**DOKUZ EYLÜL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME 2210**

**Object Oriented Analysis and Design**

**CARDMAYHEM**

**by**

**Oğuzhan YARDIMCI - 2015510066**

**Fatma KAHRAMAN - 2015510122**

**Ahmet Talat GEDİKLİ - 2016510028**

**Gaye EROL - 2016510106**

**CHAPTER ONE**

**INTRODUCTION**

* 1. **Purpose**

The aim of the project is to develop a two-player card game. Each player chooses one of 4 characters. These characters are Axe, Blood, Light and Stone. If you set your opponent’s health to zero or below player wins the game.

* 1. **Scope**

In our project, every character has his/her own specialty. Each character has ten unique cards which are designed to improve the character’s perk and weapon. Common cards are also designed to add bonus effects for specific characters.

* 1. **Definitions, Acronyms, and Abbreviations**

CardMayhem is a turn-based card game in which you may choose amongst on four characters with four almost-unique decks for each of them. These characters are Axe, Blood, Light and Stone who focus on hitting high melee damage, leeching opponent’s health, dealing magical damage which ignores armor and bearing huge amount of armor which reduces the melee damage, respectively.

Two players start the game after choosing one of the characters. The aim is setting the opponent’s remaining health to 0. To reach it, every character will try his/her own specialty and will have ten unique cards which are designed the improve the character’s perk and weapon. Players have one action point for each turn, action point is spent whether hitting with your weapon or playing a card from your hand. Action point per turn will be increased during the game as the turns pass.

* 1. **References**

We inspired from the Blizzard’s game Hearthstone.

* 1. **Overview**

There are totally 60 cards in the game and they are separated as common cards and unique cards. Common cards are 20 in total and they can be used by any character. These have 8 different effects: Restoring health, adding extra damage to weapon, gaining armor and dealing direct damage cards, Each effect has 4 cards in the common cards pool and other 4 cards which are also called epic cards have 4 different effects: Shuffling your hand into your deck and drawing that many cards, summoning two cards from the graveyard where all used cards are stored, gaining one action point for one turn and the last effect is discarding your whole hand to fully restore your health. Beside those, there are 10 unique cards for each character. One of those 10 unique cards for each character is called Legendary and 4 of those unique cards for each character are called Rare. Legendaries consume all the action points on the turn that they are played.

# **CHAPTER TWO**

**OVERALL DESCRIPTION**

1. **Product Perspective**.

We planned to offer users such a game with dynamic mechanics that we have thirty-card-sized deck which we draw cards at the beginning of every turn, manage those cards so that we can hold them in our hand or play as soon as possible for obtaining maximum value.

* 1. **Product Functions**

In BattleField class has showHistory( ) function for the player can see all played cards. ICharacter is a interface class and ICharacter has play( ), getName( ), hit( ), generateDeck( ) functions. Hand class has pick( ) and draw( ) functions. AxeClass, BloodClass, LightClass, StoneClass will use generateDeck( ) function for create deck to each characters. AxeDeck, BloodDeck, LightDeck, StoneDeck classes has their unique functions

* 1. **User Characteristics**

Characters are:

### Axe:

This character specializes dealing huge melee damage with the aid of both common pool cards and his unique cards.His statistics are:

Health:⭐⭐⭐⭐

Weapon:⭐⭐⭐⭐⭐

Armor:⭐⭐

Magic: None

### Light:

This character specializes dealing magical damage which ignores the armor and directly reduces opponent’s health.

Health:⭐⭐⭐

Weapon:⭐⭐

Armor:⭐

Magic: ⭐⭐⭐⭐⭐

### Blood:

Blood has a unique effect in game. When she deals damage anyhow, the amount of damage dealt also heals her that much.

Health:⭐⭐

Weapon:⭐⭐

Armor:⭐

Magic: ⭐⭐⭐⭐

### Stone:

This tank character bears shield to gain armor throughout the game which reduces the melee damage taken.

Health:⭐⭐⭐

Weapon:⭐⭐⭐

Armor:⭐⭐⭐⭐⭐

Magic: ⭐

* 1. **Constraints**

Some cards can be used in the game after a certain turn, each character has a single deck option that each player can use in each game. Constraints are the cards that can only be played after some certain turns, specific decks for each character and excluding some exceptions, having one and after a while having two action points.

