-SE 115-

FATIH ANAMASLI

ProjectCardTry / 20220601002

Which Requirements Are Completed

Functional Requirement 1: The program must be able to create a deck of cards.
Functional Requirement 2: The program must be able to shuffle the deck
Functional Requirement 3: The program must be able to cut the deck.
Functional Requirement 4: The program must be able to move cards from the deck to the players and the boards.
Functional Requirement 5: The program must be able to calculate the player score. Also I calculated the PC score.
Functional Requirement 6: The program must be able to store a "high score list" on a file which stores top 10 scores and names of the players who scored them.
Functional Requirement 7: The program must be able to play for the computer

Extension: when all cards were played, I collected the cards which were in the middleCollectedCard array and sent it to the last winner.

You can see the project's history in GitHub ScreenShot.

1,2,3 are completed in Card Class

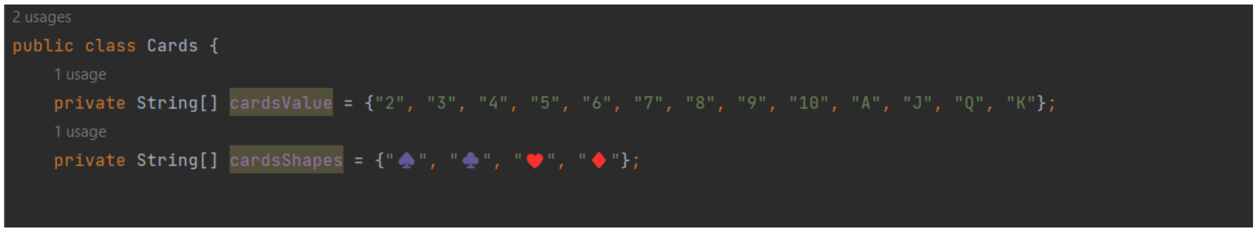
4,5,7,extension are completed in Mechanic Class

6 is completed in ScoreList

Content

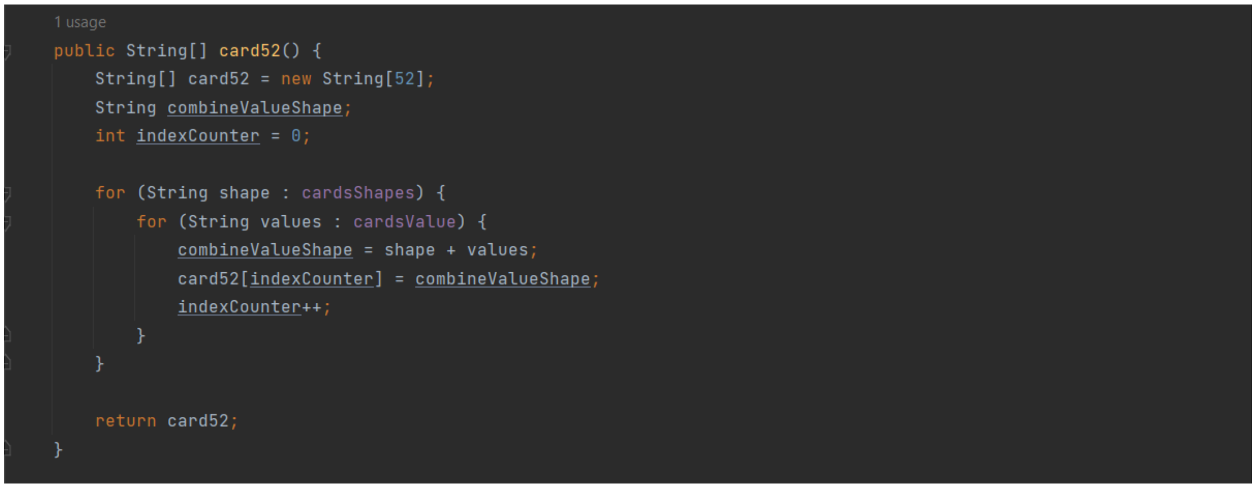
Card Class

Declaration of Cards information and Creating 52 card



```
2 usages
public class Cards {
    1 usage
    private String[] cardsValue = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "A", "J", "Q", "K"};
    1 usage
    private String[] cardsShapes = {"♠", "♣", "♥", "♦"};
```

In this part, I define 2 private variables to store cards symbols and values.



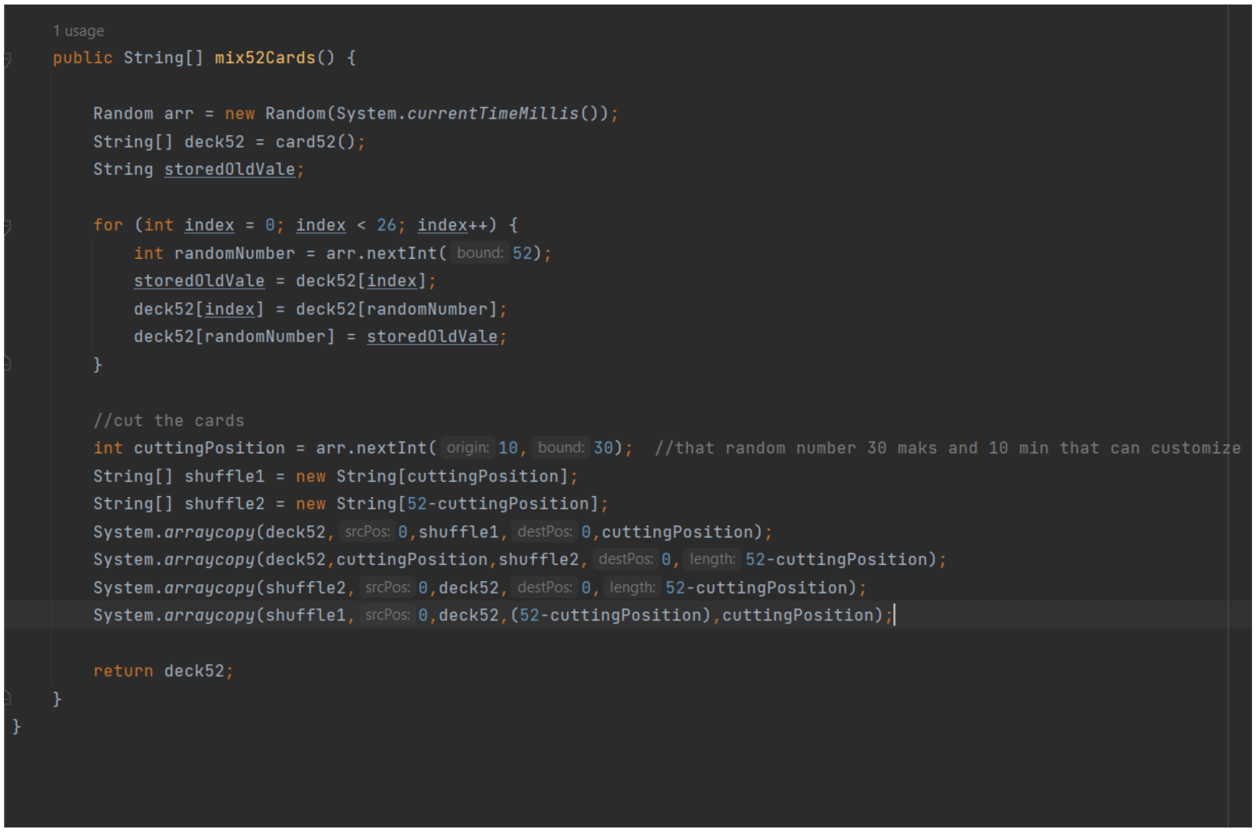
```
    1 usage
    public String[] card52() {
        String[] card52 = new String[52];
        String combineValueShape;
        int indexCounter = 0;

        for (String shape : cardsShapes) {
            for (String values : cardsValue) {
                combineValueShape = shape + values;
                card52[indexCounter] = combineValueShape;
                indexCounter++;
            }
        }

        return card52;
    }
```

I wrote the *card52* method for creating 52 cards. For creating 52 cards, first of all I have to define an array for store created cards. at the first line you can see this array declaration. Then I defined a string for the concatenate card symbol and its value.

Second part of the card52 method I have written a loop which turns *cardsShapese* elements in each execution. Current turned element name in the loop is ***SHAPE***. I had to concatenate the shape of the card with 13 values, so I wrote one more loop which turns the card's value in each execution in the cardsShapes loop. Current turned element name in the *cardValue's* loop is ***VALUES***.

- In the 2 loop, I concatenated ***VALUE*** and ***SHAPE*** in the *combinedValueShape* variable.
- I had to write an integer which counts how much loop execution happened. I used this counter to symbolize card52's all element indexes in order.
- I wrote an integer whose name is *indexCounter* then I increased this variable in each execution. (that means at the and of this process this counter will be 52)
- Lastly, I defined *combinedValueShape to* ***card52****[indexCounter],* at the end of the 2 loop I returned card52.

Shuffle and Cut Card52



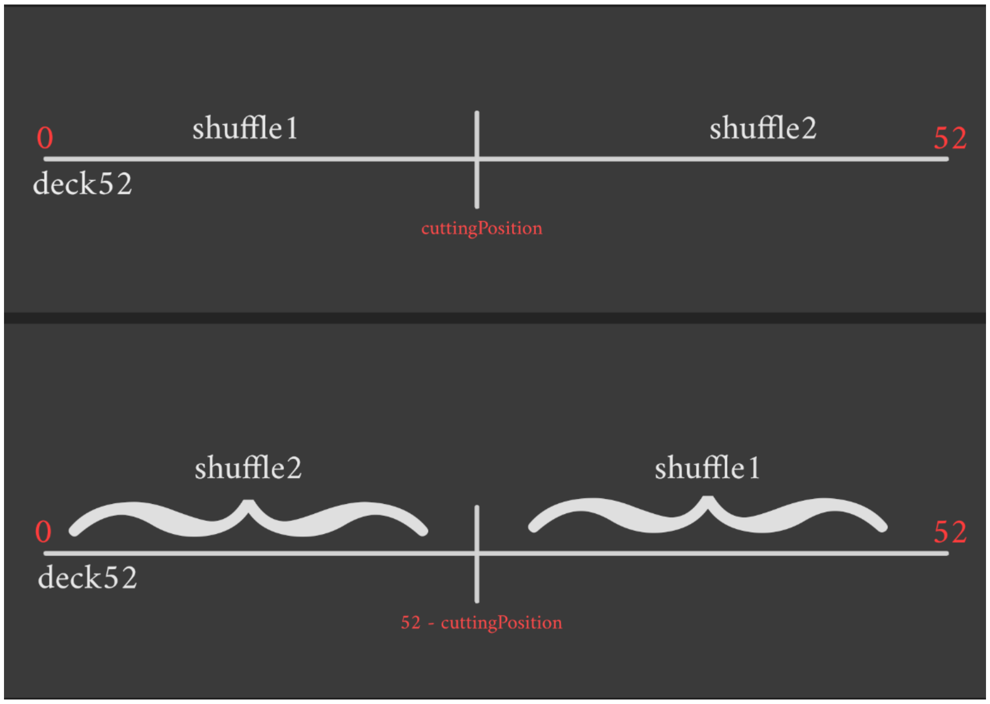
```
1 usage
public String[] mix52Cards() {

    Random arr = new Random(System.currentTimeMillis());
    String[] deck52 = card52();
    String storedOldVale;

    for (int index = 0; index < 26; index++) {
        int randomNumber = arr.nextInt( bound: 52);
        storedOldVale = deck52[index];
        deck52[index] = deck52[randomNumber];
        deck52[randomNumber] = storedOldVale;
    }

    //cut the cards
    int cuttingPosition = arr.nextInt( origin: 10, bound: 30);  //that random number 30 maks and 10 min that can customize
    String[] shuffle1 = new String[cuttingPosition];
    String[] shuffle2 = new String[52-cuttingPosition];
    System.arraycopy(deck52, srcPos: 0,shuffle1, destPos: 0,cuttingPosition);
    System.arraycopy(deck52,cuttingPosition,shuffle2, destPos: 0, length: 52-cuttingPosition);
    System.arraycopy(shuffle2, srcPos: 0,deck52, destPos: 0, length: 52-cuttingPosition);
    System.arraycopy(shuffle1, srcPos: 0,deck52,(52-cuttingPosition),cuttingPosition);

    return deck52;
}
}
```

I had written this function for souffle and cut the card52's elements. First of all, I had to find a solution for shuffling cards. My first idea was to create an empty array ,which has 52 empty null elements, then send card52 elements to this empty array randomly, but I couldn't do that, because I could not prevent copying cards and missing some cards. Then I found this solution. <u>This solution is changing card52 array element position randomly.</u>

- First of all, the random function had to implement in ***mixed52Cards()*** function.
- For shuffle cards I had to create a connection with card52 to take 52 cards information from the card52 function. To do that, I called the card52 function and I stored the returned array in the ***deck52*** variable.
- For changing position of 2 elements, I should create the variable to store variables which will be lost while changing. This variable name is ***storedOldValue*.**
- Loop is written to start to change the element's position. This code is called 26 times. That number doesn't have a specific importance. It can be defined by you how you want.
- I defined *randomNumber* for defined second element index randomly.
- Then I defined ***storedOldValue*** to deck52[index].
- The deck52[index] changed with deck52[randomNumber].
- Now we stored ***deck52***[index] old value in ***storedOldValue***. Thanks to this, we defined ***deck52***[randomNumber]=***storedOldValue***.

