13 times.

An example of a game log file is at Fig-1.



```
0 eli: [DEUCE of CLUBS, KING of CLUBS, TEN of DIAMONDS, FIVE of SPADES, DEUCE of DIAMONDS, KING of HEARTS, QUEEN of HEARTS,
JACK of HEARTS, NINE of DIAMONDS, SEVEN of HEARTS, FOUR of CLUBS, SEVEN of DIAMONDS, EIGHT of HEARTS]
1 eli: [EIGHT of SPADES, ACE of HEARTS, JACK of SPADES, ACE of SPADES, FIVE of DIAMONDS, TEN of SPADES, SEVEN of CLUBS, FOUR of
HEARTS, NINE of SPADES, NINE of HEARTS, FIVE of CLUBS, JACK of CLUBS, SIX of CLUBS]
2 eli: [DEUCE of SPADES, FOUR of SPADES, ACE of DIAMONDS, DEUCE of HEARTS, JACK of DIAMONDS, QUEEN of CLUBS, TEN of CLUBS,
FIVE of HEARTS, FOUR of DIAMONDS, THREE of CLUBS, EIGHT of DIAMONDS, NINE of CLUBS, THREE of DIAMONDS]
3 eli: [SIX of DIAMONDS, TEN of HEARTS, ACE of CLUBS, SEVEN of SPADES, KING of SPADES, SIX of HEARTS, EIGHT of CLUBS, QUEEN of
DIAMONDS, THREE of HEARTS, QUEEN of SPADES, KING of DIAMONDS, SIX of SPADES, THREE of SPADES]
--------------------------------
########## turn 0 #######
Game Type: KUPAALMAZ
startingPlayer: 0
gainer Player: 3
0 has thrown card: DEUCE of CLUBS
1 has thrown card: SEVEN of CLUBS
2 has thrown card: QUEEN of CLUBS
3 has thrown card: ACE of CLUBS
########## turn 1 #######
Game Type: KUPAALMAZ
startingPlayer: 3
gainer Player: 2
3 has thrown card: SIX of DIAMONDS
0 has thrown card: TEN of DIAMONDS
1 has thrown card: FIVE of DIAMONDS
2 has thrown card: ACE of DIAMONDS
########## turn 2 #######
Game Type: KUPAALMAZ
startingPlayer: 2
gainer Player: 1
2 has thrown card: DEUCE of SPADES
3 has thrown card: SEVEN of SPADES
0 has thrown card: FIVE of SPADES
1 has thrown card: EIGHT of SPADES
########## turn 12 #######
Game Type: KUPAALMAZ
startingPlayer: 0
gainer Player: 0
0 has thrown card: EIGHT of HEARTS
1 has thrown card: SIX of CLUBS
2 has thrown card: NINE of CLUBS
3 has thrown card: THREE of SPADES
########## hand is finished  #######
0. player point : -270
1. player point : -120
```

```
2. player point : 0
3. player point : 0
```

Figure 1. Example Log File

4.1.3. Forming state action pairs

Forming state-action pairs part of the project will consist of database which is a set of tables. The log files from the previous part will be parsed and inserted into corresponding database table. Main tables are State, GameType, Action, AgentType. There will be one relation called ActionOf connecting these tables. The database schema is like the following ER diagram.

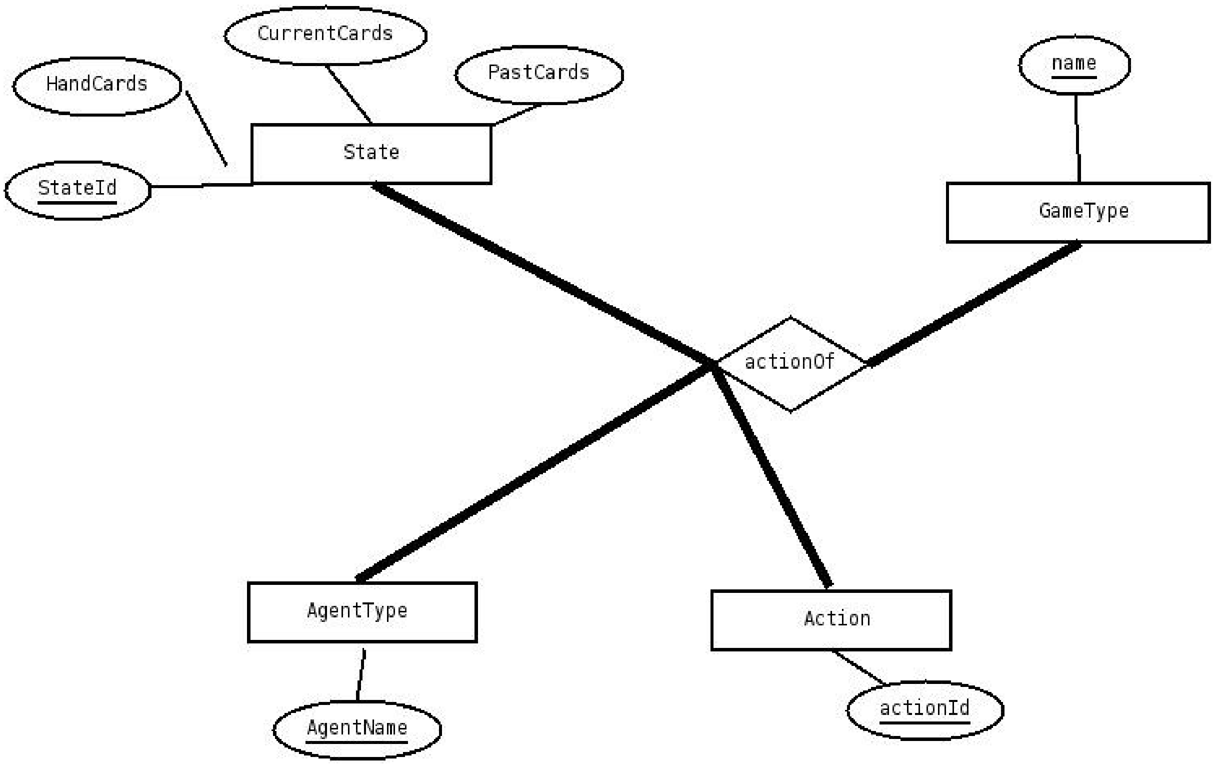


Figure 2: ER diagram of state-action DB

4.2. Data Dictionary

Action : is a database table which is the output of ActionOf relation.