* 1. **Assumptions and Dependencies**

We tried to design the game so that after a certain point in the game, the course of the game depends on a very small amount of luck, but it became more dependent on the player's strategy. The game has become as dynamic as possible because almost every card gives bonuses to 4 characters in different ways. To make the game this way, every game that will be played, will always be slightly different from other games that have been played, and we assume that provide a new game experience each time.

**CHAPTER THREE**

# **SPECIFIC REQUIREMENTS**

1. **External Interfaces**

ICharacter interface is implemented to BloodClass, StoneClass, LightClass and AxeClass in our game.

* 1. **Functions**

BattleField is our management class. In BattleField class has showHistory( ) function for the player can see all played cards. ICharacter is a interface class and ICharacter has play( ), getName( ), hit( ), generateDeck( ) functions. Hand class has pick( ) and draw( ) functions for the players can pick and draw cards from deck.

AxeClass, BloodClass, LightClass, StoneClass will use generateDeck( ) function for create deck to each characters. Also, AxeDeck, BloodDeck, LightDeck, StoneDeck classes has their unique functions.

AxeDeck class has generateSharpenedEdge( ), generateBlacksmithsFavor( ), generateHardenerSkin( ), generateBloodFrenzy( ), generateUnarmTheWeak( ), generateTakesTheHearth( ), generateGoForTheHead( ), generateColdBlood( ), generateParry( ), generateCrumble( ) unique functions for use his specific features and apply to opponent.

BloodDeck class has generateBloodBath( ), generateRedAura( ), generateWildPoison( ), generateDodge( ), generateAmuletsGeis( ), generateEmptyVeins( ), generateDarkTerror( ), generatedDevilsAce( ), generateParalyze( ), generateDelusion( ) unique functions for use her specific features and apply to opponent.

LightDeck class has generateColdAura( ), generateEnlightenment( ), generateFirebolt( ), generateMagesWill( ), generateArchmagesFavor( ), generateCryonicState( ), generateShiningOfSkies( ), generateRapidMagic( ), generateGemstuddedStaff( ), generateTwistingDarkness( ) unique functions for use her specific features and apply to opponent.

StoneDeck class has generateTankUp( ), generateKnightsRelic( ), generateBlessing( ), generateCover( ), generateAntiMagicAmulet( ), generateSpikedShield( ), generatePurification( ), generateKingSlayer( ), generateHolyWater( ), generateShildBash( ) unique functions for use his specific features and apply to opponent.

* 1. **Performance Requirements**

The game needs to be completed successfully. It doesn’t encounter any bugs until the game is over.

* 1. **Logical Database Requirements**

In our project, database must be dynamic and useful.

* 1. **Design Constraints**

Characters can only access the cards in their deck. Each character's deck and cards are isolated from each other.

* 1. **Software System Quality Attributes**

According to our project, software system quality attributes are readability, extensibility, testability, maintainability and reliability.

* 1. **Object Oriented Models**

In the game, Game cards properties are going to be held in Card class in which cards will have their own shapes.Cards in the player’s deck will use Stack. Common Card Pool is going to be created in the class Deck. This class is going to be abstract and will be extended by BloodClass, StoneClass, LightClass and AxeClass classes. Draw class is going to be abstract, too. Functions in the individual characters’ decks are override as the game offers some bonuses to specific cards for the characters.

* + 1. **Analysis Class Model (Static Model)**

AxeClass, LightClass, BloodClass, StoneClass, Card, Weapon, Hand classes are created appropriate to the Static Model.

* + 1. **Analysis Collaborations (Dynamic Model)**

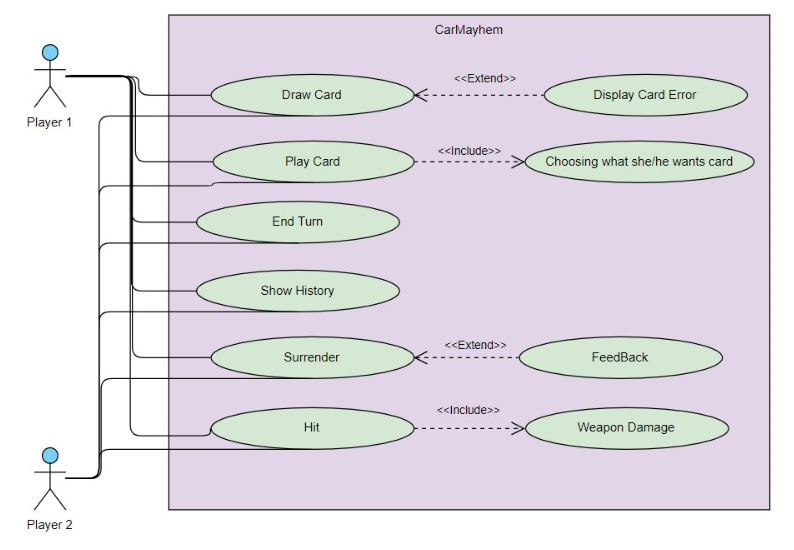
Deck, AxeDeck, LightDeck, BloodDeck, StoneDeck, Battlefield classes can be changed, developed or updated.