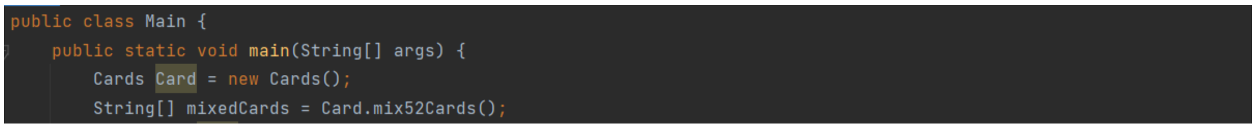
Till here I changed 52 card's positions randomly. Now, I will cut the deck from a random card.

Cut the card deck;



- For cutting cards, first I should declare a random variable whose name is *cuttingPosition*. Also I need 2 arrays which store ***[0 , (cuttingPosition-1)th]*** and ***[cuttingPosition , 51th]*** index.
- Now I should divide 2 part the deck52 cards. These two array names are ***SHUFFLE1 / SHUFFLE2.*** I assigned *suffle1* and *suffle2* with ***arraycopy*** function.
- Then I should assign the *suffle2* array to deck52 from the *0th* index. Then, I start to assign *Suffle1* from the last element index of *suffle2*.
- To explain the cutting code I prepared a photo which is above.
- At the end, I ***returned*** shuffled and cutted cards which are called ***deck 52***.

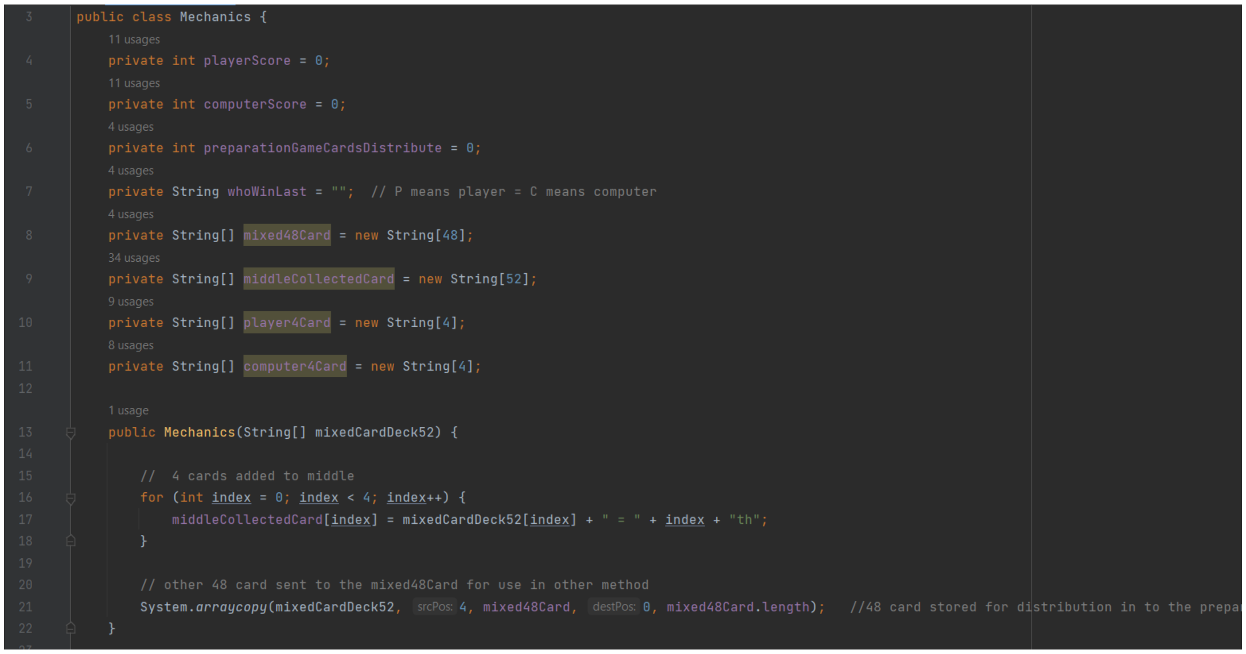
Call Card Class in main:



```
public class Main {
    public static void main(String[] args) {
        Cards Card = new Cards();
        String[] mixedCards = Card.mix52Cards();
```

Lastly, I linked the card class in the main, also I have stored ***mixed52Cards()*** returned value in *mixedCards* variable. I didn't mention why I sored. Shortly, I used this stored data *in Mechanics Class*.

Mechanics Class

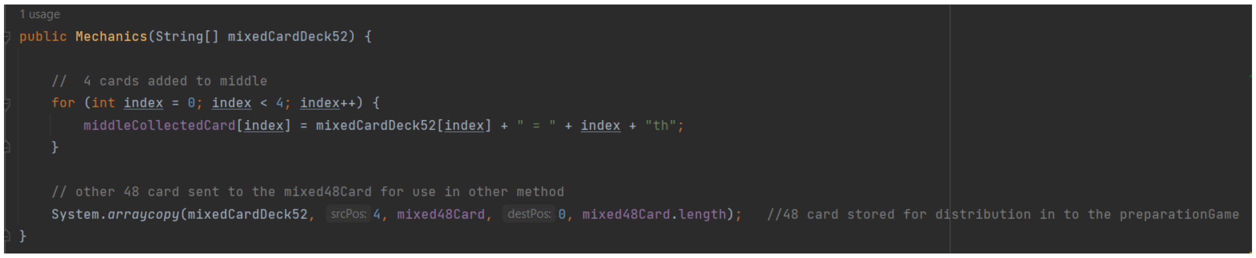


```
public class Mechanics {
    11 usages
    private int playerScore = 0;
    11 usages
    private int computerScore = 0;
    4 usages
    private int preparationGameCardsDistribute = 0;
    4 usages
    private String whoWinLast = "";  // P means player = C means computer
    4 usages
    private String[] mixed48Card = new String[48];
    34 usages
    private String[] middleCollectedCard = new String[52];
    9 usages
    private String[] player4Card = new String[4];
    8 usages
    private String[] computer4Card = new String[4];

    1 usage
    public Mechanics(String[] mixedCardDeck52) {

        //  4 cards added to middle
        for (int index = 0; index < 4; index++) {
            middleCollectedCard[index] = mixedCardDeck52[index] + " = " + index + "th";
        }

        // other 48 card sent to the mixed48Card for use in other method
        System.arraycopy(mixedCardDeck52, srcPos: 4, mixed48Card, destPos: 0, mixed48Card.length);   //48 card stored for distribution in to the prepa
    }
```

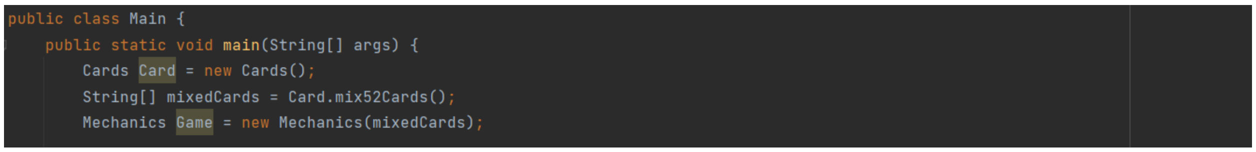
First of all I want to explain the purpose of all these variables before starting to explain each function.

- *playerScore:* Store the player cards scores.
- *computerScore:* Store the computer cards scores.
- *preparationGameCardsDistribute:* While distributing cards to player and computer I used this variable.
- *whoWinLast:* For defining the last winner.
- *mixed48Card:*Stored 48 cards data. (52-4 = 48)
- *middleCollectedCard:* While we were playing games, I should store cards Which are played. I use this array to do that.
- *player4Card:* Player 4 cards stored here.
- *computer4Card:* Computer 4 cards stored here.

Constructor of Mechanic Class



```
1 usage
public Mechanics(String[] mixedCardDeck52) {

    //  4 cards added to middle
    for (int index = 0; index < 4; index++) {
        middleCollectedCard[index] = mixedCardDeck52[index] + " = " + index + "th";
    }

    // other 48 card sent to the mixed48Card for use in other method
    System.arraycopy(mixedCardDeck52, srcPos: 4, mixed48Card, destPos: 0, mixed48Card.length);   //48 card stored for distribution in to the preparationGame
}
```

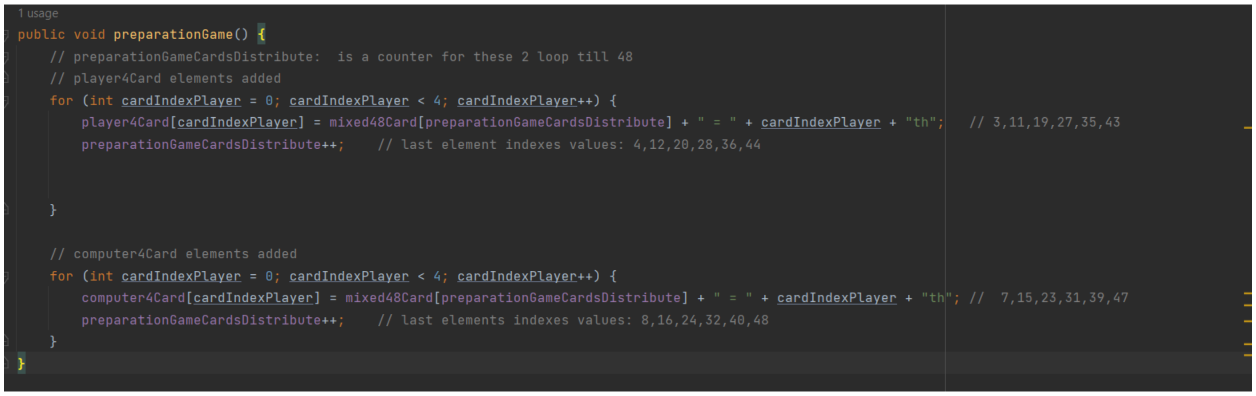


```
public class Main {
    public static void main(String[] args) {
        Cards Card = new Cards();
        String[] mixedCards = Card.mix52Cards();
        Mechanics Game = new Mechanics(mixedCards);
```

Thanks to the constructor function, I can take 52 mixed and cutted cards into the Mechanic Class. *mixedCardDeck52* is the key point of transporming cards to the Mechanic Class.

