

**Card Game Engine**  
*Specifics for Card Game: Idiot*

**Formal Design Specification**

**Version 1.0**

**October 19, 2015**

**Kurt E. Clothier**  
**Lovedeep Gondara**  
**Kyle Kampfen**  
[Group 5]

**CSC 578 B — Software Engineering**  
**University of Illinois Springfield**  
**Instructor: West, Roger, Ph.D.**  
**Fall 2015**  
**Revision History**

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

<b>Date</b>	<b>Version</b>	<b>Description</b>	<b>Author</b>
<b>10/19/2015</b>	<b>1.0</b>	<b>First Edition</b>	<b>Group 5</b>

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

# Table of Contents

## [1. Terminology](#)

### [1.1 Definitions, Acronyms and Abbreviations](#)

### [1.2 Game components](#)

## [2. Component Template Design](#)

### [2.1 Cards](#)

#### [2.1.1 Keywords](#)

#### [2.1.2 Example Use](#)

### [2.2 Game Board](#)

#### [2.2.1 Keywords](#)

#### [2.2.2 Example Use](#)

### [2.3 Rules](#)

#### [2.3.1 Keywords](#)

#### [2.3.2 Example Use](#)

### [2.4 Players](#)

### [2.5 Game](#)

#### [2.5.1 Pseudocode](#)

### [2.6 Game Engine](#)

## [3. Graphical User Interface](#)

## [Appendix A : Diagrams](#)

### [Figure A-1 - Model Driven Architecture](#)

### [Figure A-2 - Game Engine State Diagram](#)

### [Figure A-3 - Idiot Game Board Rendering](#)

### [Figure A-4 - Game Engine Class Diagram](#)

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

# 1. Terminology

## 1.1 Definitions, Acronyms and Abbreviations

GUI	Graphical User Interface
UI	User Interface
UML	Unified Modelling Language

## 1.2 Game components

Game Component	File system component	File format	Coupling
Cards	deck.template.txt	txt	
Rules	rules.template.txt	txt	Cards, Board
Game board	board.template.txt	txt	
Players	User Prompt	n/a	
Game	Java Object	.java	Game Engine
Game engine	Java Source Code	.java	Game

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

## 2. Component Template Design

In order for the game engine to correctly read game component data from external text files, a format for those file must be created. Each type of component file uses a similar format with various keywords in order to convey appropriate data to the game engine. See requirement specification IDs C1, R1, and G1.

In these files, comments may be placed using `<#>` and terminate at the end of the line. Capitalization is ignored entirely. In-line whitespace is ignored except where whitespace is used to separate numerous parameters. This file consists of numerous keywords, followed by parameters. The game engine will be responsible for locating the keywords in the file and parsing the following parameter values. The Interpreter Pattern is used to parse information from these text file in software.

### 2.1 Cards

In accordance with specification ID C2, the external cards file is referred to as a “deck.” This deck of cards is represented by a plain text file using the naming format: `deck.<NAME>.txt` (ID: C3) where `<NAME>` should be substituted for the name of the deck, for example: `deck.french_suits.txt`. Variants of this card deck can be specified by hyphenating the variant after the standard deck name (ID: C4), for example: `deck.french_suits-with_jokers.txt`.

#### 2.1.1 Keywords

This table provides the valid keywords followed by a description of the accompanying parameters. At any time, a value of “null” can be used to specify the absence of that parameter.

keyword	parameter format	description
deck	string	the name of this deck
size	whole number	how many cards are in the deck
faces	string, csv	comma separated faces of the cards in this deck
grouping	string	how these cards are grouped, for example: suits, colors, etc.
groups	string, csv	comma separated groups, for example: clubs, hearts, spades, diamonds
specify-quantity	<yes, no>	yes if specifying the quantity of cards after the “cards” keyword
cards	numbers, csv	comma separated cards, using numbers in place of the faces and groups above (starting with 0). Whitespace is completely ignored, and cards can be separated onto new lines with or without a trailing comma, or can all be on one long line.