ActionOf : is a relation in database which connects State, GameType, AgentType and Action tables.

Agent : is a Java class representing AI agents and extending Player class

AgentType : is a database table which keeps different type of agents.

Card : is a java class. There are 4 suits and 13 ranks for each suit.

Game : is a java class representing the whole game.

GameLogFile: is a special formatted binary file to keep logs.

GameType : is a database table which keeps seven different game types.

HumanPlayer: is a Java class representing human players and extending Player class.

Inserter: is a Java class which updates the state-action DB whenever needed.

Lobby : is a java class holds the list of active tables and players.

Player : is a java abstract class which is extended by all player classes.

Reader: is a Java class which parses game log files and sends necessary information to Inserter object.

State : is a database table which keeps hand, currentCars, pastCards and a unique id attributes.

Table : is a Java class in which a game object and four player objects are held.

Trainer: is a Java class, manages game logging part by creating TrainingTable objects.

TrainingTable:is a Java class which is a special kind of Table object.

5. System Architecture

5.1. Architectural Design

Our system will have a modular structure in which functionality of the system is divided into subsystems. There will be four components to accomplish the system's goals shown in Figure 3.

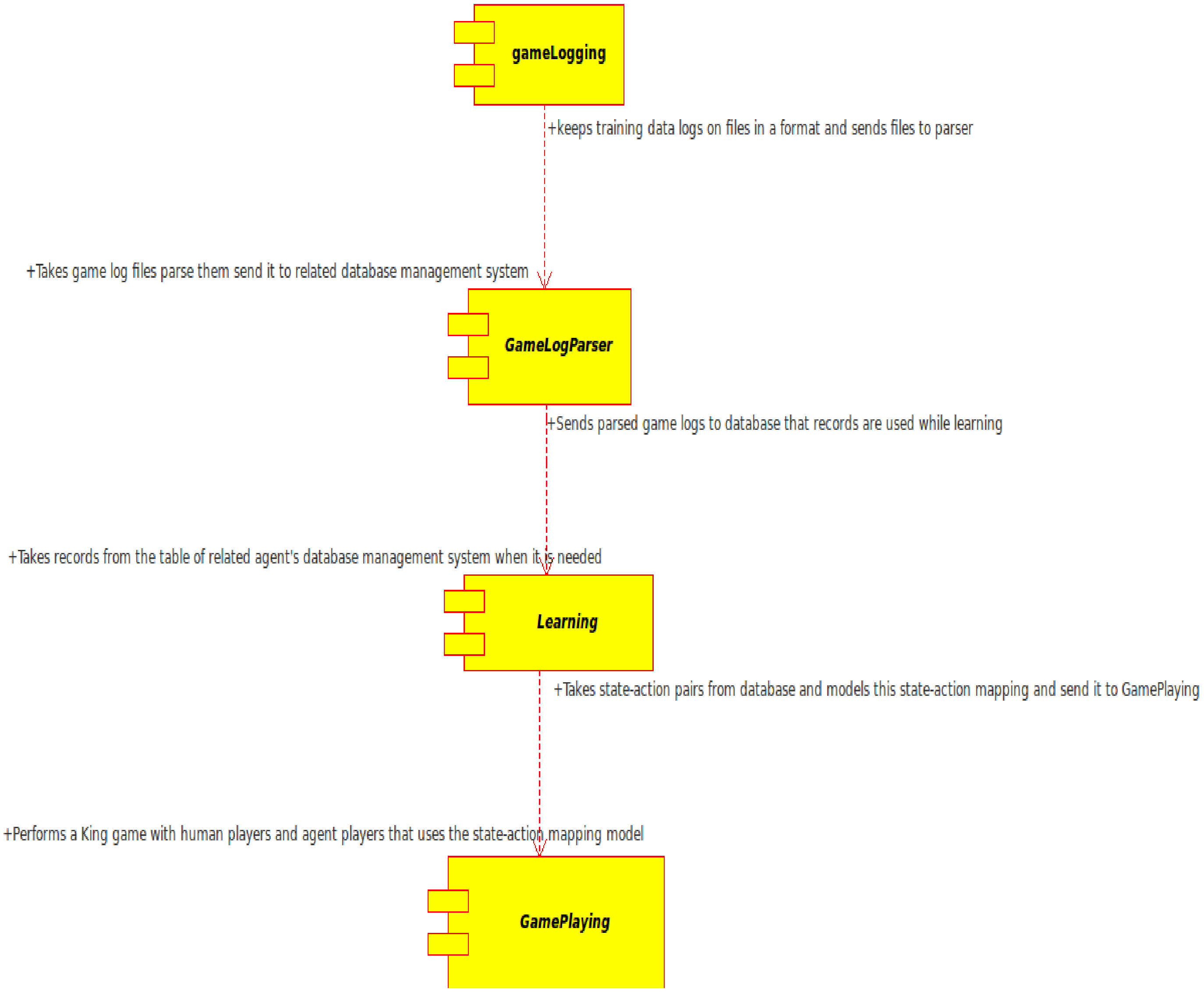

gameLogging
+keeps training data logs on files in a format and sends files to parser
+Takes game log files parse them send it to related database management system
GameLogParser
+Sends parsed game logs to database that records are used while learning
+Takes records from the table of related agent's database management system when it is needed
Learning
+Takes state-action pairs from database and models this state-action mapping and send it to GamePlaying
+Performs a King game with human players and agent players that uses the state-action mapping model
GamePlaying

Figure 3: Component Diagram of The System

First component of the system is the main system in which the game 'King' is played. This subsystem will consist of many classes namely; Lobby, Table ,Game, Player, Card, GameType, AvailableGameTypes, PlayerFactory. Player class is an abstract class which is extended by HumanPlayer and agent classes. Second component of the system is the one which collect game logs from the games played amongst our rule-based and hybrid agents and write them to the files using the first subsystem. This subsystem gives us the opportunity to keep logs in one determined file format. For this reason, operations on these files will be simpler. Third component of the system is responsible for parsing the game logs and transferring the data represented in the files to the database according to some constraints. The output of this system will be directly used in our agents' learning phase and to keep training data in the database provides us to use the database queries. After training data is created, the final subsystem will be ready to run to complete the learning phase. In this component, a suitable learning algorithm will be implemented and some data structures and some mathmatical models will be used to make this implementation easier.

5.2. Description of Components

5.2.1. Game Playing Component

This is the topmost layer of the system. All components will be combined into this component to make users play 'King' however they want i.e against human player or against agents.

