Troop War Documentation

Created by

Chanon Rattanajaratkul 6331308021

Supakorn Ratnagupt 6331346821

2110215 Programming Methodology

Semester 2 Year 2020

Chulalongkorn University

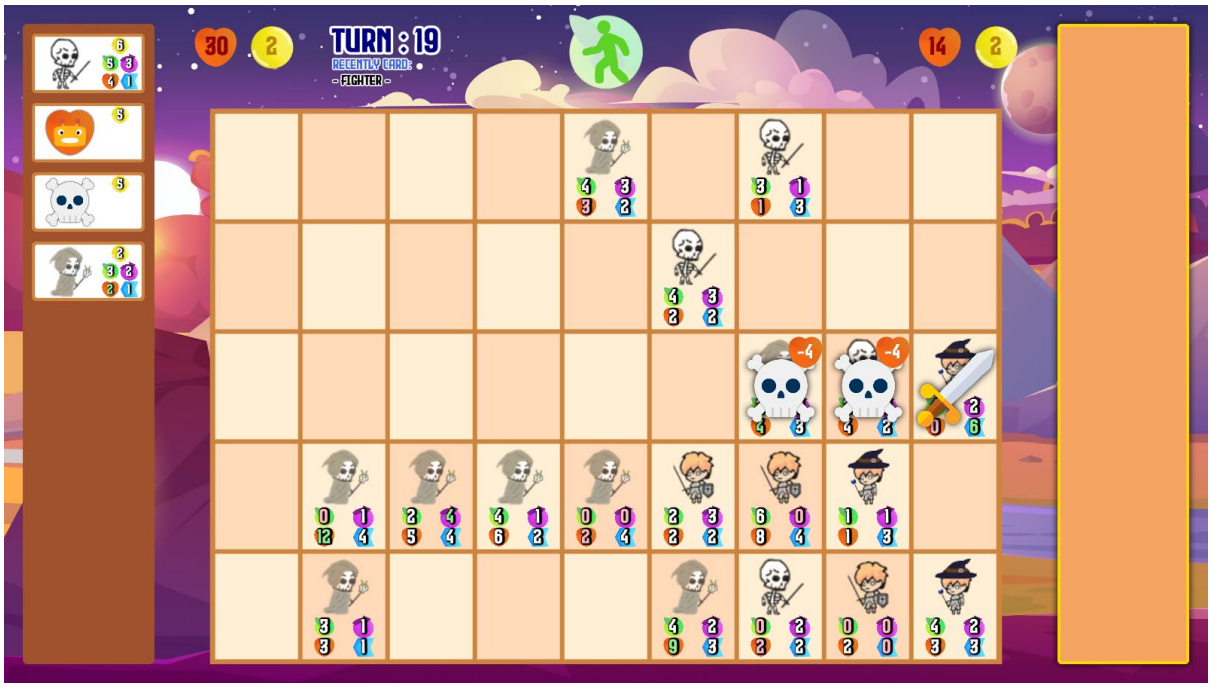
Troop War

Introduction

Nowadays, there are a lot of games for us to play. One type of games that we, the game developers, interest is card game. Card game has long story, it includes the card we played our daily life. It benefits the players to improve patience, concentration and also boost motor skills. So, we decided to create card game which require strategic planning skill, and decisive actions to win the game. The game we created was inspired by many strategic games, such as Plant vs. Zombies Heroes, The Battle Cats, and Line Rangers.

Rules

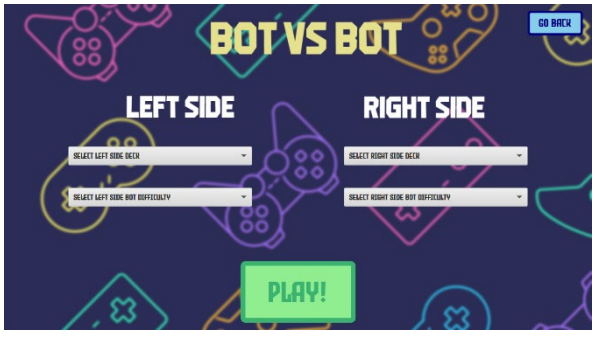
There are three game mode in TroopWar: PvP (Player vs. Player), PvB (Player vs. Bot) and BvB (Bot vs. Bot). Player and Bot is controller types. You can control the player. Bot is auto play. When game start, there are two playing side: left and right. Both controller side have health, money and 4 start cards. The player would gain money every turn, so that we can spend to play the cards in his/her hand, which classify into 3 types: the fighter, the magician, and the trick. When the fighter card can be placed on the board one cell in front of player's side, it would be the troop, which is going to attack the opponent player. The magician is like the fighter card, but it has the tricks that the fighter card doesn't have. The trick card has a lot of effects on the game, it can increase the troop's attack damage, attack range, speed, or health. It can also destroy the card, change players health, or make players draw more cards. The income that the player get will increase by 1 each turn (max is 20), for example, in turn 1 each player gets 1, turn 2 gets 2. You will not lose money in the end of turn. Each turn there are 4 phases. First is draw cards. Second is play phase. Playing side which play first will be random in first turn. In next turn, side that play first will play after. Third is move phase. The side that plays first will move card after. Forth is fight phase. If card can attack enemy in their row (enemy in its attack range), it will attack. Enemy card health will reduce by attack damage of attacker. Cards that have health 0 will dead but still can attack until be removed after attacking finish in each row. When the troops are in the 1 cell in front of the opponent side, they would sacrifice themselves to attack the controller which makes the controller health decreases. The controller will win the game if the opponent controller has 0 health.