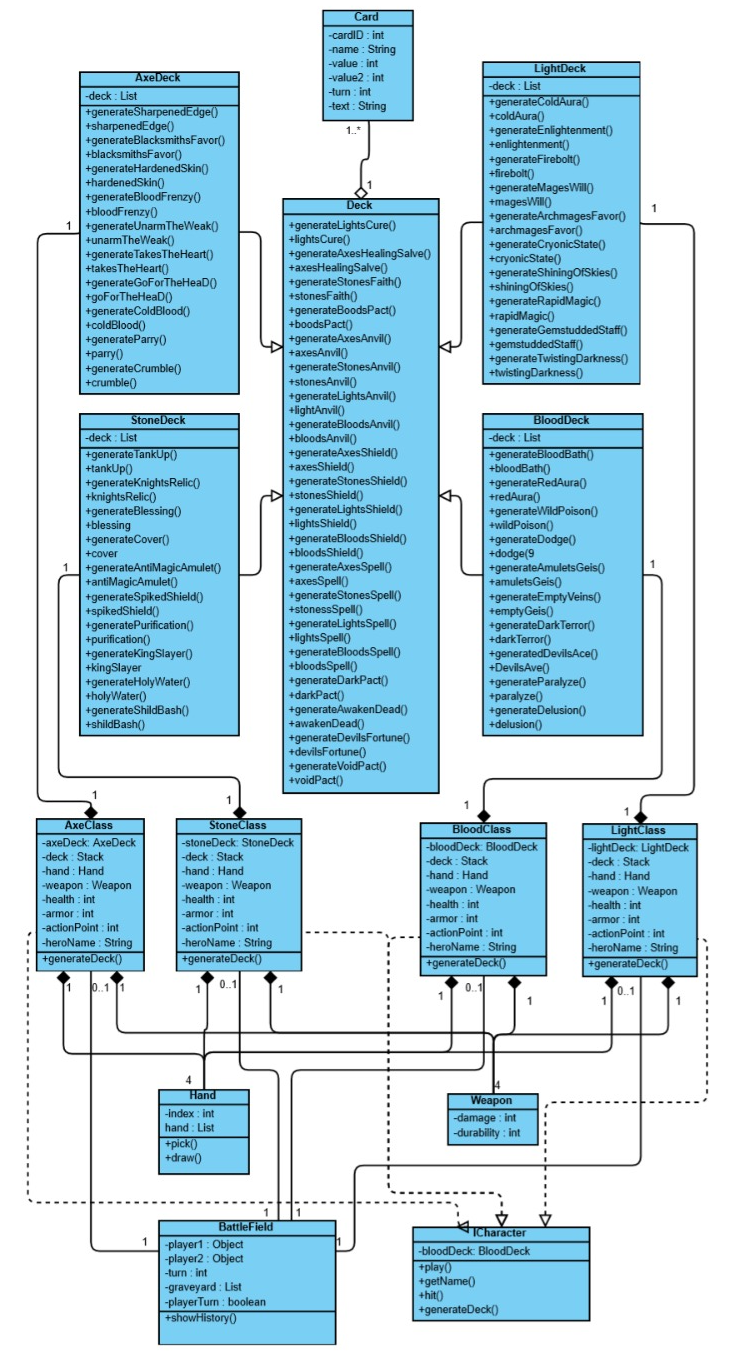
**CHAPTER FOUR**

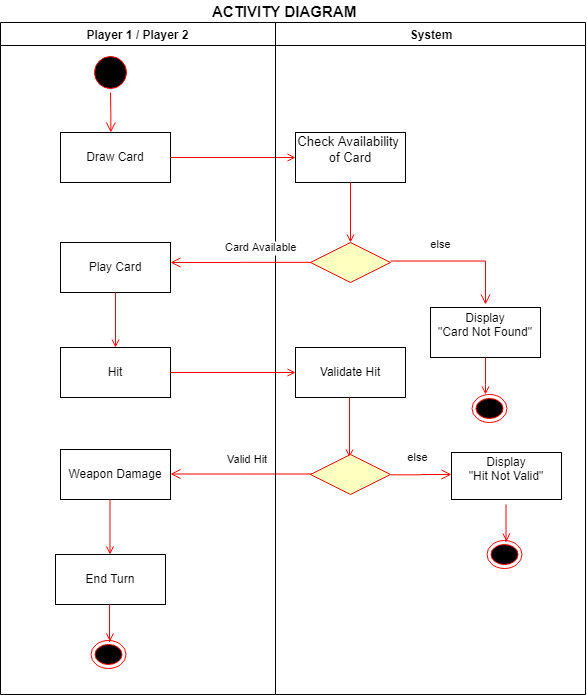
**UML DIAGRAMS**

Use Diagram gives the opportunity to view the project as a user. Creating the inputs taken by the users and states the situations that can be encountered as the user does something unexpected from the code. Thus, users’ demands are indicated to the programmer. Repeating the code and designing is avoided with those diagrams. Logical errors are reduced to minimum and communication among programmers is provided.