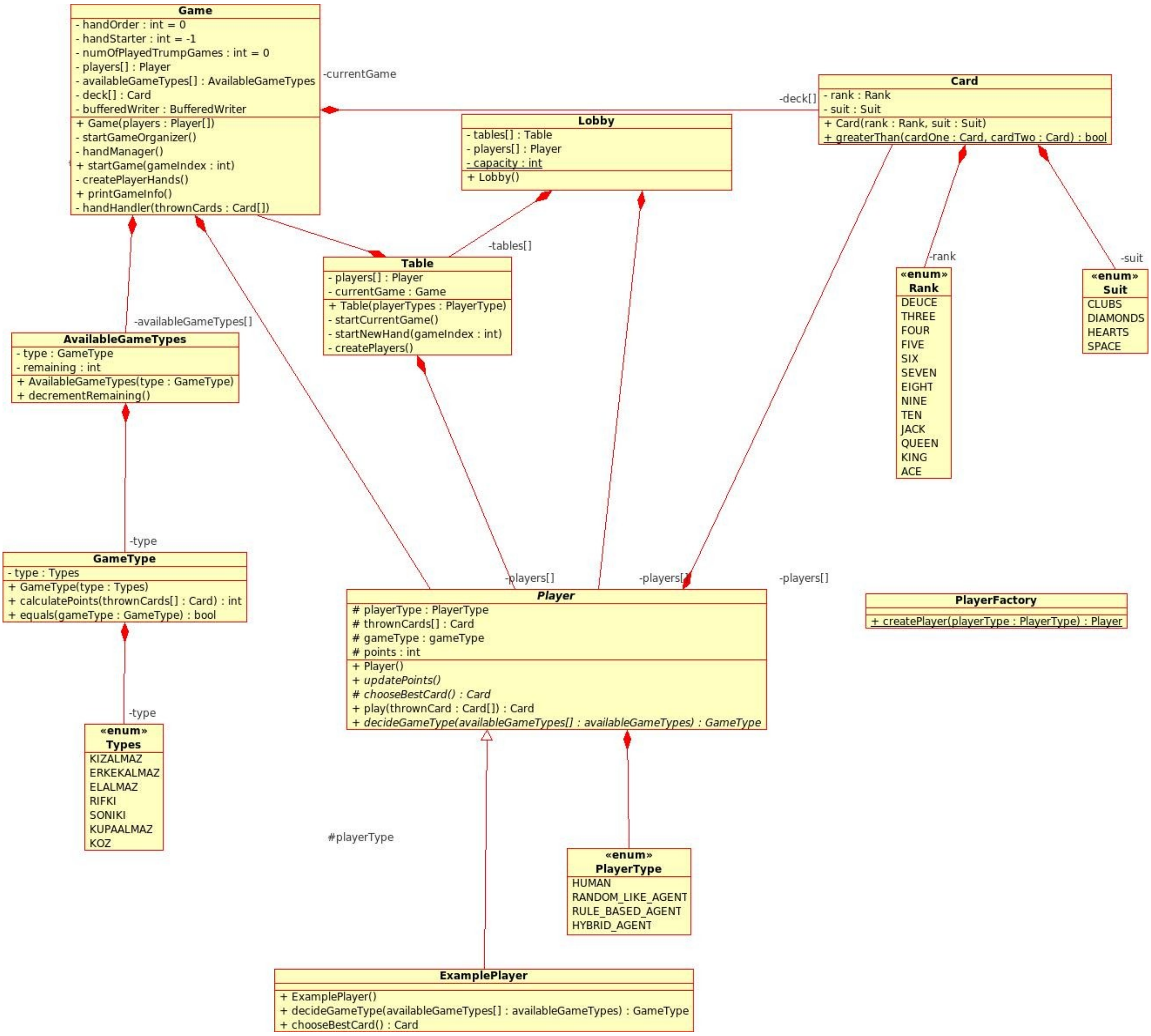


Figure 4: Class Diagram of Game Playing Component

5.2.1.1. Processing Narrative For Game Playing Component

Game playing component stands for the complete online ‘King’ game system. The component is responsible for user actions. These actions are logging in, logging out, creating a table, chatting with other players, checking game scores, joining to a table, editing account information. These actions will be available through website, and user information will be kept in a secure database. All kinds of exceptions such as the unexpected server crashes, player’s exit during a game, problems in database which contains agents’ look up tables will be handled by this component. This component is also responsible for the game rules. For instance when playing No Hearts, the players can not play hearts before a heart is thrown onto another suit.

5.2.1.2. Game Playing Component Interface Description

The components’ input interfaces are the events that users trigger. Such events are mouse clicks, keyboard entries etc. The output interface is the graphical user interface which is accessible via web browsers. The figure 13-14 is an example of the game playing component’s gui. The details are explained in Chapter 6.

5.2.1.3. Game Playing Component Processing Detail

This component itself does not have an important algorithmic implementation. The important points of the implementation is handling of the events that user trigger and the game rules. The synchronization amongst players is also another key issue for this component.

5.2.1.4. Dynamic Behavior of Game Playing Component

The data objects which are described in Chapter 4.1 in game playing section explained the interactions between the classes of the component. The sequence diagram, use case diagrams one of which for human player and the other one for agent player, and the activity diagram are provided below in Figure 5-6-7 in order to visualize the interactions.