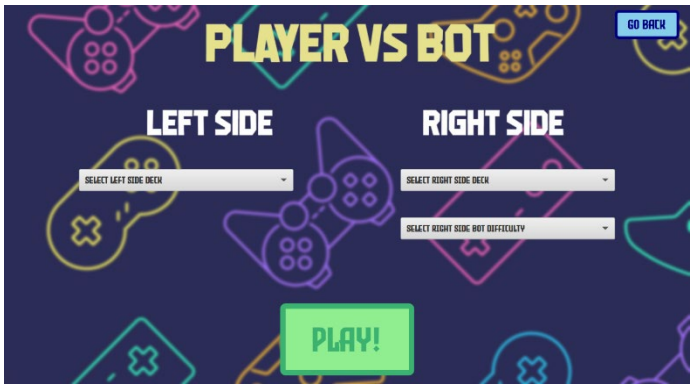


Scene

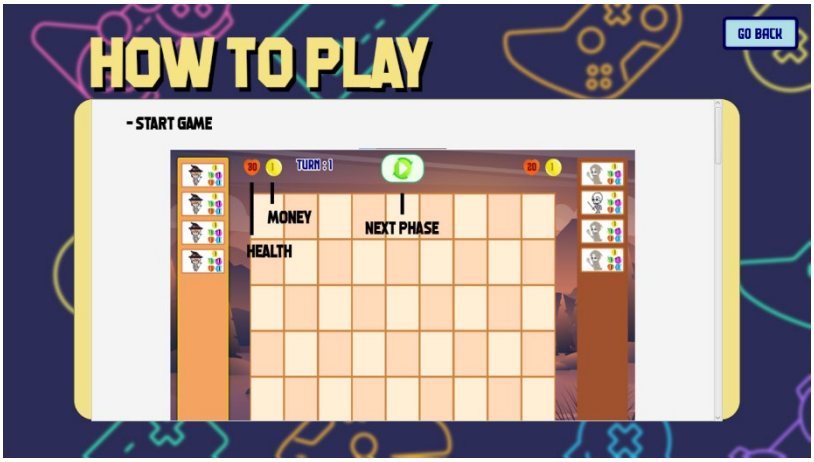
SelectGameModeScreen

SettingScreen

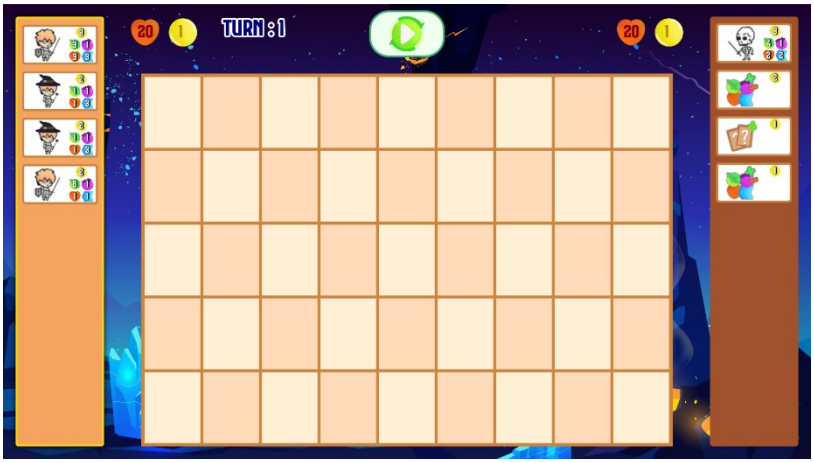




HowToPlayScreen

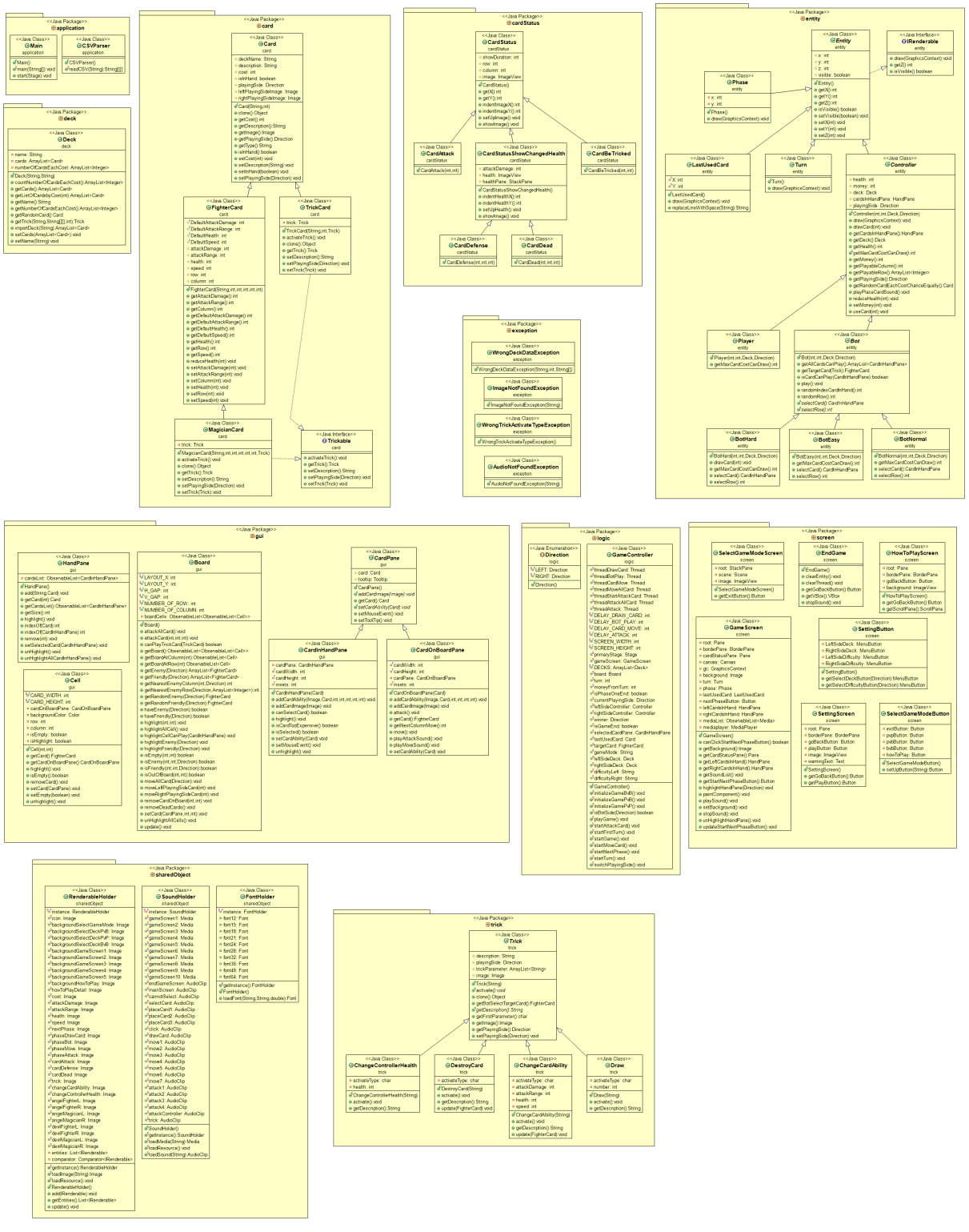


GameScreen



EndGameScreen

Class diagram



** Noted that Access Modifier Notations are listed below.*

+ (public)

(protected)

- (private)

Underline (static)

Italic (abstract)

1. Package application

1.1 Class CSVParser

This class is used to open and read csv files which containing cards data.

1.1.1 Methods

Name	Description
+ String[][] readCSV(String filename)	Receive filename and open .csv file in res/csv folder. Read file and convert into String[][] and return it.