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

## 2.1.2 Example Use

Here is an example deck file, using the standard 52 card French Suits deck: *deck.french\_suits.txt*.

```

deck french_suits
size 52
faces null,ace,2,3,4,5,6,7,8,9,10,jack,queen,king
grouping suits
groups hearts,diamonds,clubs,spades
specify-quantity no
cards
    1,0,1,1,1,2,1,3      # Ace of Hearts, Diamonds, Clubs, Spades
    2,0,2,1,2,2,2,3      # 2
    3,0,3,1,3,2,3,3, 4,0,4,1,4,2,4,3, 5,0,5,1,5,2,5,3
    6,0,6,1,6,2,6,3, 7,0,7,1,7,2,7,3, 8,0,8,1,8,2,8,3
    9,0,9,1,9,2,9,3, 10,0,10,1,10,2,10,3
    11,0,11,1,11,2,11,3   # Jack
    12,0,12,1,12,2,12,3   # Queen
    13,0,13,1,13,2,13,3   # King

```

Notice the use of “null” as the first card face. This is done so that the number representation used to define the cards after the *cards* keyword matched the value of the numeric card faces. This is optional, but can be done to enhance understanding in the file if desired.

Here is another small example, this time specifying the quantity of each card in an Uno deck: *deck.uno.txt*.

```

deck uno
size 108
faces 0,1,2,3,4,5,6,7,8,9,skip,draw_two,reverse,wild,wild_draw_four
grouping colors
groups wild,red,yellow,green,blue
specify-quantity yes
cards
    0,1,1, 0,2,1, 0,3,1, 0,4,1      # 1 '0' in each color
    1,1,2, 1,2,2, 1,3,2, 1,4,2      # 2 '1s' in each color
    2,1,2, 2,2,2, 2,3,2, 2,4,2
    3,1,2, 3,2,2, 3,3,2, 3,4,2, 4,1,2, 4,2,2, 4,3,2, 4,4,2, 5,1,2, 5,2,2, 5,3,2, 5,4,2
    6,1,2, 6,2,2, 6,3,2, 6,4,2, 7,1,2, 7,2,2, 7,3,2, 7,4,2, 8,1,2, 8,2,2, 8,3,2, 8,4,2
    9,1,2, 9,2,2, 9,3,2, 9,4,2
    10,1,2, 10,2,2, 10,3,2, 10,4,2 # 2 'skips' in each color
    11,1,2, 11,2,2, 11,3,2, 11,4,2 # draw 2
    12,1,2, 12,2,2, 12,3,2, 12,4,2 # reverse
    13,0,4                                     # 4 'wild' cards
    14,0,4                                     # 4 'wild draw 4' cards

```

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

## 2.2 Game Board

Similar to the deck file, a game board file is created to represent the board layout for a chosen game (ID: G1). This file is interpreted by the game engine (G4) and is named as such: board.<GAME>.txt or board.<GAME>-<VARIANT>.txt (ID: G2, G3).

### 2.2.1 Keywords

This table provides the valid keywords followed by a description of the accompanying parameters. At any time, a value of “null” can be used to specify the absence of that parameter.

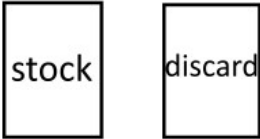
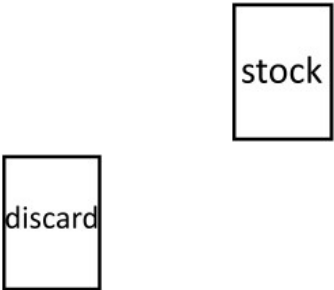
keyword	parameter format	description
game	string	the name of the game described by this board file
cardpile	n/a	this keyword should be on its own line
<b>cardpile specific parameters - start on a newline after ‘cardpile’</b>		
name	string	name of this cardpile - must be unique to the owner
owner	<common, player>	common - owned by all players, such as a discard pile player - pile owned by each player (every player has 1 of this pile)
visibility	<all, owner, other, none>	specifies who can see the cards in this pile, if any are face up all - cards visible to all players    owner cards visible only to owner other - visible to all but owner    none - visible to no one (face down)
visible	<[0,n], all, none, top>	Sets the number of face up cards on top of the pile. [0,n] - a whole number    none - no cards are face up all - all cards are face up    top - the top card is always face up
placement	<stack,spread, spaced, messy>	Specifies how the cards are placed on the table. stack - all cards placed directly on top of one another spread - cards are placed slightly overlapping one another spaced - cards are placed beside one another, not touching messy - cards are in a disorderly pile
orientation	<landscape, portrait>	specifies how the cards are placed with regard to the owner landscape - longways    portrait - upright
tiling	<horizontal, vertical>	specifies how spread and spaced cards are placed and dealt horizontal - left to right    vertical - bottom to top
removal	<top, bottom, random, any, none, all>	Specifies how cards are removed from a pile (played, dealt, etc) top - top card removed first bottom - bottom card removed first