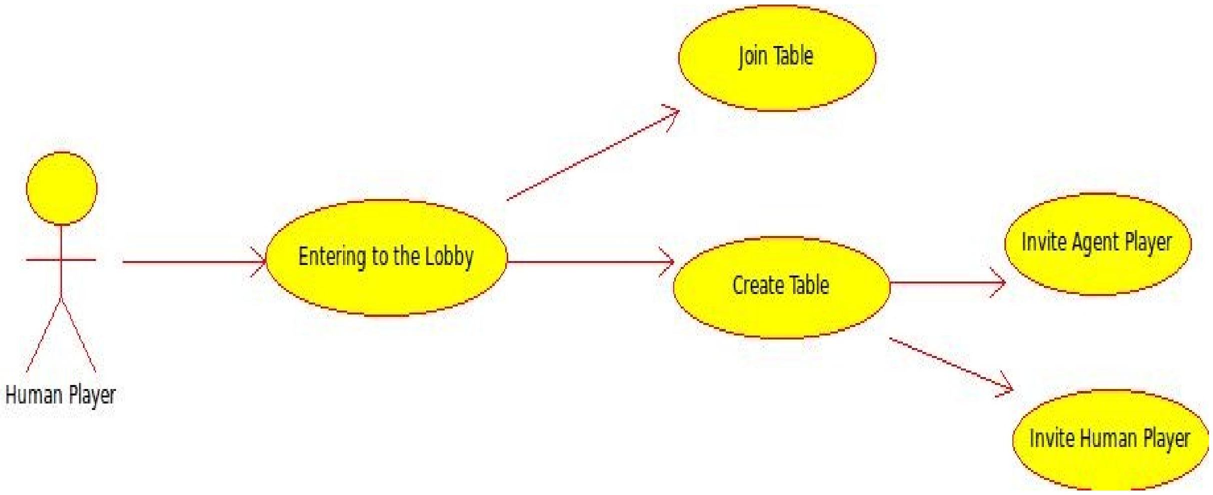


Figure 5 : Use case for Human Player

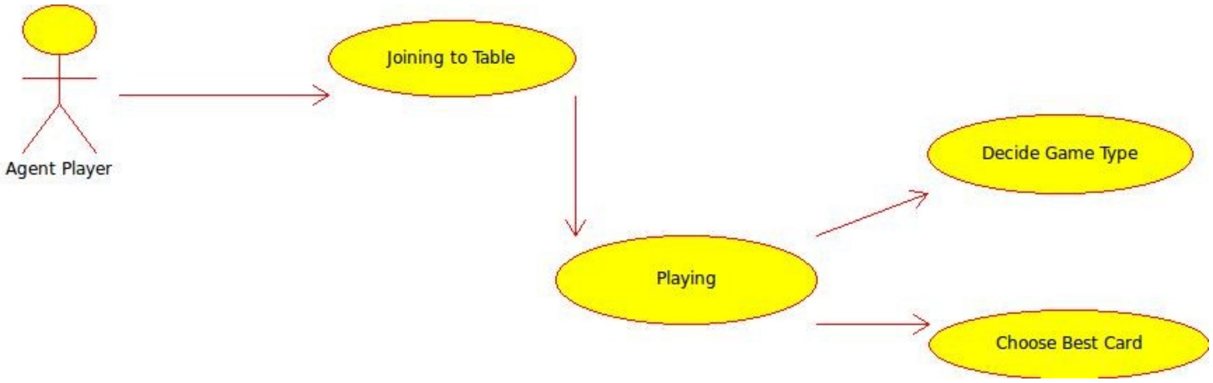


Figure 6 : Use Case for Agent Player

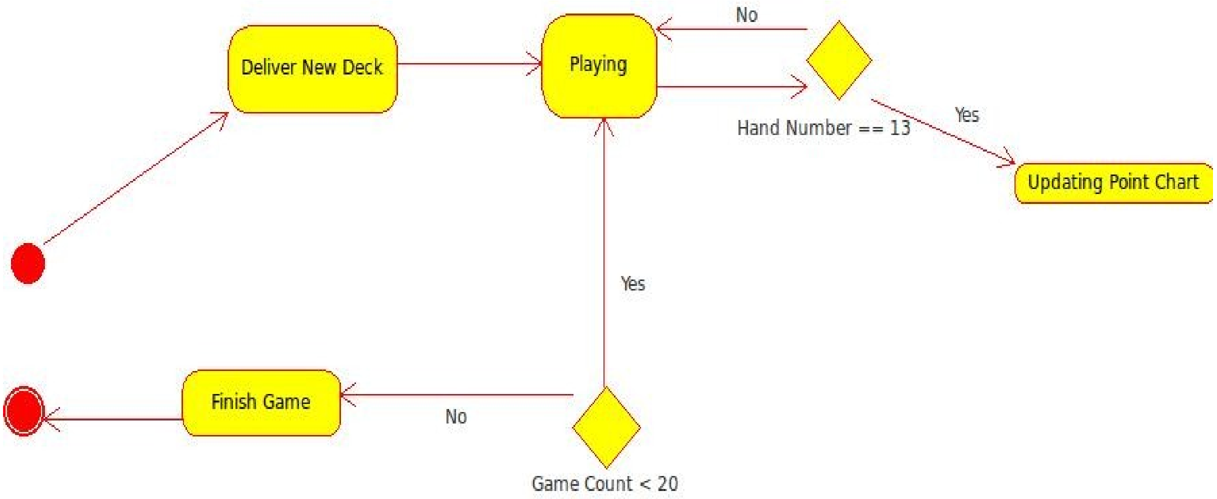


Figure 7 : Activity Diagram For Game Playing

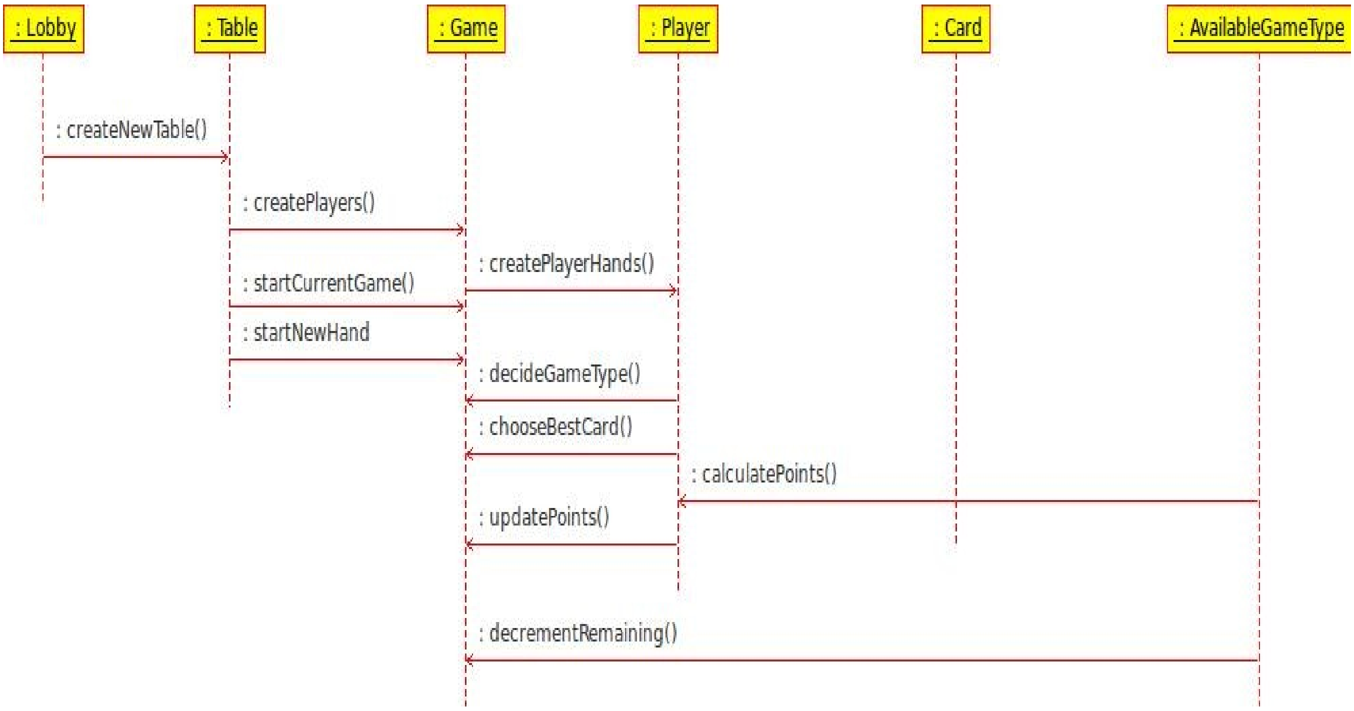


Figure 8: Sequence Diagram of Game Playing Component

5.2.2. Game Logging Component

This component is responsible for keeping the logs of the games. Game logging component will be used for two purposes: to collect data from agent games for learning phase and to collect data from human players' game for updating learning data.

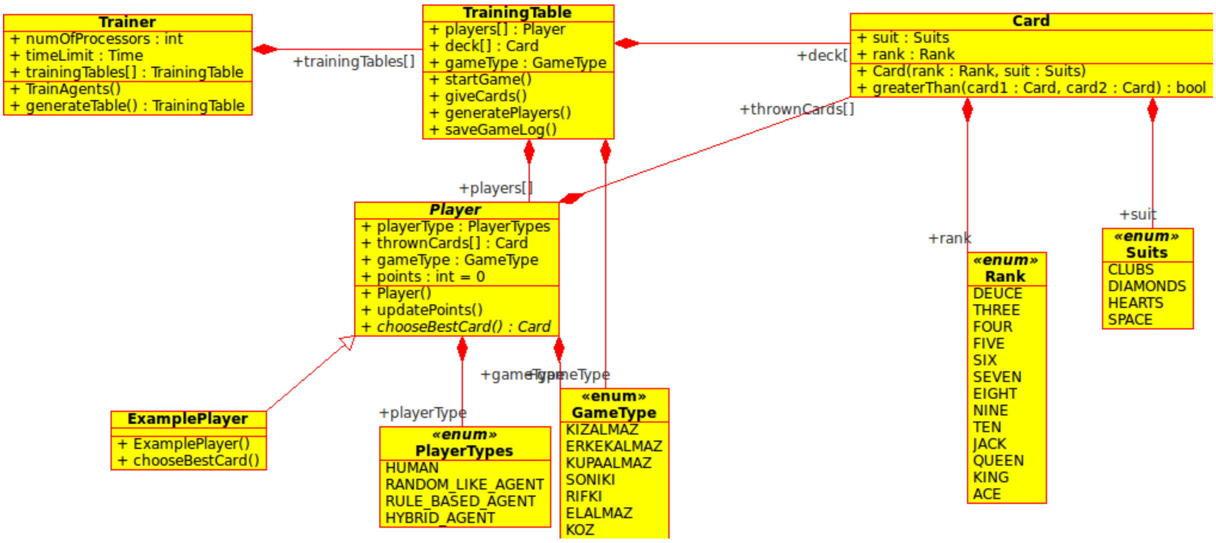


Figure 9: Class Diagram of Game Logging Component

5.2.2.1. Processing Narrative For Game Logging Component

Game logging component stands for the part of the project which will make agents play amongst themselves and writing the game data to the files in the format which is mentioned in chapter 4. This will give us the opportunity to transfer the data from these files to the database afterwards.