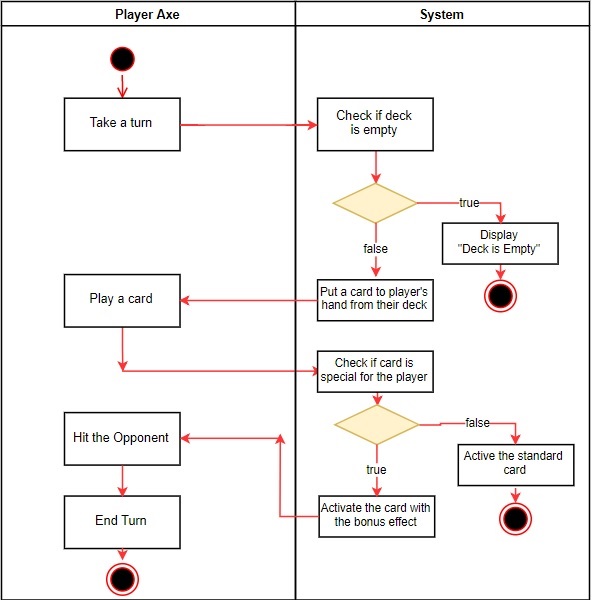
Class Diagram is the collaboration of the all classes and their associates to have the view of them altogether. Thus, all the connections among classes are stated clearly.



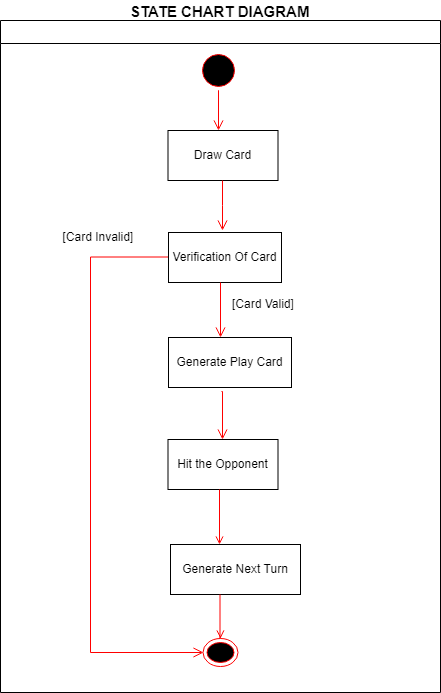




This activity diagram shows a single turn taken by the Player Axe drawing a card and playing a card and during this turn the system controls the availability of the card of the deck and checks if there is a special bonus for the played card for the Axe character and player ends the turn.