1.2 Class Main extends Application

1.2.1 Methods

Name	Description
+ void main(String[] args)	Launch application
+ void start(Stage primaryStage)	Set scene to SelectGameModeScreen Initialize primaryStage Set primaryStage resizable false Show the primaryStage using function provided by javafx

2. Package card

2.1 Class Card implements Cloneable

This class is for storing card data. There are 3 card types (Fighter, Magician, Trick), that inheritance from this class.

2.1.1 Fields

Name	Description
# String deckName	Deck name of this card
# Int cost	Cost for playing this card
# Boolean isInHand	True, if it in controller hand
# Image leftPlayingSideImage	card image for left playing side
# Image rightPlayingSideImage	card image for right playing side
# String description	Card description
# Direction playingSide	Side of this card (LEFT or RIGHT)

2.1.2 Constructor

Name	Description
+ Card(String deckName, int cost)	Initialized each card according to the deckName and cost parameter Set description as “” (blank)

2.1.3 Methods

Name	Description
+ Object clone()	Return card we want to copy
+ Image getImage()	Return card image according to playingSide field
+ String getType()	Return type of the card; Fighter, Magician or Trick
+ void setCost(int cost)	Set cost which can’t below 0
+ getter and setter	

2.2 Class **FighterCard** extends Card

Fighter card can place on board. It can move and attack.

2.2.1 Fields

Name	Description
# final int DefaultAttackDamage	Normal value of Attack damage
# final int DefaultAttackRange	Normal value of Attack range
# final int DefaultHealth	Normal value of Health

# final int DefaultSpeed	Normal value of Speed
# int attackDamage	Attack damage for each attack
# int attackRange	Number of cell in front of this card that it can attack
# int health	Health of card
# int speed	Number of cell that card will move forward in each turn
# int row	Position of the card in y axis (up-down)
# int column	Position of the card in x axis (left-right)

2.2.2 Constructor

Name	Description
+ FighterCard(String deckName, int cost, int attackDamage, int attackRange, int health, int speed)	Initialized card according to deckName, cost, attackDamage, attackRange, health, and speed. Set description as “” (blank). Set leftPlayingSideImage and rightPlayingSide according to deckName.

2.2.3 Methods

Name	Description
+ void setAttackDamage(int attackDamage)	Set attackDamage which can’t below 0
+ void setAttackRange(int attackRange)	Set attackRange which can’t below 0
+ void setHealth(int health)	Set health which can’t below 0
+ void setSpeed(int speed)	Set speed which can’t below 0
+ void reduceHealth(int attackCard)	Set new health if the card is attacked by attackCard parameter
+ getter and setter	

2.3 **Class MagicianCard** extends FighterCard implements Trickable

Magician card is Fighter card but it has trick. Trick will activate when play this card. There are 4 abilities of trick in the game ChangeCardAbility (it will change ally’s or opponent’s attack damage, attack range, health, or speed), DestroyCard (it will destroy a card on the board), Draw (it will make the controller draw the cards) and ChangeControllerHealth (it will reduce or increase the controller’s health)

2.3.1 Fields

Name	Description
- Trick trick	Trick that magician can use

2.3.2 Constructor

Name	Description
+ MagicianCard(String deckName, int cost, int attackDamage, int attackRange, int health, int speed, Trick trick)	Initialized card according to deckName, cost, attackDamage, attackRange, health, speed, and trick. Set description according to trick description by method setDescription(). Set leftPlayingSideImage and rightPlayingSide according to deckName.

2.3.3 Methods

Name	Description
+ void activateTrick()	Activate the trick
+ Object clone()	Return magicainCard we want to clone
+ String setDescription()	Return trick.getDescription()
+ void setPlayingSide(Direction playingSide)	Set playing side of the magician and its trick
+ getter and setter	

2.4 Interface Trickable

Magician card and Trick card can trickable. They have trick that can be activated.

Trick will activate when controller play these card.

2.4.1 Methods

Name	Description
+ void activateTrick()	Activate the trick
+ Trick getTrick()	Get trick
+ void setTrick(Trick trick)	Set trick
+ void setPlayingSide(Direction playingSide)	Set playing side
+ String setDescription()	Set Description

2.5 **Class TrickCard** extends Card implements Trickable

Trick card can't place on board. When controller play this card, the trick will be activated and the card will disappear. There are 4 abilities of trick in the game which are ChangeCardAbility (it will change ally's or opponent's attack damage, attack range, health, or speed), DestroyCard (it will destroy a card on the board), Draw (it will make the controller draw the cards) and ChangeControllerHealth (it will reduce or increase the controller's health)

2.5.1 Fields

Name	Description
- Trick trick	Trick that card use

2.5.2 Constructor

Name	Description
+ TrickCard(String deckName, int cost, Trick trick)	Initialized card according to deckName, cost, and trick

2.5.3 Methods

Name	Description
+ void activateTrick()	Activate the trick
+ void setPlayingSide(Direction playingSide)	Set card playingSide and trick playingSide
+ String setDescription()	Set trick description
+ void setPosition(int row, int column)	Set position according to row and column parameter
+ Object clone()	Return trickCard we want to clone
+ getter and setter	

3. Package cardStatus

3.1 Class CardAttack extends CardStatus

This class is for showing image when card is attacking.

3.1.1 Constructor

Name	Description
+ CardAttack(String Name, String fileName)	Show attack card image

3.2 Class CardBeTricked extends CardStatus

This class is for showing image when be tricked.

3.2.1 Constructor

Name	Description
+ CardBeTricked(String Name, String fileName)	Show card being tricked image

3.3 Class CardDead extends CardStatusShowChangedHealth

This class is for showing image when be attacked and dead.

3.3.1 Constructor

Name	Description
+ CardDead(String Name, String fileName)	Show card dead image

3.4 **Class CardDefense** extends CardStatusShowChangedHealth

This class is for showing image when card defense (be attacked but not dead).

3.4.1 Constructor

Name	Description
+ CardDefense(String Name, String fileName)	Show card defense image

3.5 Class CardStatus

Card status is for showing image when card attack, be tricked, dead or defense.

3.5.1 Fields

Name	Description
# int showDuration	Duration of image appearance
# int row	Row in the board, that will appearance
# int column	Column in the board, that will appearance
# ImageView image	Status image

3.5.2 Methods

Name	Description
+ int getX()	Calculate and return x-coordinate of screen according to column
+ int getY()	Calculate and return y-coordinate of screen according to row
+ int indentImageX()	Return value to plus x-coordinate to make position more accurate
+ int indentImageY()	Return value to plus y-coordinate to make position more accurate
+ void setUpImage()	Set position and size of status image
+ void showImage()	Use setUpImage(), add all images to game screen's CardStatusPane, make the image appear as long as showDuration, after that remove all images from game screen's CardStatusPane

3.6 **Class CardStatusShowChangedHealth** extends CardStatus

Card status inheritance from this will also show image of health reducing.