5.2.2.2. Game Logging Component Interface Description

There are not any graphical user interfaces in this component. However; the files which are created by this component will be used by the third component.

5.2.2.3. Game Logging Component Processing Detail

This component does not have any algorithm implementation. The component only includes the game rules to make agents play correctly and a file format which uses less space in order to make the component more efficient.

5.2.2.4. Dynamic Behavior of Game Logging Component

The data objects which are described in keeping game logs section explained the data operations inside the component. The sequence diagram is provided below in Figure X in order to visualize the interactions.

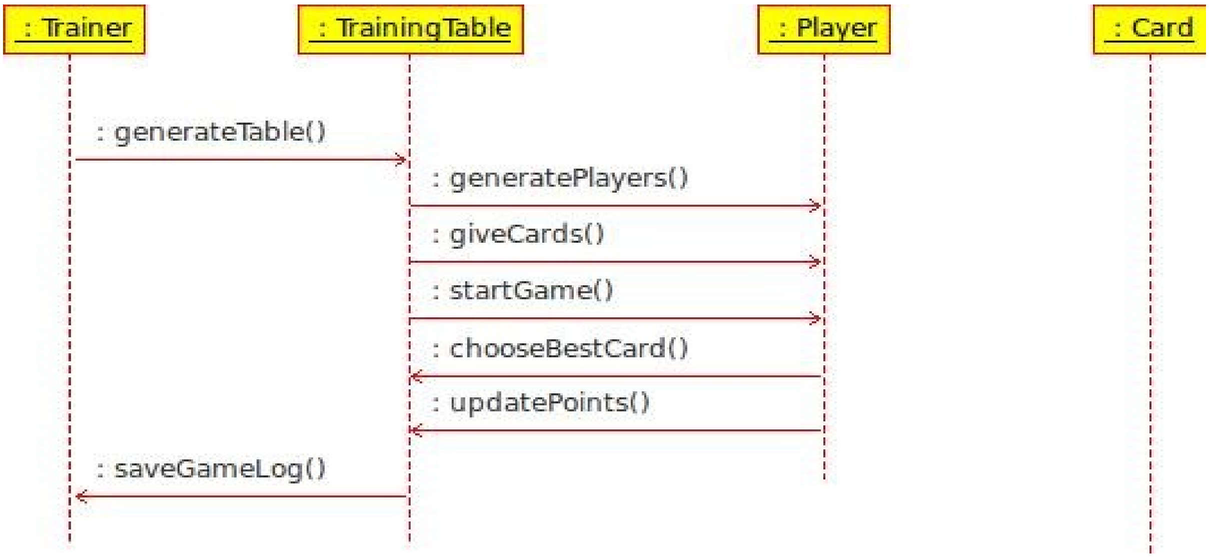


Figure 10: Sequence Diagram of Game Logging Component

5.2.3. Log Parser Component

This component's job is totally related with the Game Logging component. Game Logging component is going to run for approximately a month on a multiprocessor environment. Therefore enough data will be obtained for Learning component.