Then at the beginning of the game, I had to send 4 cards to the middle for just one time, thusI completed this requirement in the constructor function. After that, I should delete 4 cards from ***mixedCardDeck52*. To do it I wrote a new array whose name is *mixed48Card*** and mixedCardDeck52's 48 cards [4index,52index) copied to mixed48Cards array by arrayCopy function.

preperationGame Function



```
1 usage
public void preparationGame() {
    // preparationGameCardsDistribute:  is a counter for these 2 loop till 48
    // player4Card elements added
    for (int cardIndexPlayer = 0; cardIndexPlayer < 4; cardIndexPlayer++) {
        player4Card[cardIndexPlayer] = mixed48Card[preparationGameCardsDistribute] + " = " + cardIndexPlayer + "th";   // 3,11,19,27,35,43
        preparationGameCardsDistribute++;    // last element indexes values: 4,12,20,28,36,44


    }

    // computer4Card elements added
    for (int cardIndexPlayer = 0; cardIndexPlayer < 4; cardIndexPlayer++) {
        computer4Card[cardIndexPlayer] = mixed48Card[preparationGameCardsDistribute] + " = " + cardIndexPlayer + "th"; //  7,15,23,31,39,47
        preparationGameCardsDistribute++;    // last elements indexes values: 8,16,24,32,40,48
    }
}
```

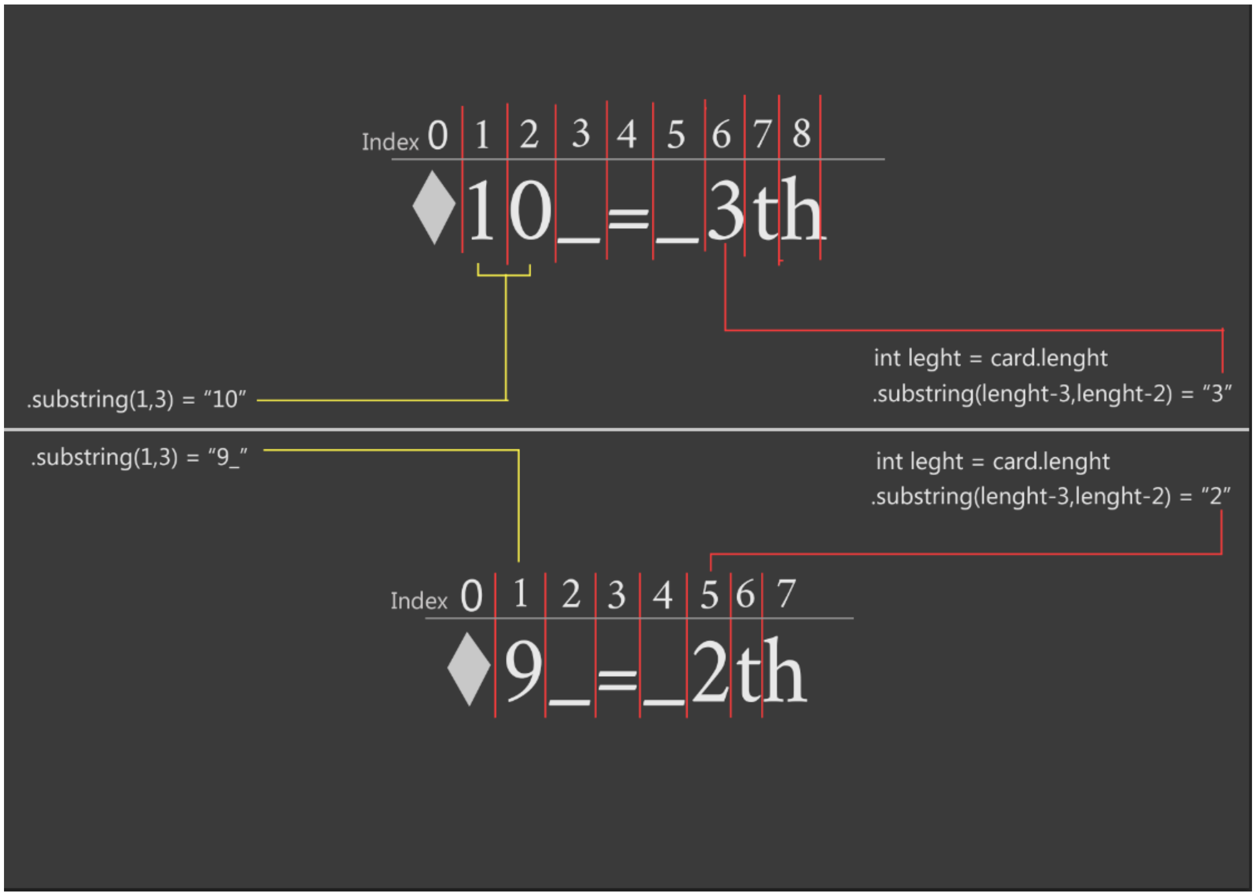
The Preparation method's mission is, send 4 cards to the player and computer each execution. Also this function has to be usable many times directly in the game process.

I wrote 2 loops which execute 4 times after the execution of the function. One of these loop's was written for assign 4 cards to player4Card, other was written for assign 4 card to Computer4Card.

Faced ERROR: Due to running 2 loops separately I should define an integer which counts how many times these loops are executed. According to this integer's value, I defined the index *number of **mixed48Card's***. I forgot to mention the integer variable name, it's name is ***preparationGameCardsDistribute***.

Fundamental step of checking point and selecting card to play: Computer and player's 4 cards assigned like ***"♠10 = 3th"***, and I explained that in figure.

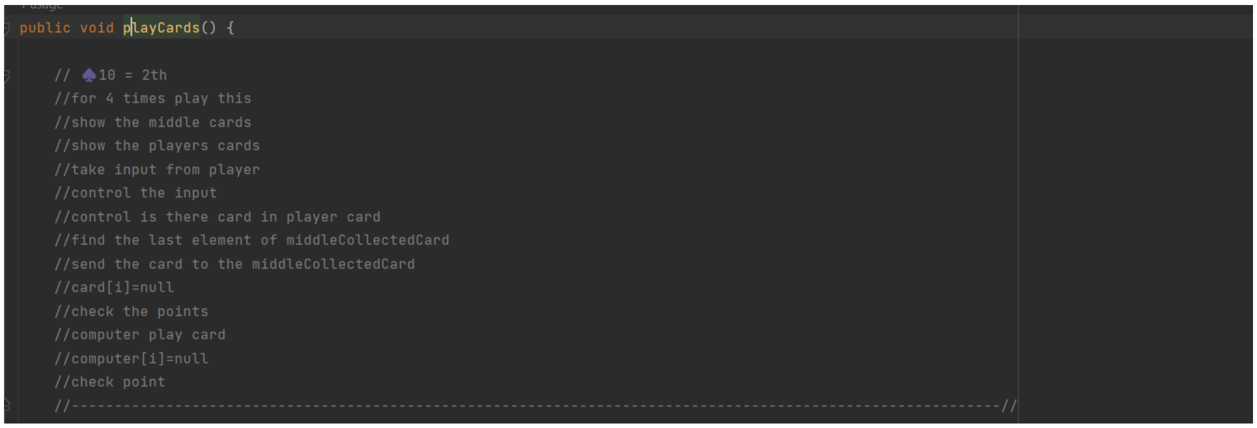
Fig 2 Card Value System



10th Cards Problem: In my other project a lot of errors appeared because of 10th cards. I found a solution which is shown in the figure. That assignment gives me a chance to write less if-else statements, because thanks to this assignment of cards, I could use the same codes for 10th and other cards.

- **YELLOW:** Used for *pointCounterComputer ,pointCounterPlayer,controlCard*.
- ***RED:*** *isThereCardSelected functions.*

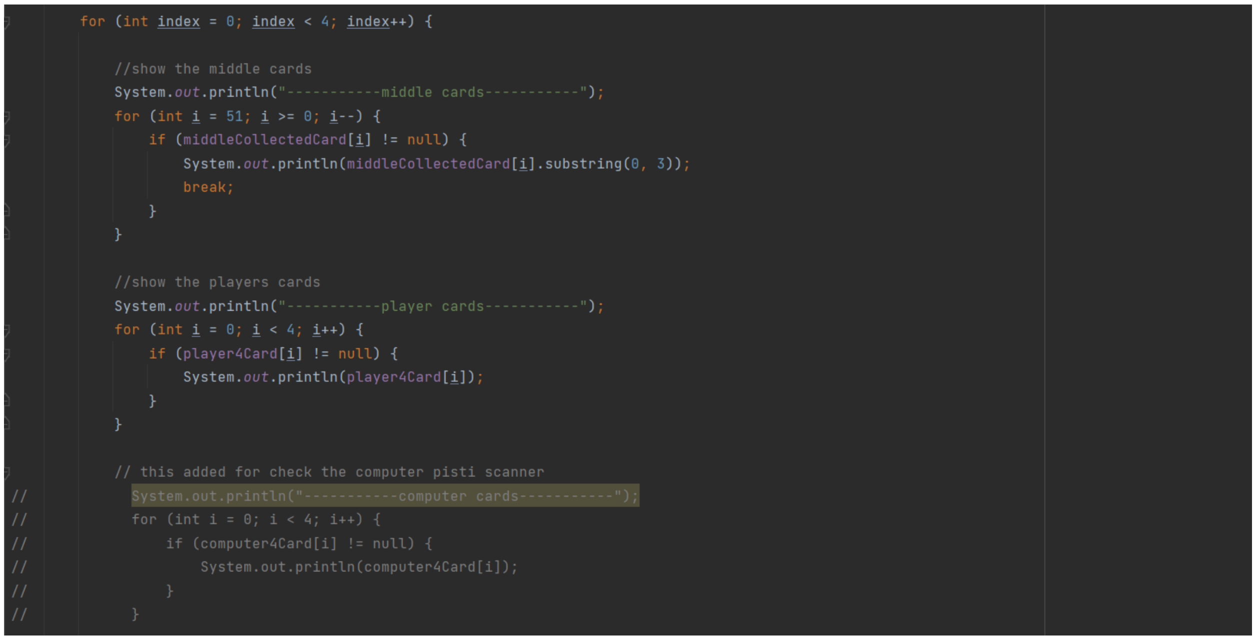
PlayCards Function



```
public void playCards() {

    // ♠10 = 2th
    //for 4 times play this
    //show the middle cards
    //show the players cards
    //take input from player
    //control the input
    //control is there card in player card
    //find the last element of middleCollectedCard
    //send the card to the middleCollectedCard
    //card[i]=null
    //check the points
    //computer play card
    //computer[i]=null
    //check point
    //---------------------------------------------------------------------------------------------------------------//
```

Playcards function is the center of the game process. Till here, I prepared the games requirements for playing ***PİŞTİ***. In this method I called other functions in order and managed the data transfer. In the figure above you can see this function's ***pseudocode*****,** It explains the game process step by step. Lastly, you will see lines ,which are interested with function, above the functions according to functions missions.

Show the Cards of player and middle



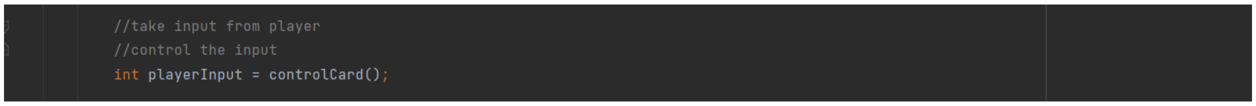
```
        for (int index = 0; index < 4; index++) {

            //show the middle cards
            System.out.println("-----------middle cards-----------");
            for (int i = 51; i >= 0; i--) {
                if (middleCollectedCard[i] != null) {
                    System.out.println(middleCollectedCard[i].substring(0, 3));
                    break;
                }
            }

            //show the players cards
            System.out.println("-----------player cards-----------");
            for (int i = 0; i < 4; i++) {
                if (player4Card[i] != null) {
                    System.out.println(player4Card[i]);
                }
            }

            // this added for check the computer pisti scanner
//              System.out.println("-----------computer cards-----------");
//              for (int i = 0; i < 4; i++) {
//                  if (computer4Card[i] != null) {
//                      System.out.println(computer4Card[i]);
//                  }
//              }
```

In this part I wrote 2 loops to show the middle last card and player 4 cards. ***First loop*** was written to show the ***middleCollectedCard’s*** last element’s first 3 char values. I scanned all elements of the array from end to start, if it finds a value which is different from null, prints it to the screen. After that, it breaks the loop. ***Second loop*** was written to show the player's 4 cards.