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

		random - a random card is removed any - any card can be selected none - no cards may be removed from this pile all - all cards are removed at once
board-layout	n/a	this keyword should be on its own line
<b>board-layout specific parameters - start on a newline after 'board-layout'</b>		
shape	<sided, circle, semi-circle>	specifies the shape of the playing area. the size is dynamic with respect to the number of players
dealer	<yes, no>	yes, if a space is reserved for a NPC dealer
player-layout	<spread, semi-spread>	spread - players are spread evenly around the perimeter semi-spread - players are equally spaced around one side
common-piles	n/a	this keyword should be on its own line
player-piles	n/a	this keyword should be on its own line
<b>&lt;owner&gt;-piles specific parameters - start on a newline after '&lt;owner&gt;-piles'</b>		
<name><filler>	filler = <glue, space>	glue - expands as needed, pushes piles as far as it can space - single rigid area, half the width of a playing card

## 2.2.2 Example Use

The <owner>-piles keywords are used to describe the layout of these piles on the game board. Two filler keywords are allowed: glue and space. The first example shows a simple layout using two common piles, stock and discard.

Layout	Text
	common-piles glue stock space discard glue
	common-piles glue stock glue discard glue



Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

This final example is the full layout for the game of Uno: *board.uno.txt*.

game uno

cardpile # stock card pile is the left over cards after hands are dealt

name stock  
owner common  
visibility none  
visible none  
placement stack  
orientation portrait  
removal top

cardpile # discard pile holds cards discarded by the players

name discard  
owner common  
visibility all  
visible all  
placement stack  
orientation portrait  
removal all

cardpile # a player hand

name hand  
owner player  
visibility owner  
visible all  
placement spread  
orientation portrait  
tiling horizontal  
removal any

board-layout

shape sided  
dealer no  
player-layout spread

# Common card piles Layout

common-piles  
glue stock space discard glue

# Player card piles layout

player-piles  
glue hand glue

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

## 2.3 Rules

A rules file is created to represent the board layout for a chosen game (ID: R1). This file is interpreted by the game engine (R4) and is named as such: rules.<GAME>.txt or rules.<GAME>-<VARIANT>.txt (ID: R2, R3).

This section is incomplete at this time. It will be appended once the final design of the rules text file is complete.

### 2.3.1 Keywords

This table provides the valid keywords followed by a description of the accompanying parameters. At any time, a value of “null” can be used to specify the absence of that parameter.

keyword	parameter format	description
game	string	the name of the game described by this rules file
deck	string	name of the deck to be used (without any file pre or post fix)
ranking	csv, low->high	ranking of the faces from the specified card deck
alias	<face> <alias>	rename any special cards, such as: <i>alias 2 deuce</i>
board	string	name of the board to be used (without any file pre or post fix)
direction-of-play	<cw, ccw>	clockwise or counter-clockwise direction of game play
...		

### 2.3.2 Example Use

This section is a stub, it should be expanded upon once the keywords are updated.

Example rules for the game of idiot: *rules.idiot.txt*.

```
game idiot
deck french_suits
ranking 2,3,4,5,6,7,8,9,10,jack,queen,king,ace
alias 2 restart
alias 5 reverse
alias 10 burn
board idiot
direction-of-play cw
```

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

## 2.4 Players

Players for the card game are prompted to enter any information requested by the game. This is done for one or multi-player games (ID: P2, P3). This user prompt will be handled by the game engine itself, once a game has been selected.