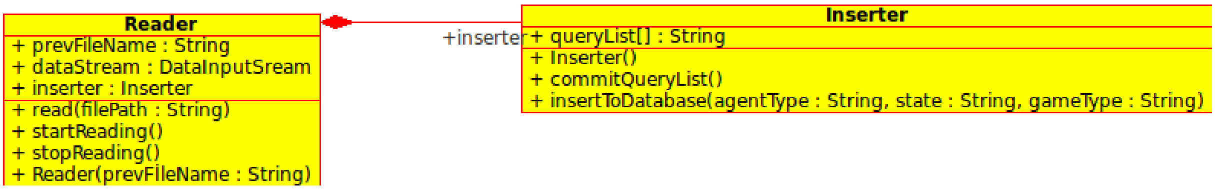

Reader
+ prevFileName : String
+ dataStream : DataInputSream
+ inserter : Inserter
+ read(filePath : String)
+ startReading()
+ stopReading()
+ Reader(prevFileName : String)
+inserte
Inserter
+ queryList[] : String
+ Inserter()
+ commitQueryList()
+ insertToDatabase(agentType : String, state : String, gameType : String)

Figure 11: Class Diagram of Log Parser Component

5.2.3.1. Processing Narrative For Log Parser Component

The main job of Log Parser Component is to parse the game log files and insert them to a database accordingly. The ER diagram of the database is given in Figure 123. Each log file will has 13x4 rows to be inserted to the db. Because every card played by each of four player is another state-action pair. And each player has 13 cards in the begining. So it leads to 52 different state for each log file. In future different parsers and different db schemas can be considered in case of any difficulties for implementing learning agents.

5.2.3.2. Log Parser Component Interface Description

There is not any GUI for this component. It will be a background process. Input is the log files from Game Logging component and output is the database tables.

5.2.3.3. Log Parser Component Processing Detail

Since the log files will be pile file, there is no need for indexing and search in the files. The process will be straightforward reading from files. And after each move of each player in the file, one new row will be inserted to the actionOf relation in the db. If the corresponding row is already in the table some attributes will be updated such as points, occurence.

5.2.3.4. Dynamic Behavior of Log Parser Component

The data objects which are described in chapter 4 explained the interactions between the classes of the component. The sequence diagram is provided below in Figure 9 in order to visualize the interactions.

5.2.4. Learning Component

This component is the final step for creating intelligent agents. Our previous agents were also intelligent but their intelligence were restricted with their implementation. On the other hand the learning agents will allways continue to learn and play accordingly.

5.2.4.1. Processing Narrative For Learning Component

There is no distinct ways of machine learning. Especially for this kind of learning project almost nothing is strictly definite. Some algorithms from supervised and unsupervised learning will be tried. Supervised learning is the task of inferring a function from supervised training data. A supervised learning algorithm analyzes the training data and produces an inferred function. Unsupervised learning is a class of problems in which one seeks to determine how the data are organized. Many methods are based on data mining methods used to preprocess the data. The state-action database formed by Log Parser component is going to be used as training data samples and several algorithms are going to be used to achieve a level of learning.

5.2.4.2. Learning Component Interface Description

The state-action database is going to be used as input of this component. If it comes out that there is not sufficient information for learning. Log parser component will be revised accordingly and designed again.

Since this is the final component of the project, the output, which is an intelligent and learning agent, will be directly used by Game Playing component.

5.2.4.3. Learning Component Processing Detail

As stated in 5.2.4.1 , supervised and unsupervised machine learning algorithms will be tried to correctly model the agents namely random-like, rule-based and hybrid agents which are already implemented. Artificial Neural Networks (ANN) or Support Vector Machines (SVN) are going to be used to implement learning part. ANN is appropriate to use with both supervised and unsupervised learning. SVN is for supervised learning and also for statistical classification.

Artificial neural networks (ANNs) are essentially simple mathematical models defining a function f: X → Y or a distribution over X or both X and Y, but sometimes models are also intimately associated with a particular learning algorithm or learning rule. An illustriation of ANN is given in Figure XX.

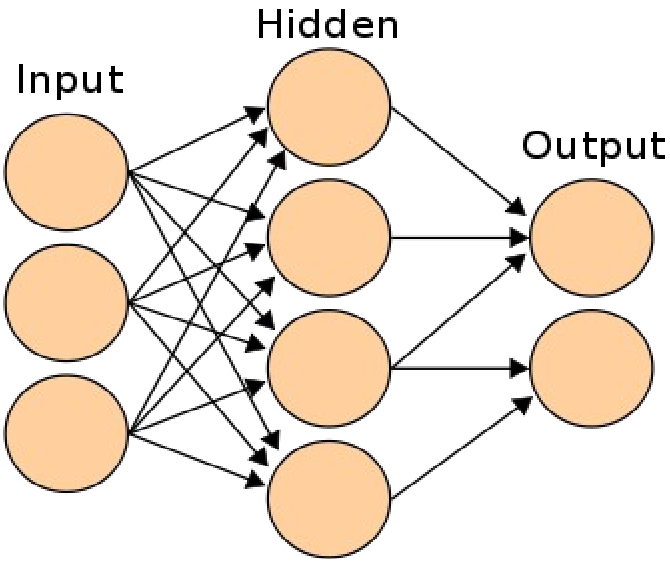


Figure 12: Artificial Neural Network

5.2.4.4. Dynamic Behavior of Learning Component

The data objects will be described in detailed design report which shows internal interactions between classes of this component.

5.3. Design Rationale

This decomposition is choosed because of some considerations. First one is to have a rich and easy to manipulate training data set. This has to be available before starting the learning part. So design of the game logging and log parser components are well defined and conceived. Second one is to have a complete game playing component before starting anything. So things that are definitely necessary for game, became obvious. And the design of other components are effected by this knowledge possitively.

Some different decompositions are also possible. However this one seemed the most reasonable among other decompositions because other ones were not suitable for some design principles which are obeyed by the project designers.