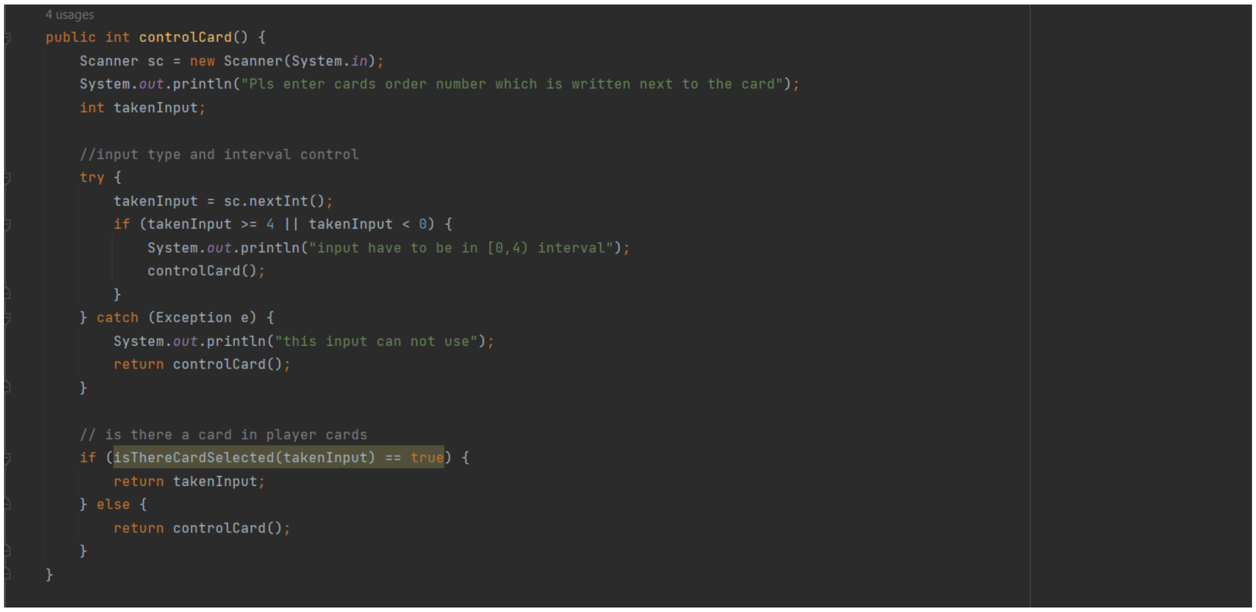
(NOTE: While we were playing the game, computer and player cards were sent to the middleCollectedCard. The player and computer card’s values format is ***like this “♠10 = 3th''***. so, I should show just 3 char values from the 0th index, when I wanted to show cards symbol and numbers.)

ERROR: When a player played his/her cards I should delete them: To solve this problem I added an if statement which controls the player4Card’s element's values. If its values equal null, it won’t write the element of player4Card.

Take Input From User and Control It



```java
        //take input from player
        //control the input
        int playerInput = controlCard();
```



```java
4 usages
public int controlCard() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Pls enter cards order number which is written next to the card");
    int takenInput;

    //input type and interval control
    try {
        takenInput = sc.nextInt();
        if (takenInput >= 4 || takenInput < 0) {
            System.out.println("input have to be in [0,4) interval");
            controlCard();
        }
    } catch (Exception e) {
        System.out.println("this input can not use");
        return controlCard();
    }

    // is there a card in player cards
    if (isThereCardSelected(takenInput) == true) {
        return takenInput;
    } else {
        return controlCard();
    }
}
```

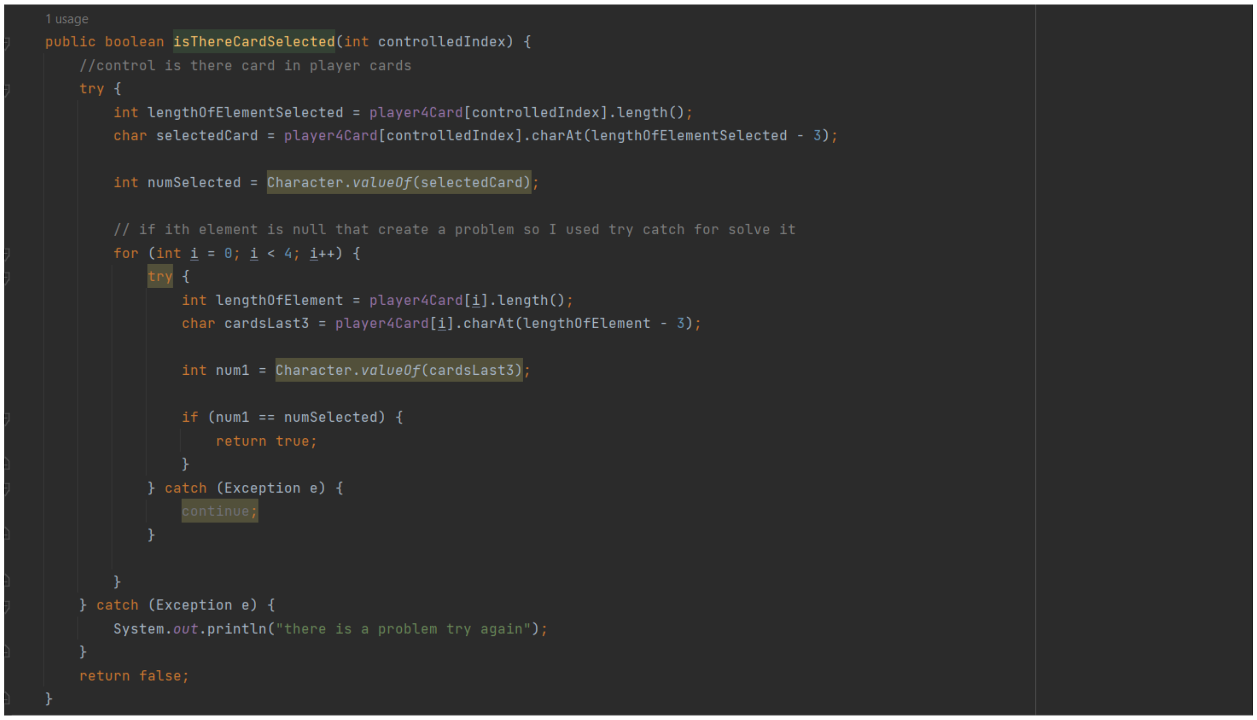
The ControlCard function was written to take input from the user and control it. If input can not be used to play the cards, the function should take new input from the user.

let's dive in controlCard function:

- First of all I defined a scanner to take input from the user.
- Short information about what he/she should do.
- Integer defined, but I didn't give any value to it. Its value will be defined by the player.
- Take input which *symbolizes the card's order number* // input can be (0,1,2,3).
- Players can enter wrong input, so code should catch this mistake and give feedback to the player about what is the problem. I used ***try-catch*** for giving feedback to players if there is any problem.
- I put an if statement which checks 2 important features of input. The If statement checks the input type. If the input type is not suitable to use, catch's part's code will execute and call the ***controlCard()*** function again ***(recursion)***. If the input type is suitable to use, the second parameter is controlled by the if statement. If input is not in the [0,4) interval, the If statement calls ***controlCard()*** function again for taking new input from the player.
- When the input had suitable features to use, the second if statement controlled the player cards. If there are no cards which have the same order number with input, code call the ***controlCard()*** again, and give feedback about this situation. The Player4Card array scans with ***isThereCardSelected(takenInput)***

Before continuing to explain other codes, the ***isThereCardSelected(takenInput)'s*** turns came to it.

isThereCardSelected(takenInput):



```
1 usage
public boolean isThereCardSelected(int controlledIndex) {
    //control is there card in player cards
    try {
        int lengthOfElementSelected = player4Card[controlledIndex].length();
        char selectedCard = player4Card[controlledIndex].charAt(lengthOfElementSelected - 3);

        int numSelected = Character.valueOf(selectedCard);

        // if ith element is null that create a problem so I used try catch for solve it
        for (int i = 0; i < 4; i++) {
            try {
                int lengthOfElement = player4Card[i].length();
                char cardsLast3 = player4Card[i].charAt(lengthOfElement - 3);

                int num1 = Character.valueOf(cardsLast3);

                if (num1 == numSelected) {
                    return true;
                }
            } catch (Exception e) {
                continue;
            }

        }
    } catch (Exception e) {
        System.out.println("there is a problem try again");
    }
    return false;
}
```

I wrote this function to control the input which was taken from the user. If input is not in the player4Deck array, it will return false.

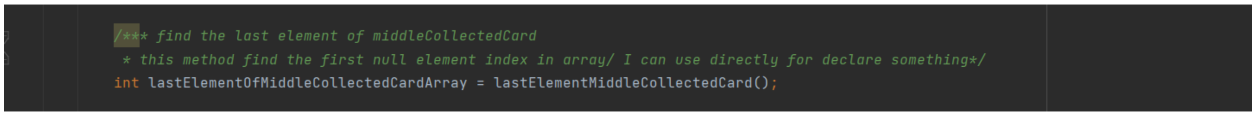
ERROR: If there is no card which has the same order number with Input in player4Card array, int *lengthOfElementSelected* and *selectedCard* can not be found. That creates the error,and stops the program. To prevent it, I wrote try-catch.

How to code works:

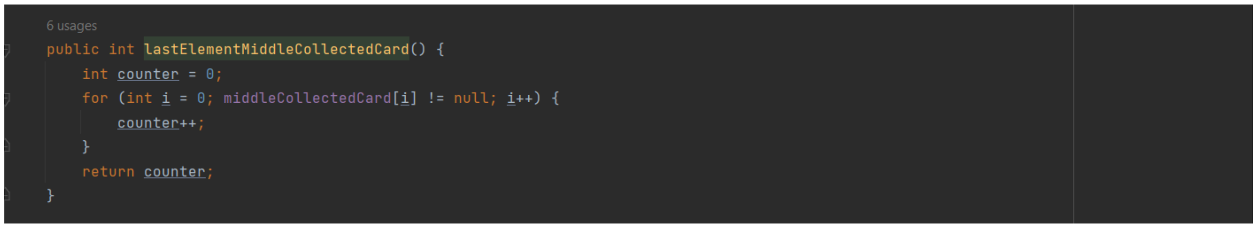
To determine the checked sequence number, I must first find the length of the String variable. Then, 3 minus of the number value found will give the sequence number of the card to be discarded. Afterwards, the 3rd-last values of all Player4Card's cards are compared with the rank values of the checked card. It is expected to be the same, otherwise the function returns false.

Lastly, *isThereCardSelected* returns true or false to the *controlCard* function. If it returns true that means there is a card which has got the same order with input.

lastElementMiddleColldCardecte Function



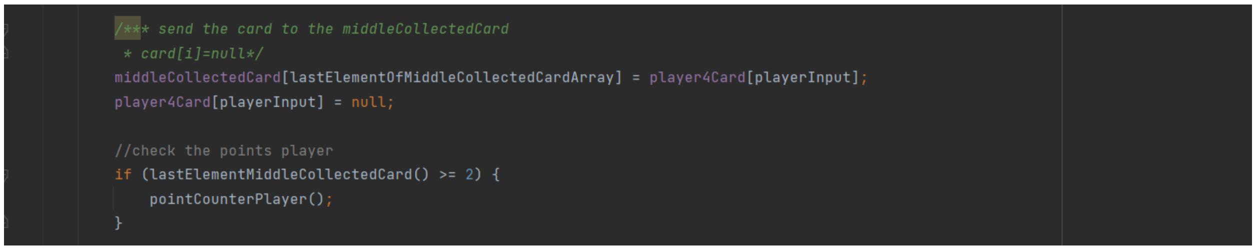
```
/*** find the last element of middleCollectedCard
 * this method find the first null element index in array/ I can use directly for declare something*/
int lastElementOfMiddleCollectedCardArray = lastElementMiddleCollectedCard();
```



```
6 usages
public int lastElementMiddleCollectedCard() {
    int counter = 0;
    for (int i = 0; middleCollectedCard[i] != null; i++) {
        counter++;
    }
    return counter;
}
```

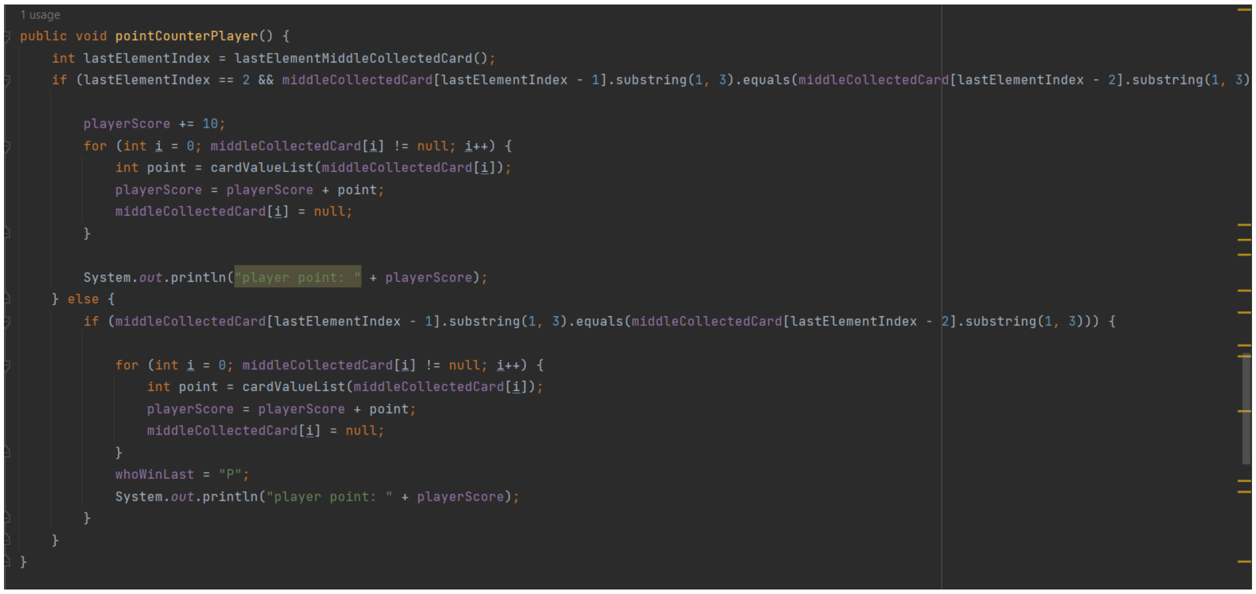
This function is used to find the ***middleCollectedCard*** array's last elements index which is different from null. I used this method many times in other functions. I have to find the last null index's number which is in the middleCollectedCard, because the selected card will be sent to the middleCollectedCard.Lastly, that function returns int value and this value stored in ***lastElementOfMiddleCollectedCard.***