Although at least one player must be human (ID: P1), it is possible to expand the capability of the engine to handle AI agents. These agents will have a specific personality file which instructs an AI object how to play the game as well as any strategies. The implementation of this feature is not necessary.

## 2.5 Game

A game is an individual object created by the game engine (ID: G11). It is not self-aware, and must be driven by another program. The game contains a collection of other game components, such as objects representing the playing cards, deck, card piles, and playing area.

In Java, this is the *CardGame* class. This class is responsible for reading the rules file of the selected game, and creating all another necessary components such as a deck of cards.

### 2.5.1 Pseudocode

Because the *CardGame* class requires a driver, the only relevant code is that which takes place in the object constructor.

```
rules = new FileCopy(rulesFile) // Create a FileCopy of the rules text file
// parse information in the file for the card deck,
// parse information in the file for the deck rankings
// create Deck object, passing appropriate deck file
deck = new Deck(deckFile, rankings)
parse information in the file about the board
create GameBoard, passing appropriate board file

// parse information about how to deal cards, create Action objects
// parse information about various rules, create Rule objects
```

Aside from this, the *CardGame* will service any requests from the driver, performing actions as directed and returning object and game data and information where appropriate.

## 2.6 Game Engine

This section is incomplete and will be filled in once the final design for the game engine has been established.

Card Game Engine - Specifics for Card Game: <i>Idiot</i>	Version: 1.0
Formal Design Specification - Group 5	Date: 10/19/2015

The game engine is a term used to describe the entire code set used to handle card based games and is written in Java. An overview of the engine state is found in *Figure A-2*. In accordance to requirements GE1 - GE13, the game engine is capable of reading information about a particular game from various text files, and creating objects to represent the information in the text files. This is handled by a number of utility classes:

- PlayingCard - a single playing card in a card game
- Deck - a collection of playing cards for a card game
- FileCopy - a copy of the relevant information contained in the component text files
- CardPile - a single pile of cards used in a card game
- FileIO - Static utilities for listing and reading text files
- GameBoard - the layout of the card game
- Player - information about a player of this card game
- CardGame - a grouping of all of the other important game components

The interaction between these classes is illustrated in *Figure A-4*. The GameEngien uses a combination of the Bridge and Adaptot structural patterns to connect the game components and GUI.

### 3. Graphical User Interface

This section is incomplete and will be filled in once the final design for the GUI has been established.

In accordance to requirements ID UI1, the user is to interact to the game by way of a GUI. Because the exact game to played is unknown, this GUI must be built dynamically after the game board file has been read. However, there are common elements to be used. An example layout is shown in *Figure A-3*. The primary components of the GUI use an Abstract Factory pattern.