3.6.1 Fields

Name	Description
# int attackDamage	Attack damage of attacker card
# ImageView health	The health icon that will show in the screen
# StackPane healthPane	StackPane contain health image and amount of reduceing health

3.6.2 Methods

Name	Description
+ int indentImageX()	Return value to plus x position to make position more accurate
+ int indentImageY()	Return value to plus y position to make position more accurate
+ void setUpHealth()	Set position and size of health image and text. Add images and text to the healthPane
+ void showImage()	Use setUpImage(), setUpHealth() add all images to game screen's CardStatusPane, make the image appear as long as showDuration, after that remove all images from game screen's CardStatusPane

4. Package deck

4.1 Class Deck

Class deck is for storing cards data

4.1.1 Fields

Name	Description
- String Name	Name of deck
- ArrayList<Card> cards	List of cards in deck

- ArrayList<Integer> numberOfCardsEachCost	List of number of cards that have this cost by index (i.e. index 1 contain number of cost 1 card)

4.1.2 Constructor

Name	Description
+ Deck(String Name, String fileName)	Set Name of deck. Set cards equal to return of importDeck(fileName) . Set numberOfCardsEachCost by use method countNumberOfCardsEachCost. Add this deck to GameController.DECKS

4.1.3 Methods

Name	Description
+ ArrayList<Integer> countNumberOfCardsEachCost()	Count the number of card to the specific card's cost and return a list that index=cost and values=number of cards (i.e. index 1 contain number of cost 1 card)
+ ArrayList<Card> getListOfCardsbyCost(int cost)	Return list of cards that have specific cost
+ Card getRandomCard()	Return random card from cards
+ Trick getTrick(String trick, String[][] deckData, int row)	Return the trick of the card according to trick, deckData, and row parameter
+ ArrayList<Card> importDeck(String filename)	Import deck data form .csv file and create every card in deck
+ getter and setter	

5. Package entity

5.1 **Class Bot** extends Controller

Bot is controller that can play the game itself.

5.1.1 Constructor

Name	Description
+ Bot(int health, int money, Deck deck, Direction playingSide)	Initialize bot as the controller

5.1.2 Methods

Name	Description

+ ArrayList<CardInHandPane> getAllCardsCanPlay()	Return list of CardInHandPane from CardsList.
+ FighterCard getTargetCard(Trick trick)	If the trick first parameter is “C”, return the random friendly troop. If the trick first parameter is “D”, return the random enemy troop
+ boolean isCardCanPlay(CardInHandPane cardPane)	Return true if the cost of the card is less than or equal to the money it have. Some tricks have to have the target, if not then they cannot be placed. canPlayTrickCard(TrickCard trickCard) function may be useful.
+ void play()	Bot will play the card until it can't play. Select a card by call method selectCard(). Use a card by call method useCard().
+ int randomIndexCardInHand()	Return random index of the card in bot hand
+ int randomRow()	Return random row to place a card on board
+ CardInHandPane selectCard()	Logic for selectCard depend on bot easy, normal or hard.
+ int selectRow()	Logic for selectRow depend on bot easy, normal or hard.

5.2 **Class BotEasy** extends Bot

5.2.1 Constructor

Name	Description
+ BotEasy(int health, int money, Deck deck, Direction playingSide)	Initialize bot as the controller

5.2.2 Methods

Name	Description
+ int getMaxCardCostCanDraw()	Return max card cost that can draw card (turn + 4)
+ CardInHandPane selectCard()	Return the random card the bot will place
+ int selectRow()	Return the random row that can play from getPlayableRow()

5.3 Class **BotNormal** extends Bot

5.3.1 Constructor

Name	Description
+ BotNormal(int health, int money, Deck deck, Direction playingSide)	Initialize bot as the controller

5.3.2 Methods

Name	Description
+ int getMaxCardCostCanDraw()	Return max card cost that can draw card (turn + 2)
+ CardInHandPane selectCard()	First random the trickable card that can be placed. If not have any, it will play any cards instead.
+ int selectRow()	70% chance to return the row that have nearest enemy card and can play first. If not have any, it will return random row that can play. 30% chance to return the random row that can play. (Use getPlayableRow() to get row that can play)

5.4 Class **BotHard** extends Bot

5.4.1 Constructor

Name	Description
+ BotHard(int health, int money, Deck deck, Direction playingSide)	Initialize bot as the controller

5.4.2 Methods

Name	Description
+ void drawCard(int number)	Draw cards. Cards will have 30% chance to get 1 extra health. Add cards into hand. If the number of cards in the hand will exceeds 9, it can’t draw the card
+ int getMaxCardCostCanDraw()	Return max card cost that can draw card (turn + 2)
+ CardInHandPane selectCard()	First random the trickable card that can be placed. If not have any, it will play any cards instead.

+ int selectRow()	Return the row that have nearest enemy card and can play first. If not have any, it will return random row that can play.

5.5 Class Controller extends Entity

Controller are player or bot.

5.5.1 Fields

Name	Description
- int health	Health of controller
- int money	Money of controller
- Deck deck	Deck that controller uses
- HandPane cardsInHandPane	List of cards in controller hand
- Direction playingSide	Left playing side or Right playing side

5.5.2 Constructor

Name	Description
+ Controller(int health, int money, Deck deck, Direction playingSide)	Set health. Must not lower than 1. Set money. Set playingSide.

5.5.3 Methods

Name	Description
+ void draw(GraphicsContext gc)	Draw health and money image.
+ void drawCard(int number)	Draw card. Add cards into hand. If the number of cards in the hand will exceeds 9, it can't draw the card
+ Card getRandomCardEachCostChanceEqually()	Return random card from controller's deck but chance to get card each cost is equally. (This method is not used)
+ int getPlayableColumn()	Return column in front of their sides
+ ArrayList<Integer> getPlayableRow()	Return list of rows that card not placed yet
+ void playPlaceCardSound()	Play placing card sound
+ void reduceHealth(int number)	Reduce controller health. If it less than 0, opponent wins the game. Show EndGame scene.
+ void useCard(int index)	Use the card in the hand and remove it. Decrease money.
+ getter and setter	

5.6 **Class Entity** implements IRenderable

Entity is every thing that appear on screen and set position by x,y.

5.6.1 Fields

Name	Description
- int x	The x coordinates of the screen
- int y	The y coordinates of the screen
- int z	For ordering entity
- boolean visible	Visibility of the object

5.6.2 Methods

Name	Description
+ getter and setter	

5.7 Interface IRenderable

5.7.1 Fields

Name	Description
+ int getZ()	Return z
+ void draw(GraphicsContext gc)	Draw object on GraphicsContext of gameScreen
+ boolean isVisible()	Is object visible

5.8 **Class LastUsedCard** extends Entity

For showing last used card text on screen.