6. User Interface Design

6.1. Overview of User Interface

Game window : It will be opened as 480X640 and it will have option for users to make window full screen.

Chat Box: Chat box will be available for the players who are on the related table.

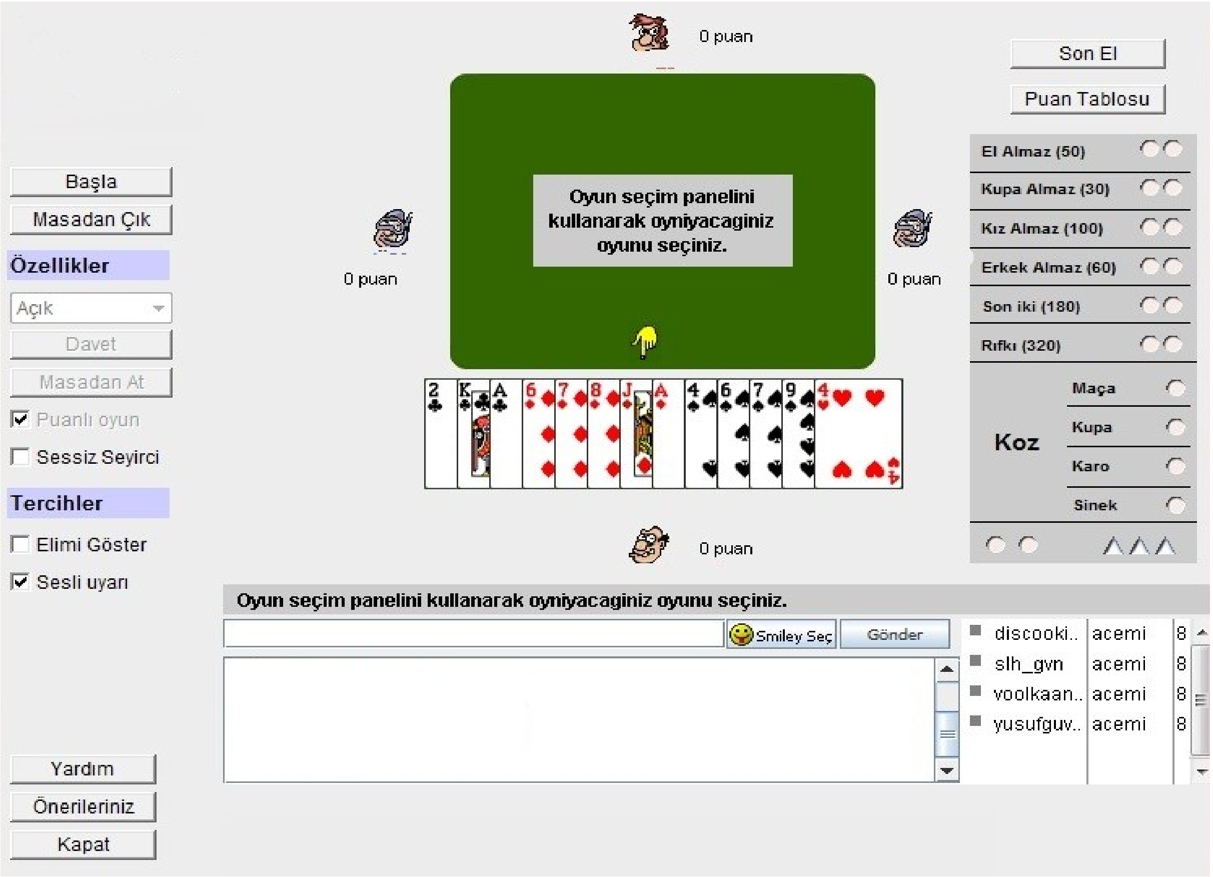
Buttons:There will be a few buttons that have different functionalities like adding an agent, opening information table,starting game.

Timer:The timer keeps the track of the time after game started.

Information Table:The information table show information about the players their scores and game status.

6.2. Screen Images

General Interface:



Oyun seçim panelini kullanarak oyniyacaginiz oyunu seçiniz.

☑ Puanlı oyun

□ Sessiz Seyirci

□ Elimi Göster

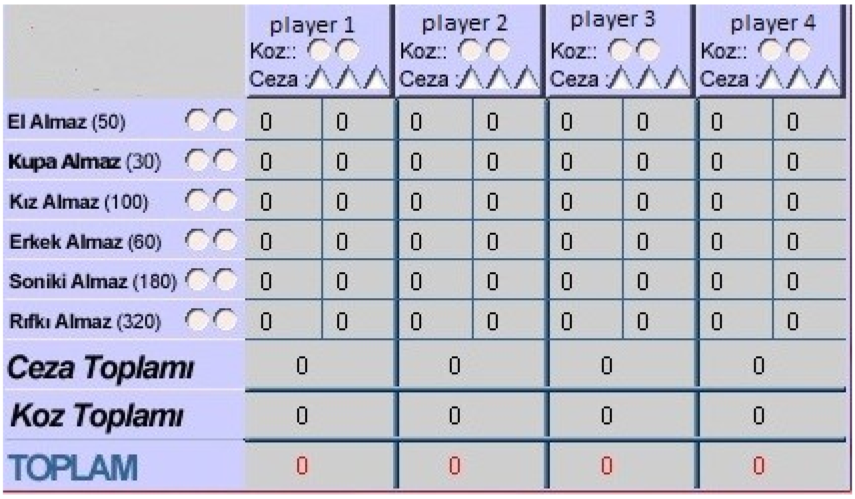
☑ Sesli uyarı

Oyun seçim panelini kullanarak oyniyacaginiz oyunu seçiniz.

discooki..	acemi	8
slh_gvn	acemi	8
voolkaan..	acemi	8
yusufguv..	acemi	8

Figure 13: Game Window

Score Table:



	player 1 Koz:: ○○ Ceza :⋀⋀⋀		player 2 Koz:: ○○ Ceza :⋀⋀⋀		player 3 Koz:: ○○ Ceza :⋀⋀⋀		player 4 Koz:: ○○ Ceza :⋀⋀⋀	
El Almaz (50) ○○	0	0	0	0	0	0	0	0
Kupa Almaz (30) ○○	0	0	0	0	0	0	0	0
Kız Almaz (100) ○○	0	0	0	0	0	0	0	0
Erkek Almaz (60) ○○	0	0	0	0	0	0	0	0
Soniki Almaz (180) ○○	0	0	0	0	0	0	0	0
Rıfkı Almaz (320) ○○	0	0	0	0	0	0	0	0
Ceza Toplamı	0		0		0		0	
Koz Toplamı	0		0		0		0	
TOPLAM	0		0		0		0	

Figure 14: Information Table

6.3. Screen Objects and Actions

Register : New users can register to the system by filling the required form.