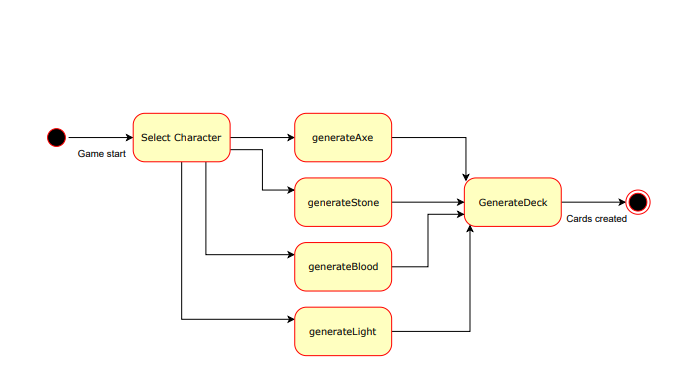


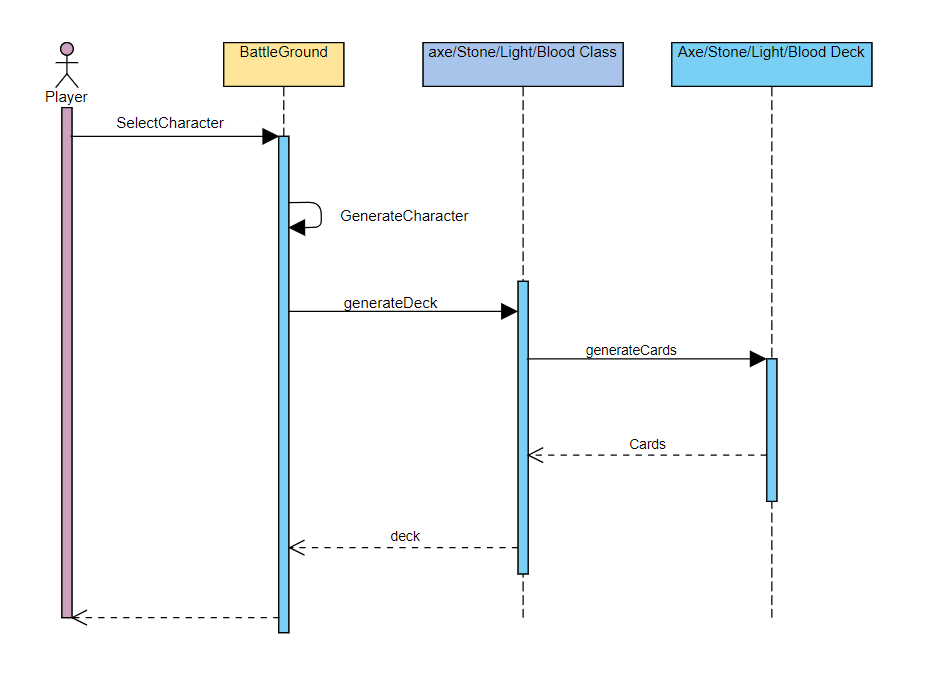
After draw card, player can hit the opponent when play card and then generating next turn.



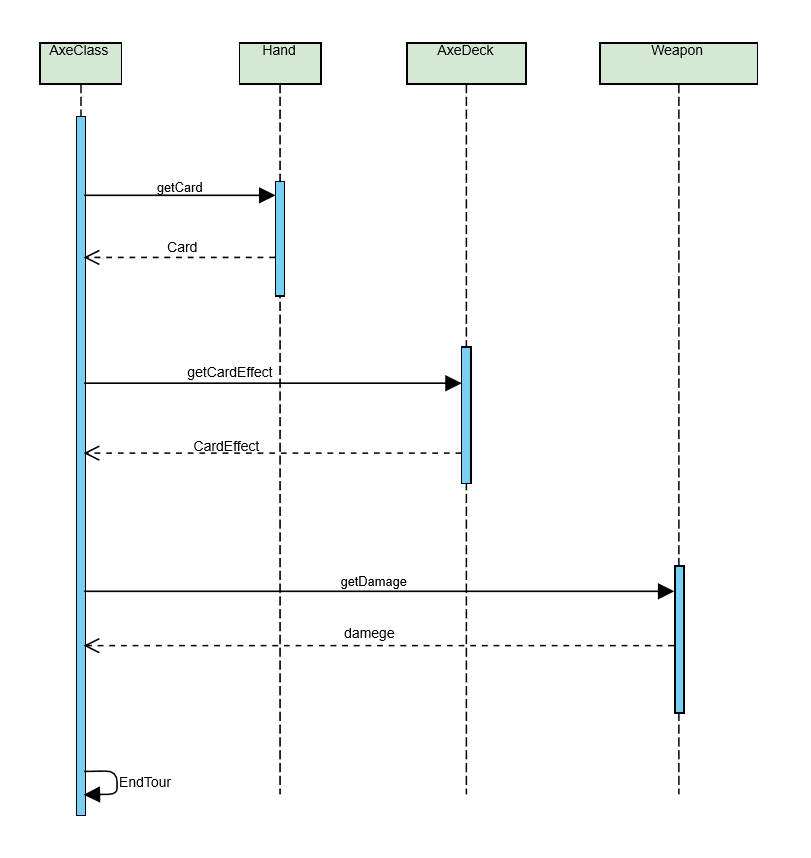
When the player wants to start the game, he/she select one character. Then a deck of selected character is created. (State 2 and sequence 1)