## Appendix A : Diagrams

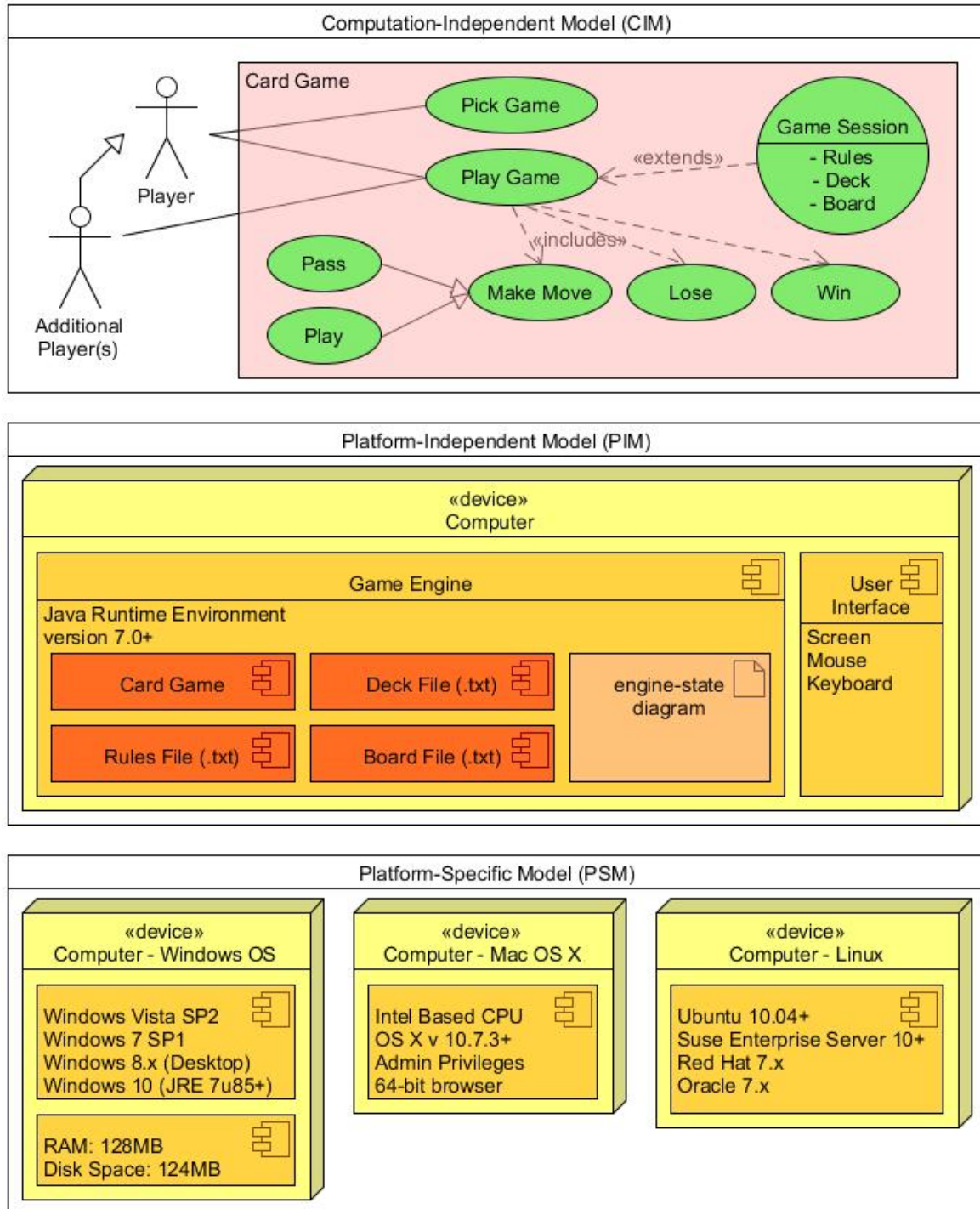
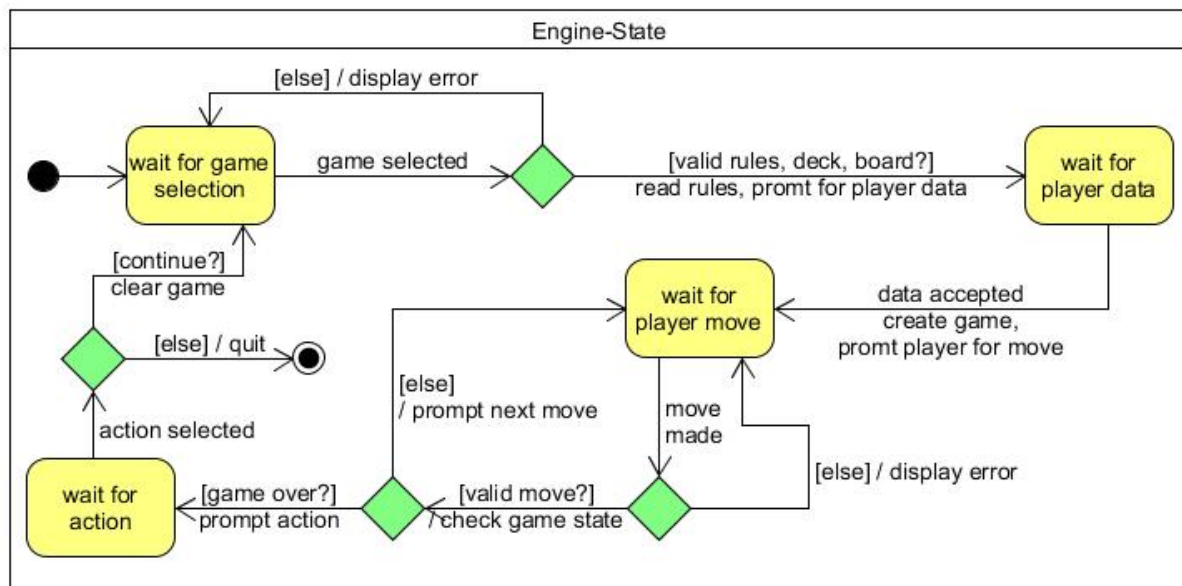