PointCounterPlayer Function



```
/*** send the card to the middleCollectedCard
 * card[i]=null*/
middleCollectedCard[lastElementOfMiddleCollectedCardArray] = player4Card[playerInput];
player4Card[playerInput] = null;

//check the points player
if (lastElementMiddleCollectedCard() >= 2) {
    pointCounterPlayer();
}
```

Before calling this function, player4Card’s selectedIndex is sent to the middleCollectedCard’s ***lastElementOfMiddleCollectedCardCard.*** Then to protect the number of cards, I assigned player4Card’s selectedIndex to the null.



```
1 usage
public void pointCounterPlayer() {
    int lastElementIndex = lastElementMiddleCollectedCard();
    if (lastElementIndex == 2 && middleCollectedCard[lastElementIndex - 1].substring(1, 3).equals(middleCollectedCard[lastElementIndex - 2].substring(1, 3)

        playerScore += 10;
        for (int i = 0; middleCollectedCard[i] != null; i++) {
            int point = cardValueList(middleCollectedCard[i]);
            playerScore = playerScore + point;
            middleCollectedCard[i] = null;
        }

        System.out.println("player point: " + playerScore);
    } else {
        if (middleCollectedCard[lastElementIndex - 1].substring(1, 3).equals(middleCollectedCard[lastElementIndex - 2].substring(1, 3))) {

            for (int i = 0; middleCollectedCard[i] != null; i++) {
                int point = cardValueList(middleCollectedCard[i]);
                playerScore = playerScore + point;
                middleCollectedCard[i] = null;
            }
            whoWinLast = "P";
            System.out.println("player point: " + playerScore);
        }
    }
}
```

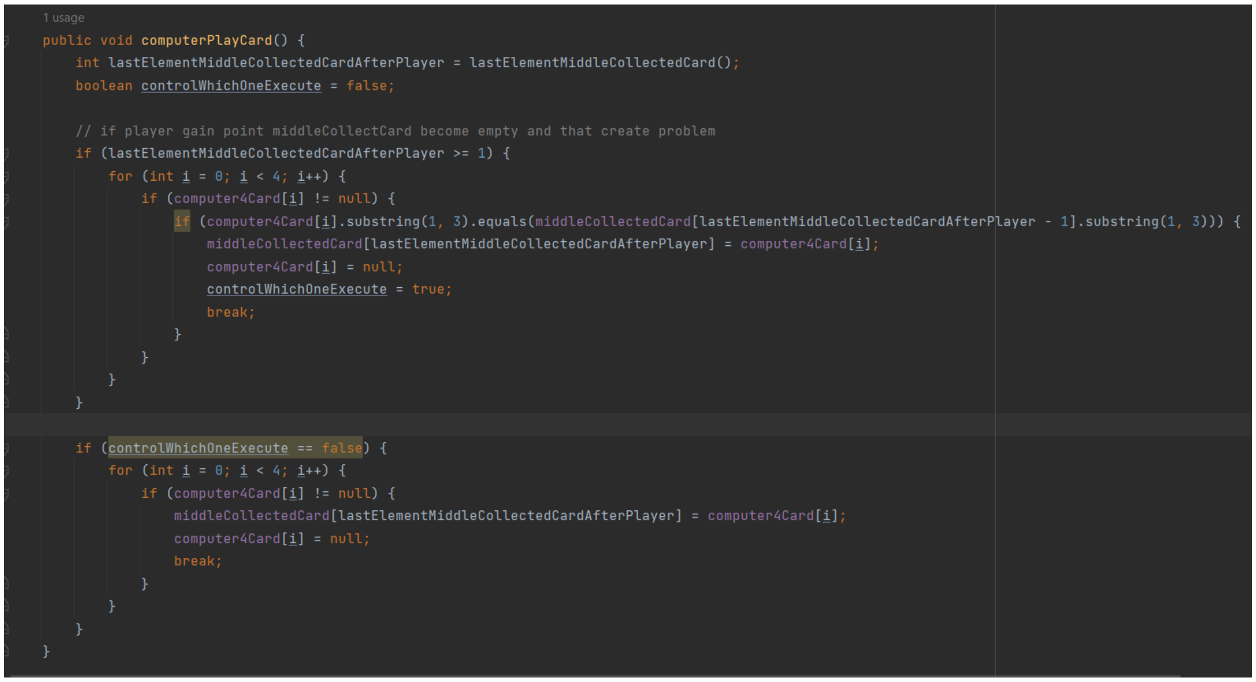
This function is controlling the middleCollectedCard last 2 elements card's values. If they are equal to each other, the player gains points. Also this function controls PİŞTİ. It controls the PISTI situation with ***lastElementOfMiddleCollectedCardCard*** function. If it returns 2 that means there are just 2 elements in midleCollectedCard. Also if ***lastElementOfMiddleCollectedCardCard*** doesn't return 2, and last 2 elements order numbers equal to each other, that means this situation is not PISTI but the player will collect middleCollectedCard.

PointCounterComputer Function

İt works the same with the PointCounterPlayer function. I just changed playerScore to computerScore

ERROR: For these 2 functions, I had to write an if statement which controls the middleCollectedCard elements number which is different from null. I controlled the last 2 card's values in these methods. If there are not 2 cards in middleCollectedCard, that leads to an error .I solved this problem by adding an if statement.

ComputerPlayCard Function



```
1 usage
public void computerPlayCard() {
    int lastElementMiddleCollectedCardAfterPlayer = lastElementMiddleCollectedCard();
    boolean controlWhichOneExecute = false;

    // if player gain point middleCollectCard become empty and that create problem
    if (lastElementMiddleCollectedCardAfterPlayer >= 1) {
        for (int i = 0; i < 4; i++) {
            if (computer4Card[i] != null) {
                if (computer4Card[i].substring(1, 3).equals(middleCollectedCard[lastElementMiddleCollectedCardAfterPlayer - 1].substring(1, 3))) {
                    middleCollectedCard[lastElementMiddleCollectedCardAfterPlayer] = computer4Card[i];
                    computer4Card[i] = null;
                    controlWhichOneExecute = true;
                    break;
                }
            }
        }
    }

    if (controlWhichOneExecute == false) {
        for (int i = 0; i < 4; i++) {
            if (computer4Card[i] != null) {
                middleCollectedCard[lastElementMiddleCollectedCardAfterPlayer] = computer4Card[i];
                computer4Card[i] = null;
                break;
            }
        }
    }
}
```

As you predicted, I wrote this function to declare how the computer will play. You see two if statements. In the first statement which is above, the computer looks at the ***middleCardsArray*** last card's values and ***compares this card's values with computer4Cards cards values***. If the computer finds the card which has the same value, it will play this card first. else the computer executes the second if statement. In this method the computer plays a card regularly.

ERROR: `middleCollectedCard[lastElementMiddleCollectedCardAfterPlayer - 1]`

This line created a problem when I collected all the cards. Because if there is no card lastElementMiddleCollectedCardAfterPlayer will be equal to 0 then code makes -1. At the end of this codes looks for ***middleColelctedCard[-1]*** element and that creates error.I added if statement to prevent this error.

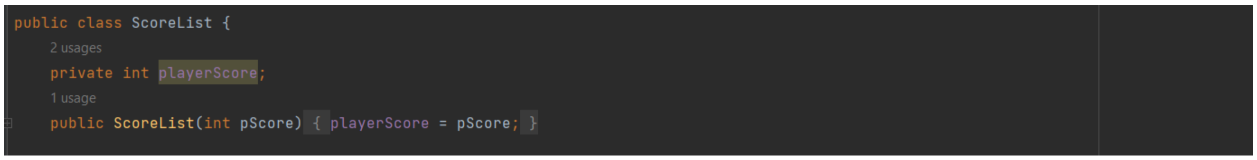
ERROR: I had to create a system to prevent executing these if statements. For do that, I ***addedcontrolledWhichOneExecute*** boolean variable to control it. If the first one executes, the second will not because the boolean variable changes TRUE, if the first one does not execute, the second will execute.

ScoreList Class

I will explain step by step to this class first, then I will mention every step of the code shortly.

- Take playerScore from main with constructor
- Declare 2D array [11][2]
- Read text file data
- Store data in 2D array
- Arrange array to highest point to lowest point
- Clear text file
- Write new arranged array first 10 elements

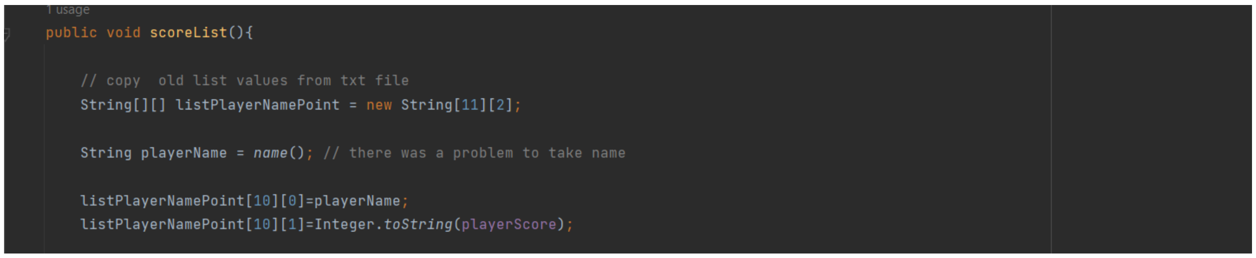
Take playerScore from main with constructor



```
public class ScoreList {
    2 usages
    private int playerScore;
    1 usage
    public ScoreList(int pScore) { playerScore = pScore; }
```

Constructed provides a chance to taking playerScore from main.

Declare 2D array [11][2] and take name()

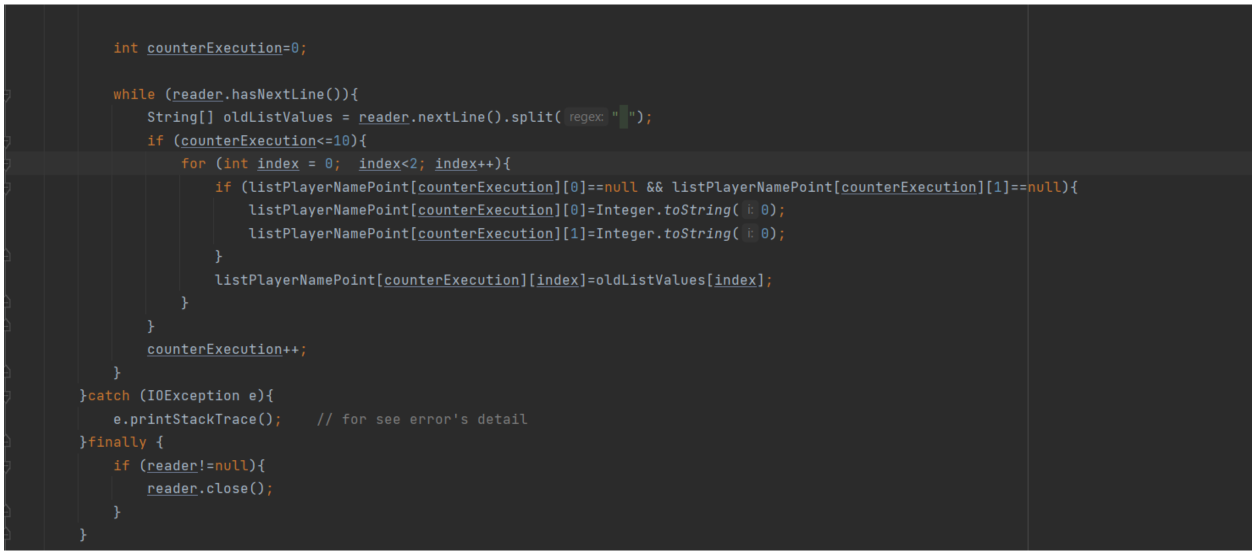


```
1 usage
public void scoreList(){

    // copy  old list values from txt file
    String[][] listPlayerNamePoint = new String[11][2];

    String playerName = name(); // there was a problem to take name

    listPlayerNamePoint[10][0]=playerName;
    listPlayerNamePoint[10][1]=Integer.toString(playerScore);
```

I had declared the 2D array which size is [11][2]. 11 is a key point, because it provides me with an easy solution to sorting bigger to smaller. At that time also I added player name and score to this array's last indexes.