5.8.1 Fields

Name	Description
- final int X	The x-coordinates of the screen
- final int Y	The y-coordinates of the screen

5.8.2 Constructor

Name	Description

+ LastUsedCard()	Initialize last use card and set it to visible

5.8.3 Methods

Name	Description
+ void draw(GraphicsContext gc)	Show last used card text on screen
+ String replaceLineWithSpace(String str)	Return one line string from multi line string

5.9 **Class Phase** extends Entity




5.9.1 Fields

Name	Description
- int x	The x-coordinates of the screen
- int y	The y-coordinates of the screen

5.9.2 Constructor

Name	Description
+ Phase()	Initialize phase and set it to visible

5.9.3 Methods

Name	Description
+ void draw(GraphicsContext gc)	Show image of current phase; Move, attack, draw cards, or bot turn.

5.10 **Class Player** extends Controller

Player is controller that user can control.

5.10.1 Constructor

Name	Description
+ Player(int health, int money, Deck deck, Direction playingSide)	Initialize player as the controller

5.10.2 Methods

Name	Description
+ int getMaxCardCostCanDraw()	Return max card cost that can draw card (Turn + 2)

5.11 **Class Turn** extends Entity

5.11.1 Constructor

Name	Description
+ Turn()	Initialize turn

5.11.2 Methods

Name	Description
+ void draw(GraphicsContext gc)	Draw turn text in the upper part of the screen

6. Package exception

6.1 **Class ImageNotFoundException** extends Exception

6.1.1 Constructor

Name	Description
+ ImageNotFoundException()	Return specific message when RenderableHolder don’t found image

6.2 **Class AudioNotFoundException** extends Exception

6.2.1 Constructor

Name	Description
+ AudioNotFoundException()	Return specific message when SoundHolder don’t found sound

6.3 Class **WrongDeckDataException** extends Exception

6.3.1 Constructor

Name	Description
+ WrongDeckDataException()	Return specific message importDeck() or getTrick() in Deck class found wrong deck data

6.4 Class **WrongTrickActivateTypeException** extends Exception

6.4.1 Constructor

Name	Description
+ WrongTrickActivateTypeException()	Return specific message when constructor of Trick receives wrong trick parameter

7. Package gui

7.1 Class **Board** extends GridPane

7.1.1 Fields

Name	Description
+ final int LAYOUT_X	X position of board
+ final int LAYOUT_Y	Y position of board
+ final int H_GAP	Hgap
+ final int V_GAP	Vgap
+ final int NUMBER_OF_ROW	Number of row in the board
+ final int NUMBER_OF_COLUMN	Number of column in the board
- ObservableList<ObservableList<Cell>> boardCells	List of the board cells contain every cell in board

7.1.2 Constructor

Name	Description
+ Board()	set board with NUMBER_OF_ROW and NUMBER_OF_COLUMN

7.1.3 Methods

Name	Description
+ void allCardAttack()	Make all cards on the board attack
+ void attackCard(int row, int column, int attackDamage)	Attack the card on specific row and column
+ boolean canPlayTrickCard(TrickCard trickCard)	If TrickCard can uses according to trick parameter then return true
+ ObservableList<Cell> getBoardAtColumn(int column)	Return a board in specific column
+ ObservableList<Cell> getBoardAtRow(int row)	Return a board in specific row
+ ArrayList<FighterCard> getEnemy(Direction playingSide)	Return list of all enemy in the board
+ ArrayList<FighterCard> getFriendly(Direction playingSide)	Return list of the our troop in the board
+ int getNearestEnemyColumn(int row, Direction playingSide)	Return nearest column that have enemy
+ int getNearestEnemyRow(Direction playingSide, ArrayList<Integer> excludedRow)	Return row that have most nearest enemy
+ FighterCard getRandomEnemy(Direction playingSide)	Return random enemy on the board, if there is no enemy on the board return null
+ FighterCard getRandomFriendly(Direction playingSide)	Return random our troop on the board, if there is not our troop on the board return null
+ boolean haveEnemy(Direction playingSide)	Return true if there are enemy on the board
+ boolean haveFriendly(Direction playingSide)	Return true if there are our troop on the board
+ void highlight(int row, int column)	Highlight the specific cell on the board
+ void highlightAllCell()	Highlight all cells on the board
+ void highlightCellCanPlay(CardInHandPane selectedCardPane)	Highlight all cells in front of their side that is empty
+ void hightlightEnemy(Direction playingSide)	Highlight the cell that enemy troop placed
+ void hightlightFriendly(Direction playingSide)	Highlight the cell that our troop placed
+ boolean isEmpty(int row, int column)	Return true if specific row and column cell on the board is empty

+ boolean isEnemy(int row, int column, Direction playingSide)	Return true if specific row and column cell on the board has enemy
+ boolean isFriendly(int row, int column, Direction playingSide)	Return true if specific row and column cell on the board has our troop
+ boolean isOutOfBoard(int row, int column)	Return true if specific row and column cell is out of board
+ void moveAllCard(Direction playingSideMoveFirst)	Move all the cards. playingSideMoveFirst make card belong to this playing side move first
+ void moveLeftPlayingSideCard(int r)	Move card left playing side, move from right to left
+ void moveRightPlayingSideCard(int r)	Move card right playing side, from left to right
+ void removeCardOnBoard(int row, int column)	Remove the card on the cell on the specific row and column
+ void removeDeadCards()	Remove the card that has health less than or equal to 0
+ void setCard(CardPane cardPane, int row, int column)	Set CardPane on the specific cell
+ void unHighlightAllCells()	Unhighlight all cells on the board
+ void update()	Update GUI of card if card ability change

7.2 **Class CardInHandPane** extends CardPane

CardInHandPane is GridPane for card in hand. It shows image of card, ability of card and contain card data.

7.2.1 Fields

Name	Description
- CardInHandPane cardPane	This class
- final int cardWidth	Card width
- final int cardHeight	Card height
- final int insets	Card insets

7.2.2 Constructor

Name	Description
+ CardInHandPane(Card card)	Initialize card in the hand according to cardPane, cardWidth, cardHeight, and insets parameter

7.2.3 Methods

Name	Description
+ void addCardAbility(Image image, Card card, int value, int defultValue, int x, int y, int columnSpan)	Add card ability image to card GridPane
+ void addCardImage(Image image)	Add card image to card GridPane
+ boolean canSelectCard()	If player can select this card then return true
+ void highlight()	Highlight the card in hand
+ boolean isCardTooExpensive()	Return true if cost of the card more than money
+ boolean isSelect()	Return true if card is selected
+ void setCardAbility(Card card)	Set card cost, attackDamage, attackRange, health , speed image to card GridPane
+ void setMouseEvent()	Set mouse event when mouse clicked, moved or exited
+ void unhighlight()	Unhighlight the card in hand

7.3 Class **CardOnBoardPane** extends CardPane

CardOnBoardPane is GridPane for card on board. It shows image of card, ability of card except cost and contain card data.