Figure A-1 - Model Driven Architecture



*Figure A-2 - Game Engine State Diagram*

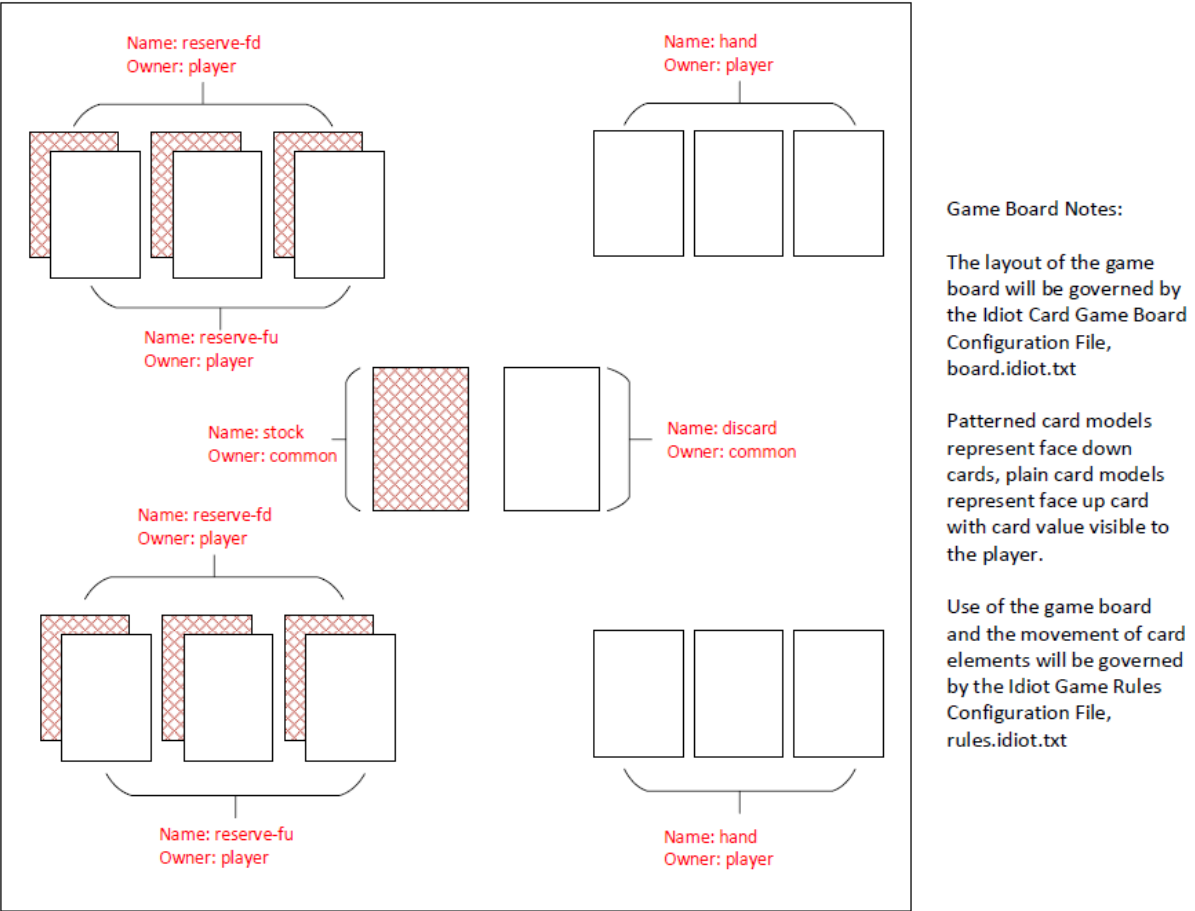