ERROR: I used the name function to take the user's name, because some problem exists because of the Scanner. To solve it, I decided to write a function for taking the player's name.

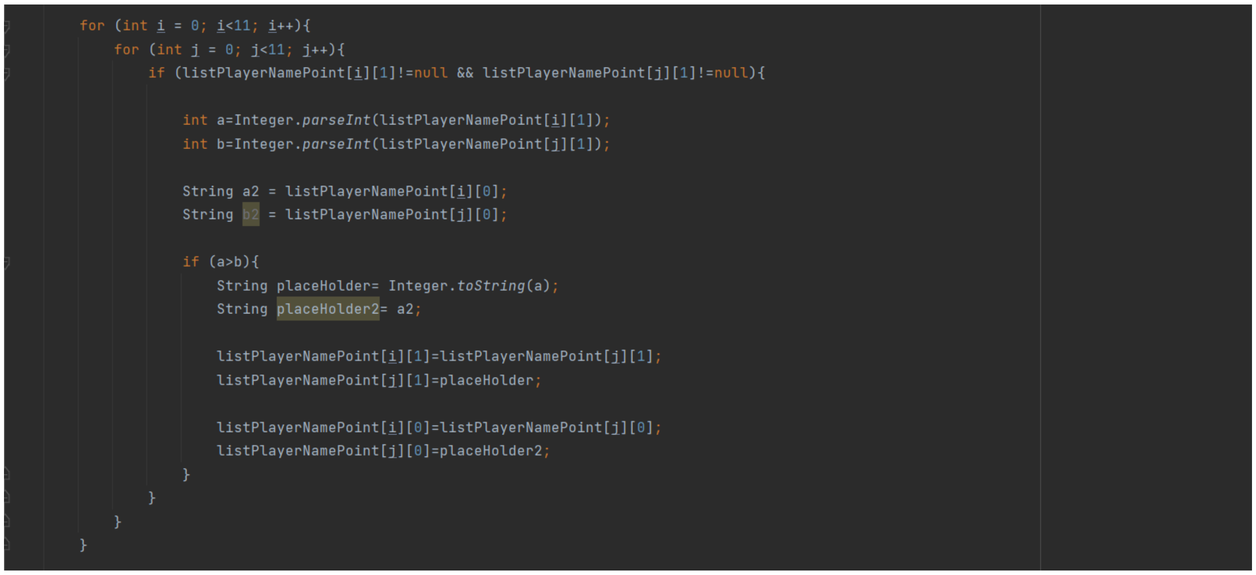
Read text file and store data in 2D array



```
        int counterExecution=0;

        while (reader.hasNextLine()){
            String[] oldListValues = reader.nextLine().split( regex: " ");
            if (counterExecution<=10){
                for (int index = 0;  index<2; index++){
                    if (listPlayerNamePoint[counterExecution][0]==null && listPlayerNamePoint[counterExecution][1]==null){
                        listPlayerNamePoint[counterExecution][0]=Integer.toString( i: 0);
                        listPlayerNamePoint[counterExecution][1]=Integer.toString( i: 0);
                    }
                    listPlayerNamePoint[counterExecution][index]=oldListValues[index];
                }
            }
            counterExecution++;
        }
    }catch (IOException e){
        e.printStackTrace();    // for see error's detail
    }finally {
        if (reader!=null){
            reader.close();
        }
    }
```

As you see, I read the text file with the reader and sell data to the array. I split each line with (“ “) . The first one is the name, the second one is the score. If statement is added to prevent errors, because null leads to a problem. Instead of null I added 0 in this part.

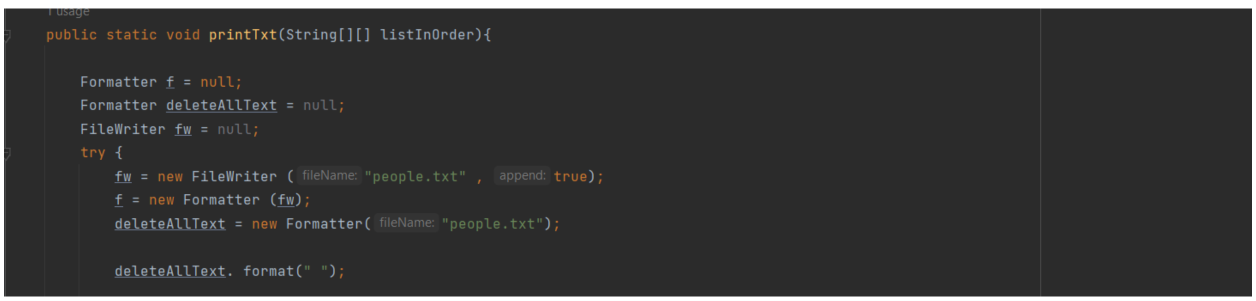
Arrange array to highest point to lowest point



```java
for (int i = 0; i<11; i++){
    for (int j = 0; j<11; j++){
        if (listPlayerNamePoint[i][1]!=null && listPlayerNamePoint[j][1]!=null){

            int a=Integer.parseInt(listPlayerNamePoint[i][1]);
            int b=Integer.parseInt(listPlayerNamePoint[j][1]);

            String a2 = listPlayerNamePoint[i][0];
            String b2 = listPlayerNamePoint[j][0];

            if (a>b){
                String placeHolder= Integer.toString(a);
                String placeHolder2= a2;

                listPlayerNamePoint[i][1]=listPlayerNamePoint[j][1];
                listPlayerNamePoint[j][1]=placeHolder;

                listPlayerNamePoint[i][0]=listPlayerNamePoint[j][0];
                listPlayerNamePoint[j][0]=placeHolder2;
            }
        }
    }
}
```

I sorted scores bigger to smaller then I sorted names according to scores.

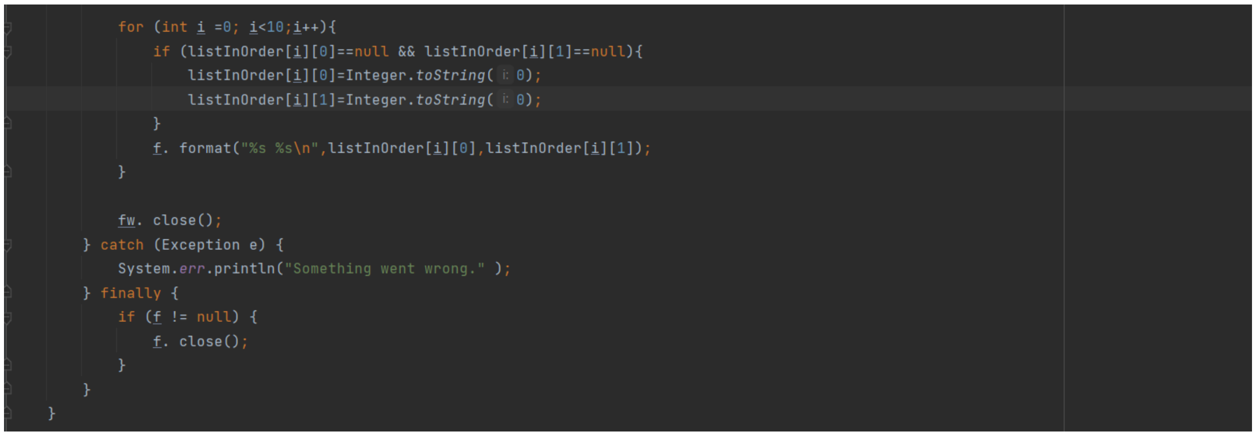
Clear text file



```java
1 usage
public static void printTxt(String[][] listInOrder){

    Formatter f = null;
    Formatter deleteAllText = null;
    FileWriter fw = null;
    try {
        fw = new FileWriter ( fileName: "people.txt" ,  append: true);
        f = new Formatter (fw);
        deleteAllText = new Formatter( fileName: "people.txt");

        deleteAllText. format(" ");
```

I defined a format, formatter and one more formatter to delete all text file. I used a formatter which wrote to delete text file.

Write new arranged array first 10 elements



```
            for (int i =0; i<10;i++){
                if (listInOrder[i][0]==null && listInOrder[i][1]==null){
                    listInOrder[i][0]=Integer.toString( i: 0);
                    listInOrder[i][1]=Integer.toString( i: 0);
                }
                f. format("%s %s\n",listInOrder[i][0],listInOrder[i][1]);
            }

            fw. close();
        } catch (Exception e) {
            System.err.println("Something went wrong." );
        } finally {
            if (f != null) {
                f. close();
            }
        }
    }
```

In that place I print a new sorted array to the text file.

ERROR: I mentioned about null problem, but one more problem and that is the empty place. ***If the people.txt file is empty,*** that leads to a problem. To prevent this error I edited the people.txt file for the first execution.