7.3.1 Fields

Name	Description
- CardOnBoardPane cardPane	Card in the hand
- final int cardWidth	Card width
- final int cardHeight	Card height
- final int insets	Card insets

7.3.2 Constructor

Name	Description
+ CardOnBoardPane(Card card)	Initialize card on the board according to cardPane, cardWidth, cardHeight, and insets parameter

7.3.3 Methods

Name	Description
+ void addCardAbility(Image image, Card card, int value, int defultValue, int x, int y)	Add card ability image to card GridPane
+ void addCardImage(Image image)	Add card image to card GridPane
+ void attack()	The card on the board will attack its attack range and reduce its opponent's health
+ FighterCard getCard()	Return the card on the field and convert to FighterCard
+ void move()	Move the card according to its speed if next cell is empty and not out of board
+ void playAttackSound()	Play random attack sound
+ void playMoveSound()	Play random move sound
+ void setCardAbility(Card card)	Set card attackDamage, attackRange, health , speed image to card GridPane

7.4 **Class CardPane** extends GridPane

CardPane is GridPane. CardInHandPane and CardOnBoardPane are CardPane.

7.4.1 Fields

Name	Description
# Card card	Card in this cardPane
# Tooltip tooltip	Contain card discription. It will be show when mouse moved on this card.

7.4.2 Methods

Name	Description
+ void addCardImage(Image image)	Add troop image to CardPane with some rowSpan and columnSpan
+ void setCardAbility(Card card)	Add every card ability image and value
+ void setMouseEvent()	Make tooltip show when mouse moved, and hide when mouse exited
+ void setToolTip()	Set font and text that show card type and description to ToolTip
+ getter and setter	

7.5 **Class Cell** extends StackPane

7.5.1 Fields

Name	Description
+ final int CARD_WIDTH	Card width
+ final int CARD_HEIGHT	Card height
- CardOnBoardPane cardOnBoardPane	Card on the board
- Color backgroundColor	Background color
- int row	row
- int column	column
- boolean isEmpty	Parameter that indicate the cell is empty
- boolean isHighlight	Parameter that indicate the cell is highlighted

7.5.2 Constructor

Name	Description
+ Cell(int row, int column)	Initialize cell in the specific location

7.5.3 Methods

Name	Description
+ FighterCard getCard()	Return Card in this cell
+ void highlight()	Highlight the cell
+ boolean isEmpty()	Return true if that cell is empty
+ void removeCard()	Remove the card on the cell
+ void setCard(CardPane cardPane)	Change GUI from cardOnHand to cardOnBoard and set to the cell
+ void unhighlight()	Unhighlight the cell
+ getter and setter	

7.6 **Class HandPane** extends Vbox

There are 2 HandPane. One is left side of screen for storing left playing side cards, The others is right side.

7.6.1 Fields

Name	Description
- ObservableList<CardInHandPane> cardsList	List of the card in the hand

7.6.2 Constructor

Name	Description
+ HandPane()	Initialize HandPane with the card in the hand and set the background and set space between them

7.6.3 Methods

Name	Description
+ void add(String deckName, Card card)	Add the card to hand pane
+ Card getCard(int index)	Return card that refer to index of the card
+ int getSize()	Return cardsList size
+ void highlight()	Highlight the HandPane in our turns
+ int indexOf(Card card)	Return the index of the card in the cardList that have this card
+ int indexOf(CardInHandPane cardPane)	Return the index of the cardPane in cardList
+ void remove(int index)	Remove the card in the card list
+ void setSelectedCard(CardInHandPane selectedCardPane)	Make game know that player select the card. High
+ void unHighlightAllCardInHandPane()	Unhighlight all the cards in the handPane
+ void unHighlight()	Unhighlight the handPane

8. Package logic

8.1 Enum Direction

Direction contain constants LEFT, RIGHT.

8.1.1 Fields

Name	Description
+ final Direction LEFT	LEFT Direction
+ final Direction RIGHT	RIGHT Direction

8.2 Class GameController

8.2.1 Fields

Name	Description
+ Thread threadDrawCard	Draw card thread in drawCard() method
+ Thread threadBotPlay	Bot play thread in play() method
+ Thread threadCardMove	Card move action thread in move() method
+ Thread threadMoveAllCard	Move all cards action thread in moveAllCard() method
+ Thread threadStartAttackCard	Card attack action thread in startAttackCard() method
+ Thread threadAttackAllCard	Card attack action thread in attackAllCard() method
+ Thread threadAttack	Card attack action thread in attack() method
+ final int DELAY_DRAW_CARD	Draw card delay = 500ms
+ final int DELAY_BOT_PLAY	Bot turn delay = 1200ms
+ final int DELAY_CARD_MOVE	Card move action delay = 200ms
+ final int DELAY_ATTACK	Card attack action delay = 500ms
+ final int SCREEN_WIDTH	Screen width = 1280
+ final int SCREEN_HEIGHT	Screen height = 720
+ Stage primaryStage	Primary stage
+ GameScreen gameScreen	Game screen
+ final ArrayList<Deck> DECKS	List of deck
+ Board board	Board
+ int turn	Current turn
+ int moneyFromTurn	Money each controller gets each turn
+ boolean isPhaseOneEnd	Phase one end or not (first controller already play or not)
+ Direction currentPlayingSide	Currnet playing side
+ Controller leftSideController	Left side controller

+ Controller rightSideController	Right side controller
+ Direction winner	Winner side
+ boolean isGameEnd	Game end or not
+ CardInHandPane selectedCardPane	Selected card in handPane for play or place on board
+ Card lastUsedCard	Last used card
+ FighterCard targetCard	Target card of trick
+ String gameMode	Game mode
+ Deck leftSideDeck	Left side deck
+ Deck rightSideDeck	Right side deck
+ String difficultyLeft	Left side bot difficulty
+ String difficultyRight	Right side bot difficulty