Figure A-3 - Idiot Game Board Rendering

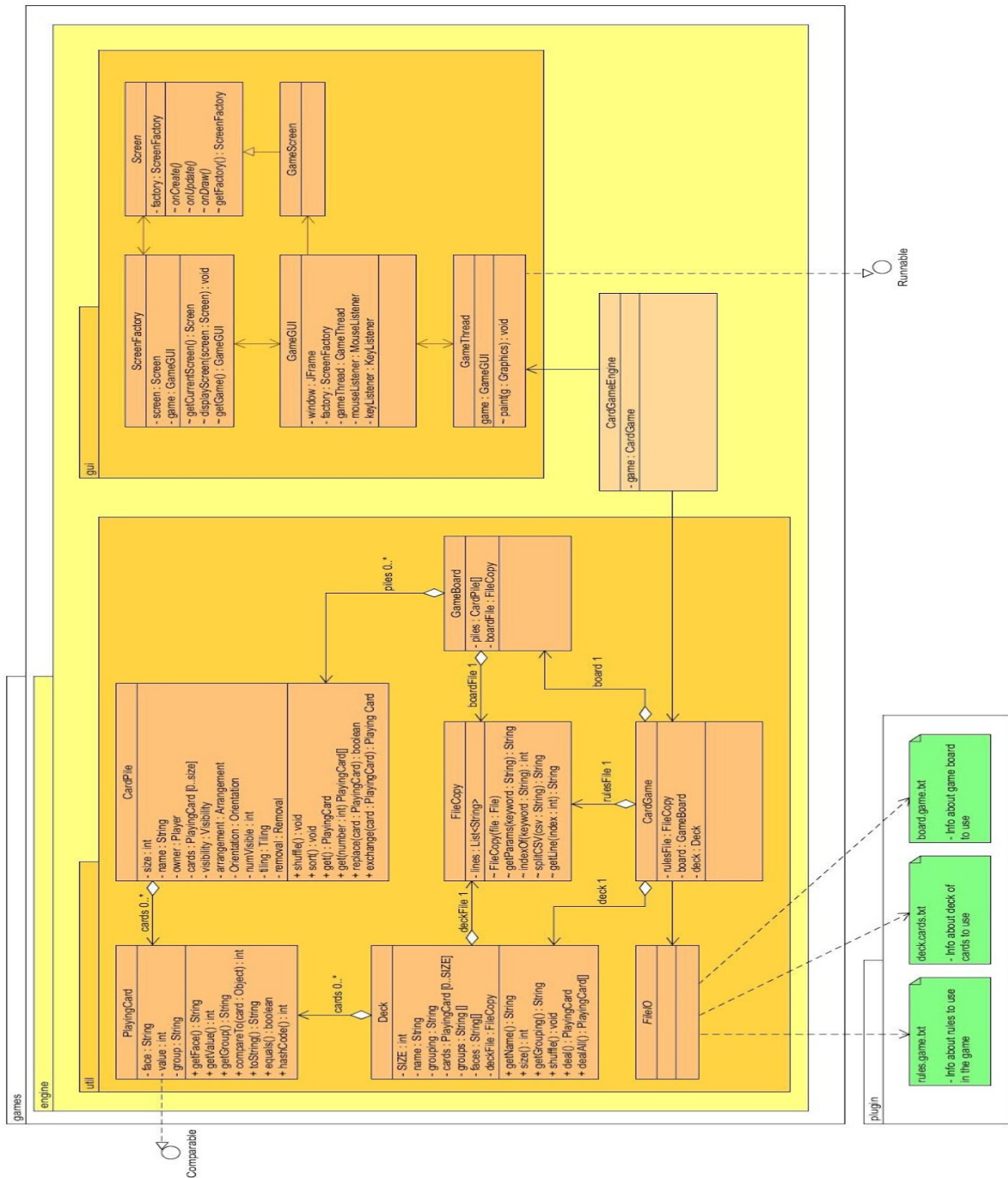


Figure A-4 - Game Engine Class Diagram