0 0

0 0

0 0

0 0

0 0

0 0

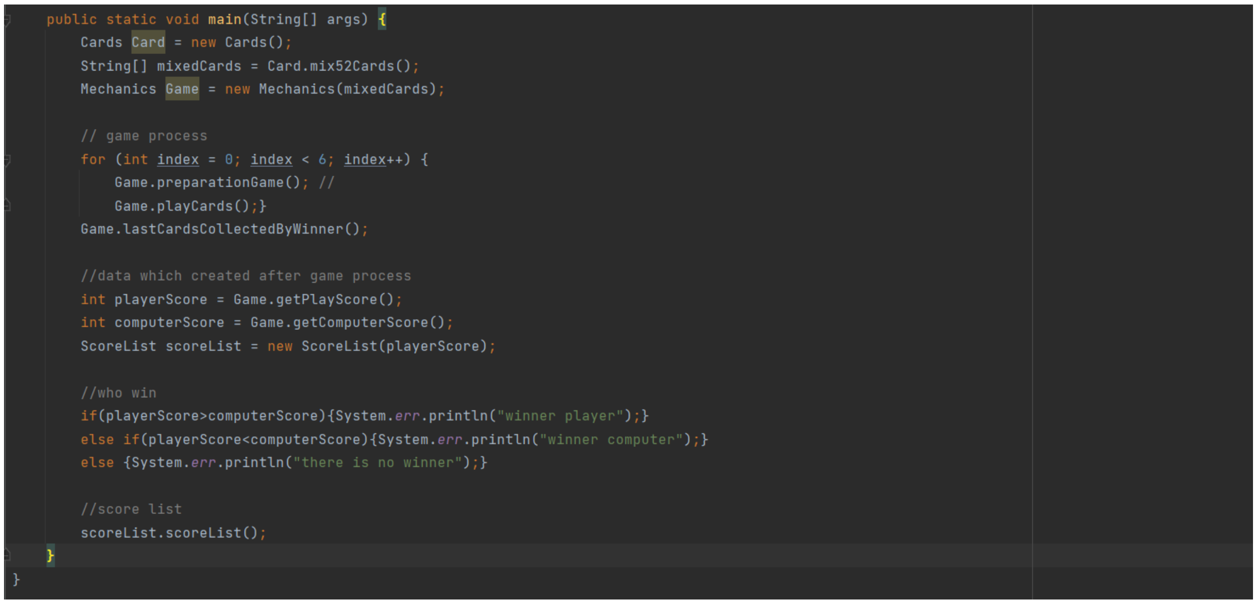
0 0

0 0

0 0

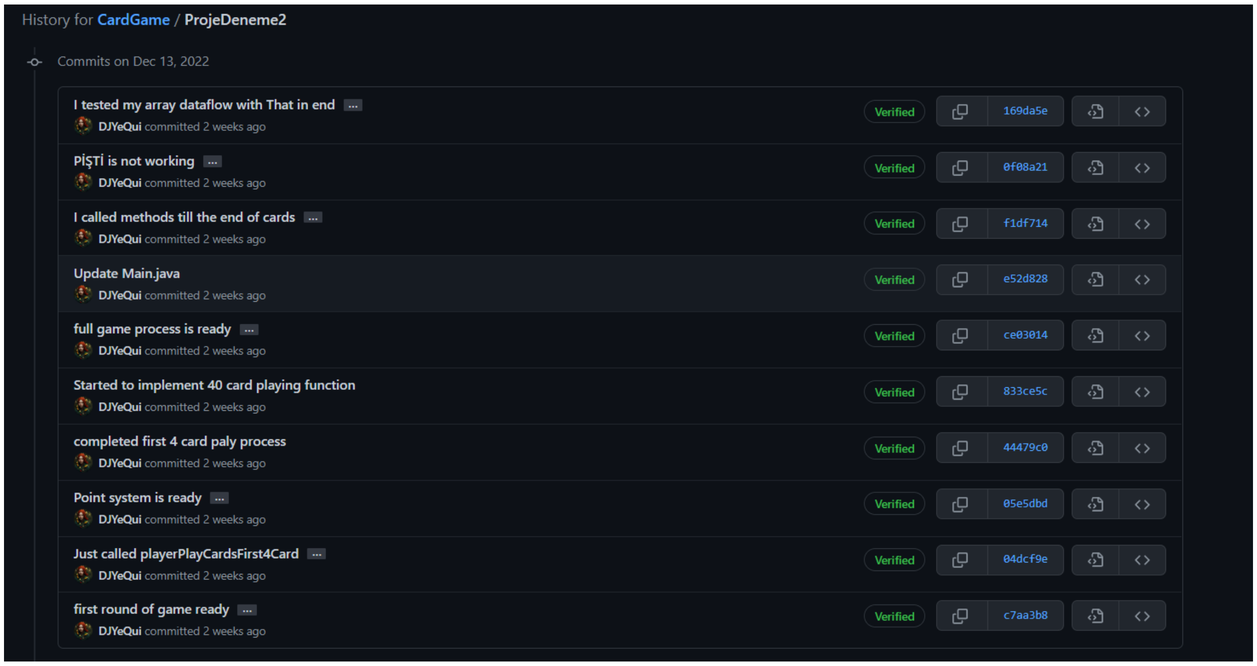
0 0

Main Class


```
    public static void main(String[] args) {
        Cards Card = new Cards();
        String[] mixedCards = Card.mix52Cards();
        Mechanics Game = new Mechanics(mixedCards);

        // game process
        for (int index = 0; index < 6; index++) {
            Game.preparationGame(); //
            Game.playCards();}
        Game.lastCardsCollectedByWinner();

        //data which created after game process
        int playerScore = Game.getPlayScore();
        int computerScore = Game.getComputerScore();
        ScoreList scoreList = new ScoreList(playerScore);

        //who win
        if(playerScore>computerScore){System.err.println("winner player");}
        else if(playerScore<computerScore){System.err.println("winner computer");}
        else {System.err.println("there is no winner");}

        //score list
        scoreList.scoreList();
    }
}
```

I added the main class at the end of the code explanation, because It gives you a bird's eye view.

History of GitHub


History for CardGame / ProjeDeneme2

Commits on Dec 13, 2022

I tested my array dataflow with That in end ...
DJYeQui committed 2 weeks ago Verified 169da5e

PİŞTİ is not working ...
DJYeQui committed 2 weeks ago Verified 0f08a21

I called methods till the end of cards ...
DJYeQui committed 2 weeks ago Verified f1df714

Update Main.java
DJYeQui committed 2 weeks ago Verified e52d828

full game process is ready ...
DJYeQui committed 2 weeks ago Verified ce03014

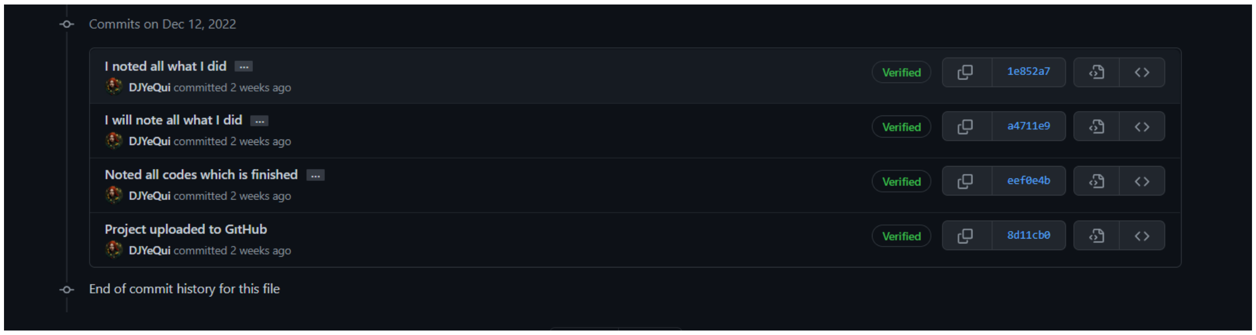
Started to implement 40 card playing function
DJYeQui committed 2 weeks ago Verified 833ce5c

completed first 4 card paly process
DJYeQui committed 2 weeks ago Verified 44479c0

Point system is ready ...
DJYeQui committed 2 weeks ago Verified 05e5dbd

Just called playerPlayCardsFirst4Card ...
DJYeQui committed 2 weeks ago Verified 04dcf9e

first round of game ready ...
DJYeQui committed 2 weeks ago Verified c7aa3b8


Commits on Dec 12, 2022

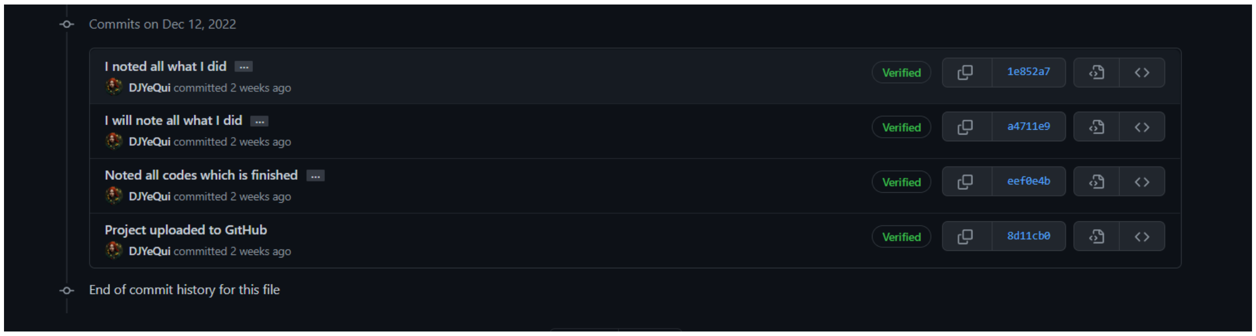
I noted all what I did ...
DJYeQui committed 2 weeks ago Verified 1e852a7

I will note all what I did ...
DJYeQui committed 2 weeks ago Verified a4711e9

Noted all codes which is finished ...
DJYeQui committed 2 weeks ago Verified eef0e4b

Project uploaded to GitHub
DJYeQui committed 2 weeks ago Verified 8d11cb0

End of commit history for this file


Commits on Dec 12, 2022

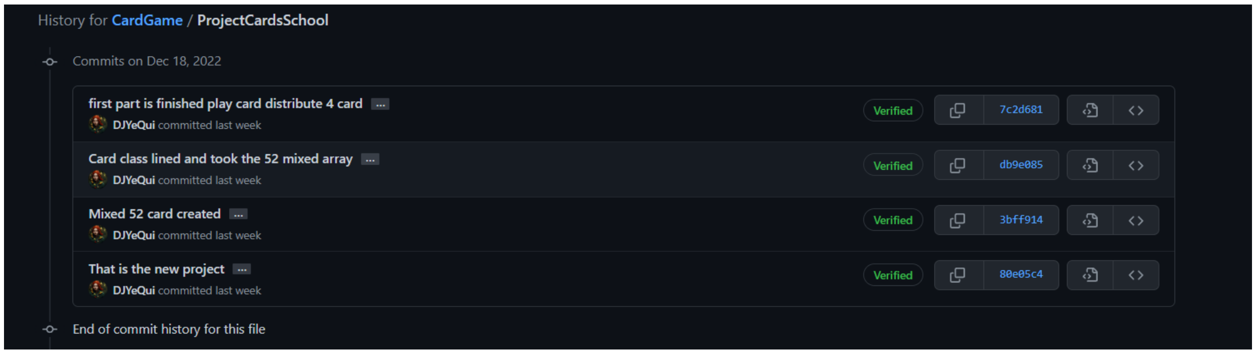
I noted all what I did
DJYeQui committed 2 weeks ago — Verified — 1e852a7

I will note all what I did
DJYeQui committed 2 weeks ago — Verified — a4711e9

Noted all codes which is finished
DJYeQui committed 2 weeks ago — Verified — eef0e4b

Project uploaded to GitHub
DJYeQui committed 2 weeks ago — Verified — 8d11cb0

End of commit history for this file


History for CardGame / ProjectCardsSchool

Commits on Dec 18, 2022

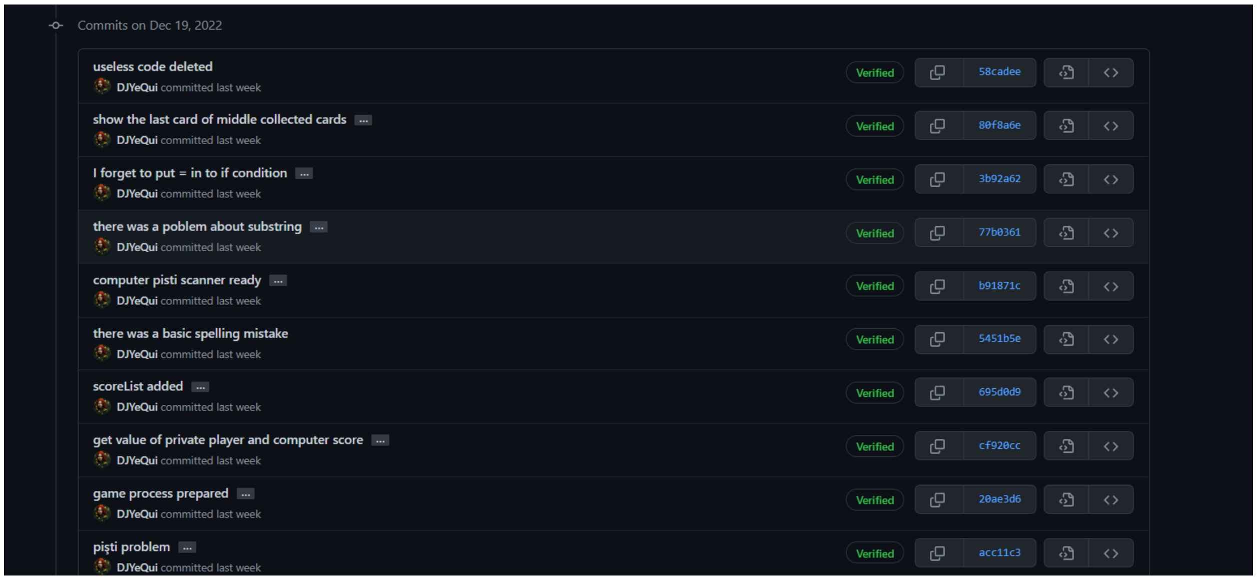
first part is finished play card distribute 4 card
DJYeQui committed last week — Verified — 7c2d681

Card class lined and took the 52 mixed array
DJYeQui committed last week — Verified — db9e085

Mixed 52 card created
DJYeQui committed last week — Verified — 3bff914

That is the new project
DJYeQui committed last week — Verified — 80e05c4

End of commit history for this file


Commits on Dec 19, 2022

useless code deleted
DJYeQui committed last week — Verified — 58cadee

show the last card of middle collected cards
DJYeQui committed last week — Verified — 80f8a6e