State 2

Sequence 1



Drawing cards by turn, playing the cards and hitting with the weapon.



**CHAPTER FIVE**

**IMPLEMENTATION**

BattleField is our management class. In BattleField class has showHistory( ) function for the player can see all played cards. ICharacter is a interface class. AxeClass, BloodClass, LightClass, StoneClass has generateDeck( ) function for creating their cards.

Some cards in each deck including common cards deck have unique effects depending on the current turn number in game. As cards have turn limitations they cannot be played until the required turn is reached. If it is tried to played error windows will be shown and user will be requested to play accordingly or end his/her turn. This situation is checked in P1/2Screen classes so that we didn’t have to check the situations for each card functions. However for bonus effects the validation is checked in those specific cards. 4 cards of the common card pool are overridden for the classes respectively so that players gets much more value out of those cards than the enemy meanwhile the enemy gets the bonus effects for his/her class too.

For hit() function we control some possible numerical errors that could occur throughout the game. Setting armor and durability to zero if it drops below zero is the same control for each hit function in player classes. As each class except Light have bonus effects in their hit function, that function is overridden in those classes to be: Axe dealing %10 more damage to enemy if s/he has no armor, Stone destroys armor %10 more than any other class and Blood restores health equals to the damage s/he deals. Those features are implemented in the Swing part of the project as such:

Firstly, In the Login Screen, J labels are created to enter the nick name of Player 1 and Player 2 using swing. Player 1 chooses one of the four characters in the game.The same process occurs for Player 2. When both players pushed the “Ready! “ . Then they press the start game button. Another swing application are that Player1Screen and Player2Screen. We are warning unwanted situations with message boxes so the user can check the his/her inputs. Another swing screen is that a table can be searched according to the card ID. Also this table showing our card’s name, type, turn and card’s information.

Moreover, we check that some cards can be used after the specified round or during the specified round. Player screen will be shown to player whose turn is the turn to play for him/her.In the dynamic jlabels the health of two players is shown in the remaining cards weapon information and the number of cards they have.Enemy information was also created at the top of the Player1Screen and the Player2Screen. The game card, weapon, end turn button are created.

**CHAPTER SIX**

**CONCLUSION AND FUTURE WORKS**

As a result, we created two player card game and game screens in Swing. In this game, we designed many cards with unique features. Players can choose one of Axe, Blood, Light and Stone as a character and start to game with using their unique cards..

CardMayhem is open to new developments and also new features can be added. As the game progresses, new card features can come. As we wish, lots of future works can be added this game.

Firstly, we discuss and choose our project subject. Then we decided introduction, project’s requirements and added in the report. Also, we start to think about.the necessarily structures like Encapsulation principle, Polymorphism and Inheritance, abstract class and interface usages and List, Stack data structures. We think about all the project together and then implement together.

Gaye created 2 diagrams as Activity diagram and State Chart diagrams of our systems. Gaye created Light class and LightDeck class and wrote unique functions of them like generate and effect functions, created game’s screens in Swing.

Fatma draw 1 State Chart diagram and created Blood class and BloodDeck class and wrote unique functions of them like generate and effect functions, created game’s screens in Swing.

Oğuzhan draw 2 diagrams as Sequence diagrams. Oğuzhan created Stone class and StoneDeck class and wrote unique functions of them like generate and effect functions, created game’s screens in Swing.

Talat draw 1 Activity diagram and created Axe class and AxeDeck class and wrote unique functions of them like generate and effect functions, created game’s screens in Swing. All other parts of projects we made together.