8.2.2 Methods

Name	Description
+ void initializeGameBvB()	Initialize Game in case of Bot vs Bot
+ void initializeGamePvB()	Initialize Game in case of Player vs Bot
+ void initializeGamePvP()	Initialize Game in case of Player vs Player
+ boolean isBotSide(Direction direction)	Return true if there is bot on the side
+ void playGame()	Initialize game according to game mode
+ void startAttackCard()	Wait till all card move finish first then attack
+ void startFirstTurn()	Each side draw 4 cards and start to play the turn
+ void startGame()	Initialize game screen, set turn 0, set winner null, and random side to start first
+ void startMoveCard()	Move all the cards on the board
+ void startNextPhase()	Start next phase. Between second controller playing phase or move card phase.
+ void startTurn()	Increse turn, and increse moneyFromTurn (not exceed 20), increase player money, and draw 2 cards both sides
+ void switchPlayingSide()	Switch playing side to make the other controller play

9. Package screen

9.1 Class GameScreen

9.1.1 Fields

Name	Description
- Pane root	Screen pane
- BorderPane borderPane	BorderPane
- Canvas canvas	Screen canvas
- GraphicsContext gc	Graphics context
- Image background	Background image
- Turn turn	Turn
- Phase phase	Phase
- LastUsedCard lastUsedCard	Last used card
- Button nextPhaseButton	Next phase button
- HandPane leftCardsInHand	Left hand pane
- HandPane rightCardsInHand	Right hand pane
- ObservableList<Media> mediaList	Media list
- MediaPlayer mediaplayer	Media player

9.1.2 Constructor

Name	Description
+ GameScreen()	Initialize Gamescreen, screen settings, set screen margin, and game settings

9.1.3 Methods

Name	Description
+ boolean canClickStartNextPhaseButton()	Return true if the player can click startNextPhaseButton
+ void getSoundList()	Add all sounds to play and shuffle
+ Button getStartNextPhaseButton()	Setup and return next phase button
+ void highlightHandPane(Direction direction)	Highlight the specific direction’s hand pane
+ void paintComponent()	Paint background and draw entity
+ void playSound()	Play the sound in the media list
+ void setBackground()	Set random background
+ void stopSound()	Stop sound in game screen
+ void unHighlightHandPane()	Unhighlight the specific direction’s hand pane
+ void updateStartNextPhaseButton()	Update next phase button visibility
+ getter and setter	

9.2 Class EndGame extends StackPane

9.2.1 Constructor

Name	Description
+ EndGame()	Initialize end game, clear entities, clear thread, and stop sound

9.2.2 Methods

Name	Description
+ void clearEntity()	Clear all entities (make everything invisible)
+ void clearThread()	Set all thread to null. To prevent game still running after game end.
+ Button getGoBackButton()	Setup and return go back button
+ VBox getVBox()	Setup and return vBox contain winner text and goBack button
+ void stopSound()	Stop gameScreen sound

9.3 Class HowToPlayScreen

9.3.1 Fields

Name	Description
- Pane root	Pane
- BorderPane borderPane	Border pane
- Button goBackButton	Go back button
- ImageView background	Background image

9.3.2 Constructor

Name	Description
+ HowToPlayScreen()	Initialize HowToPlayScreen

9.3.3 Methods

Name	Description
+ Button getGoBackButton()	Initialize goBackButton. Make if user clicked on the goBackButton, then go back to SelectGameModeScreen. Return goBackButton
+ Button getScrollPane()	Initialize ScrollPane. Set contect to HowToPlayDetail image. Return ScrollPane.

9.4 Class **SelectGameModeButton** extends GridPane

9.4.1 Fields

Name	Description
- Button exitButton	Exit button
- Button pvpButton	PVP mode button
- Button pvbButton	PVB mode button
- Button bvbButton	BVB mode button
- Button howToPlay	How to play Button

9.4.2 Constructor

Name	Description
+ SelectGameModeButton()	Initialize all the buttons

9.4.3 Methods

Name	Description
+ Button setUpButton(String name)	Set up button settings and mouse event then return button

9.5 Class SelectGameModeScreen

9.5.1 Fields

Name	Description
- StackPane root	Root of scene
- Scene scene	Scene
- ImageView image	Background image

9.5.2 Constructor

Name	Description
+ SelectGameModeScreen()	Initialize select game mode screen that has image and image stack on the StackPane(root)

9.5.3 Methods

Name	Description
+ Button getExitButton()	Setup exit button and make if the mouse click the exit button then exit the platform then return exit button

9.6 Class SettingButton extends GridPane

9.6.1 Fields

Name	Description
- MenuButton LeftSideDeck	Left side deck MenuButton
- MenuButton RightSideDeck	Right side deck MenuButton
- MenuButton LeftSideDifficulty	Left side difficulty MenuButton
- MenuButton RightSideDifficulty	Right side difficulty MenuButton

9.6.2 Constructor

Name	Description
+ SettingButton()	Add LeftSideDeck, RightSideDeck menu button. And add bot difficulty menu button if there is a bot in any side in this game mode.

9.6.3 Methods

Name	Description
+ MenuButton getSelectDeckButton(Direction direction)	Initialize select deck button then return selectDeckButton
+ MenuButton getSelectDifficultyButton(Direction direction)	Initialize selectbot dificulty button then return selectDifficultyButton

9.7 Class SettingScreen

9.7.1 Fields

Name	Description
- Pane root	Pane
- BorderPane borderPane	Border pane
- Button goBackButton	Go back button
- Button playButton	Play button
- ImageView image	Image
- Text warningText	Warning text

9.7.2 Constructor

Name	Description
+ SettingScreen()	Initialize setting screen

9.7.3 Methods

Name	Description
+ Button getGoBackButton()	Initialize goBackButton. Make if user clicked on the goBackButton, then go back to SelectGameModeScreen. Return goBackButton
+ Button getPlayButton()	Initialize playButton. Start the game when the playButton is clicked but in the condition that everything is filled. If not the warning text will appear. Return playButton

10. Package sharedObject

10.1 Class FontHolder

Font “EvilEmpire” for this game

10.1.1 Fields

Name	Description
- final FontHolder instance	Instance of this class
+ Font font12	Font 12
+ Font font15	Font 15
+ Font font18	Font 18
+ Font font24	Font 24
+ Font font28	Font 28
+ Font font32	Font 32
+ Font font36	Font 36
+ Font font48	Font 48
+ Font font64	Font 64

10.1.2 Constructor

Name	Description
+ FontHolder()	Initialize fonts and its size by call loadFont()

10.1.3 Methods

Name	Description
+ Font loadFont(String name, String fontType, double size)	Load font according to font’s name, font type, and size