I forget to put = in to if condition
DJYeQui committed last week — Verified — 3b92a62

there was a poblem about substring
DJYeQui committed last week — Verified — 77b0361

computer pisti scanner ready
DJYeQui committed last week — Verified — b91871c

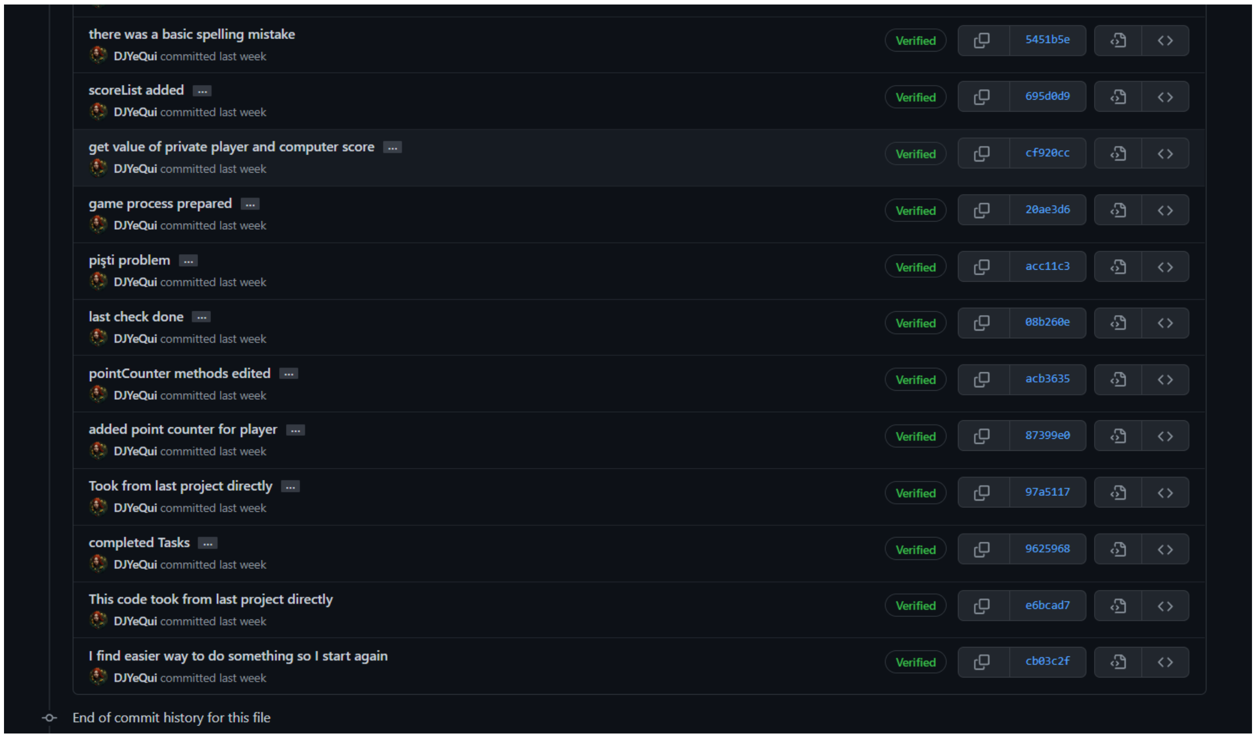
there was a basic spelling mistake
DJYeQui committed last week — Verified — 5451b5e

scoreList added
DJYeQui committed last week — Verified — 695d0d9

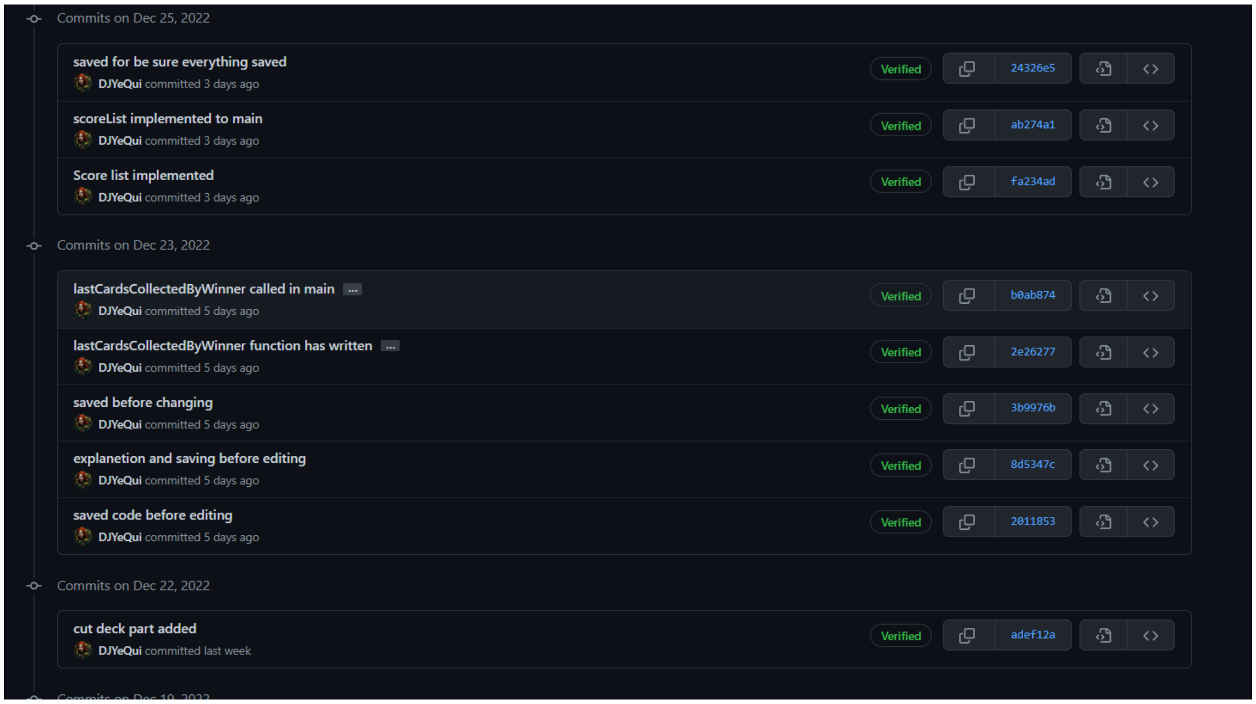
get value of private player and computer score
DJYeQui committed last week — Verified — cf920cc

game process prepared
DJYeQui committed last week — Verified — 20ae3d6

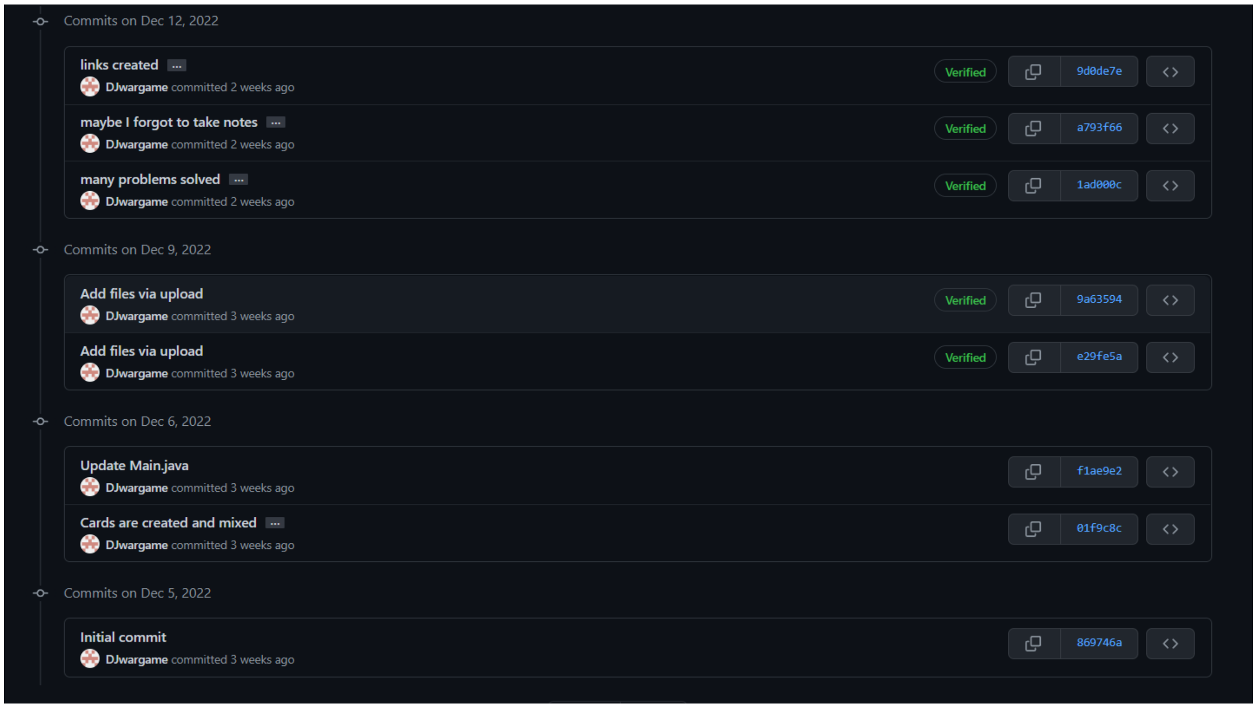
pişti problem
DJYeQui committed last week — Verified — acc11c3



```
there was a basic spelling mistake
DJYeQui committed last week                                    Verified   5451b5e
scoreList added ...
DJYeQui committed last week                                    Verified   695d0d9
get value of private player and computer score ...
DJYeQui committed last week                                    Verified   cf920cc
game process prepared ...
DJYeQui committed last week                                    Verified   20ae3d6
pişti problem ...
DJYeQui committed last week                                    Verified   acc11c3
last check done ...
DJYeQui committed last week                                    Verified   08b260e
pointCounter methods edited ...
DJYeQui committed last week                                    Verified   acb3635
added point counter for player ...
DJYeQui committed last week                                    Verified   87399e0
Took from last project directly ...
DJYeQui committed last week                                    Verified   97a5117
completed Tasks ...
DJYeQui committed last week                                    Verified   9625968
This code took from last project directly
DJYeQui committed last week                                    Verified   e6bcad7
I find easier way to do something so I start again
DJYeQui committed last week                                    Verified   cb03c2f
End of commit history for this file
```



```
Commits on Dec 25, 2022
saved for be sure everything saved
DJYeQui committed 3 days ago                                   Verified   24326e5
scoreList implemented to main
DJYeQui committed 3 days ago                                   Verified   ab274a1
Score list implemented
DJYeQui committed 3 days ago                                   Verified   fa234ad
Commits on Dec 23, 2022
lastCardsCollectedByWinner called in main ...
DJYeQui committed 5 days ago                                   Verified   b0ab874
lastCardsCollectedByWinner function has written ...
DJYeQui committed 5 days ago                                   Verified   2e26277
saved before changing
DJYeQui committed 5 days ago                                   Verified   3b9976b
explanetion and saving before editing
DJYeQui committed 5 days ago                                   Verified   8d5347c
saved code before editing
DJYeQui committed 5 days ago                                   Verified   2011853
Commits on Dec 22, 2022
cut deck part added
DJYeQui committed last week                                    Verified   adef12a
```



Commits on Dec 12, 2022

links created ...
DJwargame committed 2 weeks ago — Verified — 9d0de7e

maybe I forgot to take notes ...
DJwargame committed 2 weeks ago — Verified — a793f66

many problems solved ...
DJwargame committed 2 weeks ago — Verified — 1ad000c

Commits on Dec 9, 2022

Add files via upload
DJwargame committed 3 weeks ago — Verified — 9a63594

Add files via upload
DJwargame committed 3 weeks ago — Verified — e29fe5a

Commits on Dec 6, 2022

Update Main.java
DJwargame committed 3 weeks ago — f1ae9e2

Cards are created and mixed ...
DJwargame committed 3 weeks ago — 01f9c8c

Commits on Dec 5, 2022

Initial commit
DJwargame committed 3 weeks ago — 869746a

I had 2 GitHub profile, fatihanamasli@gmail.com and anamaslifatih@gmail.com I sign up with these emails. I changed my profile because I got the GitHub Student Pack while I was developing this project. ***Also I wrote this project 4 times. So I shared my 4 projects' histories.***

Found but unsolved error: *If you make 2 Input mistakes (type and interval) in row, next try will give you error. if you enter the right value on the 4th try, you will not see an error.*

(PLS dont break points from this situation)