Login : Users have to login in order to join the game. After logging in, a user can use other functionalities.

Join Table: Users can join any existing tables which is not full.

Create Table : Users can create table and wait for other players, if they do not want to join an existing table.

Add Agent : If a user wants to play against the game bots, he can simply add an agent with a chosen level.

Chat : Users can easily chat with other users through our chatbox.

Check Game State : Users may want to see the status of the game during playing. It is enough to click button in order to see the scores.

Dismiss Agent : If a user wants to add a new human player, he can easily dissmiss the agent from table.

7. Libraries and Tools

<u>ActionScript :</u> It is a script language and it is used primarily for the development of websites and software targeting the Adobe Flash Player platform. We will use it to implement interface of game.

Java SE 1.6 : is a widely used platform for programming in the Java language. We will use it for developing main features of the Agent

JDBC : is an API in Java programming language that defines how a client may connect to a database. It provides methods for querying and updating data in a database.

MySQL : is a relational DB management system that runs as a server providing multi-user access to a number of databases.

Netbeans : is an integrated development environment (IDE) for developing with Java.

SmartDraw : is a tool that is used for drawing Gannt Chart and ER diagram.

Umbrello : Umbrello handles all the standard UML diagram types.

Web Service : is typically an API that is accessed via http and executed on a remote system, hosting the requested service.

Weka : is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License.

8. Time Planning (Gannt Chart)

There is no strict deadline dates for phases but the complete system is going to be finished until June. To distribute the workload equally during the Project development, the following schedule may be followed.

• Design and implementation of various agents	End of January
• Collecting training data and classification of data	Mid of March
• Defining states and trying different machine learning algorithms	End of April
• Testing and Integration	Mid of June

8.1. Semester 1-2 Gannt Chart

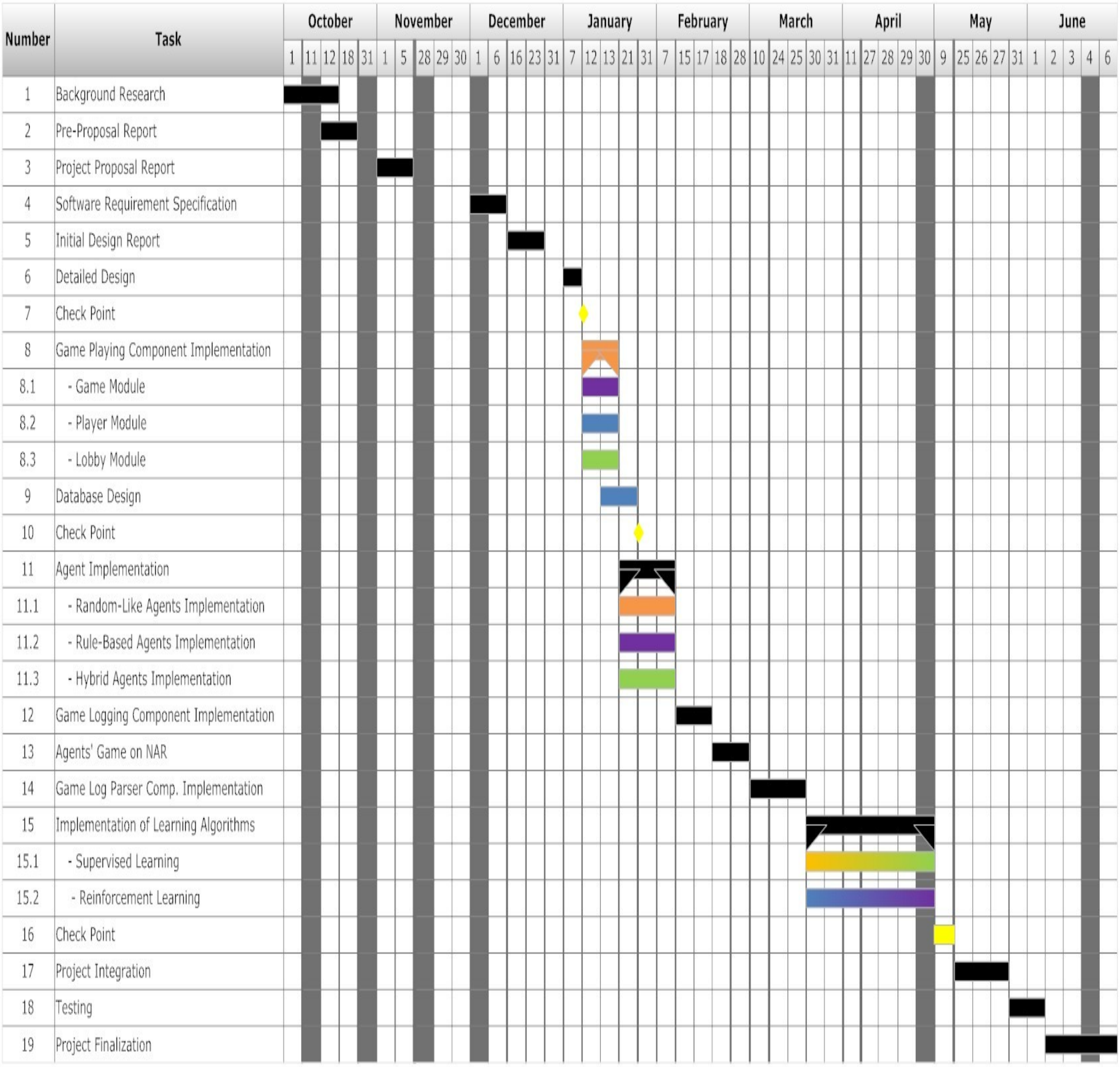


Figure 15: Gannt Chart

9. Conclusion

In conclusion, the Gambler Agent Project will be a world wide, sensational Project for AI and Machine Learning field. If the final learning agent can model the training data, later on it will be able to model the best 'king' players in the world. It will be a great opportunity for human players to play against the Gambler Agent and most famous players are going to desire to challenge with Gambler Agent.The experiment of Gambler agent project will give a result that can be positive or negative.However the experiment result will be published as an academic article and will be sent to some artificial intelligence conferences.Also this project will be used in some online game contests.