10.2 Class RanderableHolder

10.2.1 Fields

Name	Description
- final RenderableHolder instance	Instance of this class
+ Image icon	Icon of this game
+ Image backgroundSelectGameMode	SelectGameMode background image
+ Image backgroundSelectDeckPvB	SelectDeckPvB background image
+ Image backgroundSelectDeckPvP	SelectDeckPvP background image
+ Image backgroundSelectDeckBvB	SelectDeckBvB background image
+ Image backgroundGameScreen	GameScreen background image
+ Image backgroundGameScreen1	GameScreen1 background image
+ Image backgroundGameScreen2	GameScreen2 background image
+ Image backgroundGameScreen3	GameScreen3 background image
+ Image backgroundGameScreen4	GameScreen4 background image
+ Image backgroundGameScreen5	GameScreen5 background image
+ Image cost	Cost image
+ Image attackDamage	attackDamage image
+ Image attackRange	attackRange image
+ Image health	health image
+ Image speed	speed image
+ Image nextPhase	nextPhase button image
+ Image phaseDrawCard	phaseDrawCard image
+ Image phaseBot	phaseBot image
+ Image phaseMove	phaseMove image
+ Image phaseAttack	phaseAttack image
+ Image cardAttack	cardAttack status image
+ Image cardDefense	cardDefense status image
+ Image cardDead	cardDead status image
+ Image trick	Card be tricked status image
+ Image ChangeCardAbility	ChangeCardAbility trick image
+ Image ChangeControllerHealth	ChangeControllerHealth trick image
+ Image angelFighterL	Left side angel fighter image
+ Image angelFighterR	Right side angel fighter image
+ Image angelMagicianL	Left side angel magician image
+ Image angelMagicianR	Right side angel magician image
+ Image devilFighterL	Left side devil fighter image
+ Image devilFighterR	Right side devil fighter image
+ Image devilMagicianL	Left side devil magician image
+ Image devilMagicianR	Right side devil magician image
- List<IRenderable> entities	List of entities
- Comparator<IRenderable> comparator	Comparator for sort entities

10.2.2 Constructor

Name	Description
+ RenderableHolder()	Initialize list entities and comparator

10.2.3 Methods

Name	Description
+ RenderableHolder getInstance()	Return instance
+ Image loadImage(String fileName)	Return image by file name
+ void loadResource()	Load every pictures
+ void add(IRenderable entity)	Add entity and sort
+ void update()	Remove entity if it is invisible
+ getter and setter	

10.3 Class SoundHolder

10.3.1Fields

Name	Description
- final SoundHolder instance	Instance of this class
+ Media gameScreen1	Game screen 1 music
+ Media gameScreen2	Game screen 2 music
+ Media gameScreen3	Game screen 3 music
+ Media gameScreen4	Game screen 4 music
+ Media gameScreen5	Game screen 5 music
+ Media gameScreen6	Game screen 6 music
+ Media gameScreen7	Game screen 7 music
+ Media gameScreen8	Game screen 8 music
+ Media gameScreen9	Game screen 9 music
+ Media gameScreen10	Game screen 10 music
+ AudioClip endGameScreen	endGameScreen sound effect
+ AudioClip mainScreen	mainScreen music
+ AudioClip cannotSelect	Cannot select sound effect
+ AudioClip selectCard	Select card sound effect
+ AudioClip placeCard1	Place card 1 sound effect
+ AudioClip placeCard2	Place card 2 sound effect
+ AudioClip placeCard3	Place card 3 sound effect
+ AudioClip click	Click audio sound effect
+ AudioClip drawCard	Draw card sound effect
+ AudioClip move1	Move 1 sound effect
+ AudioClip move2	Move 2 sound effect
+ AudioClip move3	Move 3 sound effect
+ AudioClip move4	Move 4 sound effect

+ AudioClip move5	Move 5 sound effect
+ AudioClip move6	Move 6 sound effect
+ AudioClip move7	Move 7 sound effect
+ AudioClip attack1	Attack 1 sound effect
+ AudioClip attack2	Attack 2 sound effect
+ AudioClip attack3	Attack 3 sound effect
+ AudioClip attack4	Attack 4 sound effect
+ AudioClip attackController	AttackController sound effect
+ AudioClip trick	Trick sound effect

10.3.2 Methods

Name	Description
+ SoundHolder getInstance()	Return instance
+ loadResource()	Load media and load sound
+ AudioClip loadSound (String fileName)	Return sound according to fileName
+ Media loadMedia(String fileName)	Return media according to fileName

11. Package Trick

11.1 Class **ChangeCardAbility** extends Trick

11.1.1 Fields

Name	Description
- char activateType	Activate type; A = Random friendly, B = Random enemy, C = Select friendly, D = select enemy
- int attackDamage	Attack damage to add
- int attackRange	Attack range to add
- int health	Health to add
- int speed	Speed to add

11.1.2 Constructor

Name	Description
+ ChangeCardAbility(String trickparameter)	Initialize this trick according to trickparameter

11.1.3 Methods

Name	Description
+ void activate()	Activate this trick

+ void Update(FighterCard card)	Update target card ability
+ String getDescription()	Get trick description

11.2 Class **ChangeControllerHealth** extends Trick

11.2.1 Fields

Name	Description
- char activateType	Activate type; T = controller who activate side, E = enemy side, S = both two side
- int health	Controller health to add

11.2.2 Constructor

Name	Description
+ ChangeControllerHealth(String trickparameter)	Initialize activateType, health and image according to trickparameter

11.2.3 Methods

Name	Description
+ void activate()	Activate trick
+ String getDescription()	Return trick description

11.3 Class **DestroyCard** extends Trick

11.3.1 Fields

Name	Description
- char activateType	Activation type; B = Random enemy, D = select enemy

11.3.2 Constructor

Name	Description
+ DestroyCard(String trickparameter)	Initialize activateType and image

11.3.3 Methods

Name	Description
+ void activate()	Activate trick
+ void Update(FighterCard card)	Update target card health
+ String getDescription()	Return trick description

11.4 **Class Draw** extends Trick

11.4.1 Fields

Name	Description
- char activateType	Activation type
- int number	Number of card to draw

11.4.2 Constructor

Name	Description
+ Draw(String trickparameter)	Initialize activateType and Number of card to draw according to trickparameter

11.4.3 Methods

Name	Description
+ void activate()	Activate trick
+ String getDescription()	Return trick description

11.5 **Class Trick** implements Cloneable

11.5.1 Fields

Name	Description
# String description	Trick activation type
# Direction playingSide	Playing side
# ArrayList<String> trickParameter	List of trick parameter
# Image image	Image of trick

11.5.2 Constructor

Name	Description
+ Trick(String trickparameter)	Initialize trick parameter

11.5.3 Methods

Name	Description
+ void activate()	Activate the trick
+ Object clone()	Return clone of this trick
+ FighterCard getBotSelectTargetCard()	Return target card of bot when bot use trick
+ String getDescription()	Get description of the trick
+ char getFirstParameter()	Return first trick parameter
+ getter and setter	