FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

A PROJECT REPORT ON
“Multi-Player Tombola (Bingo) Game”

SUBMITTED BY

NAME	:	EMRE HENDEMOĞLU
ID	:	2121221037
COURSE	:	DATA STRUCTURES

IMPLEMENTATION DETAILS

- WHAT IS THE BINGO GAME?

Bingo is a game often played in large groups, primarily for entertainment purposes. The game's origins trace back to Italy and have spread over time to other parts of Europe and various regions around the world. The objective of the game is for players to match the numbers on their cards with the numbers drawn, completing a specific pattern.

Game Equipment and Setup Bingo is typically played with a bingo set, consisting of bingo cards, numbered balls or another mechanism to display numbers, and a bag or a drawing machine for the balls.

Bingo Cards: Each card usually features a grid with 3 rows and 9 columns. Each row generally contains 5 random numbers, and the remaining spaces are blank.

Numbered Balls: Balls numbered from 1 to 90 are kept in a bag or machine.

How to Play

Starting: Each player receives one or more bingo cards.

Drawing Numbers: At the start of the game, numbered balls are drawn at random. This can be done manually by an official or through an automatic machine.

Marking Numbers: When a number is drawn, players mark this number on their card(s). In modern online bingo games, this process is often automatic.

Winning: Players win by completing a certain pattern on their cards (usually one line, two lines, and then a full house) and shout "Bingo!" The game typically ends when a full card is completed.

- ## Features of the Project

Dynamic Card Generation: The game has a mechanism that can generate unique Tombola cards for each player. These cards contain a combination of random numbers.

Automatic Number Drawing: The game automatically draws numbers at set intervals and notifies the players of these numbers. The number drawing is completely random, making each game unique.

Cinko and Tombola Checking: Players can mark off numbers on their cards, and the system automatically checks for Cinko (completing a row) and Tombola (completing the entire card) conditions.

- ## Classes in bingo game class codes

TombalaCardCreator Class:

This class is responsible for generating random Tombala cards for each player. The cards consist of 3 rows and 9 columns, with each row containing 5 random numbers and 4 blanks (represented as "X"). The generateTombalaCard() method is used to create these cards. Each column is assigned a specific range of numbers (e.g., the first column contains a number between 1-10), ensuring the uniqueness of the cards.

Main Class:

This class manages the main flow of the game. Random cards are generated for two players, followed by a series of random number drawings throughout the game. After each draw, the status of the drawn number on each player's card is updated. Players' cinkos (lines) and the tombala (bingo) status are checked. The game ends when a player achieves tombala.

MultiLinkedList Class:

This class defines the data structure used for marking drawn numbers on the cards and checking the game status. It employs a multi-layered linked list structure to represent each number and blank ("X") on the card as nodes. This flexible structure allows for efficient operations on the card (e.g., marking a number).

Node Class:

This class defines the nodes used in the MultiLinkedList structure. Each node contains a value (value), a link to the next node (next), and a link to the node below (down). This is utilized to represent the relationships between rows and columns on the card.

What I learned - What I had difficulties with

- What I learned:

Multi-Layer Linked Lists: You learned how to represent a two-dimensional data structure in one dimension using the MultiLinkedList structure. This has improved your ability to understand and apply complex data structures.

Adding Links to Data Structures: You learned how to relate nodes to each other by adding horizontal (next) and vertical (down) links. This has increased your abilities to establish and manage relationships between data structures.

Control and Update Operations: You learned how to perform search and update operations on data structures by performing operations such as checking and updating the drawn numbers on the card.

Problem Solving and Algorithm Development: By developing algorithms to meet the requirements of the project, you have developed your problem-solving skills and algorithmic thinking ability.

- ## What I had difficulties with:

Complexity of Data Structures: Representing a two-dimensional data structure with a linked list may seem complicated at first, especially establishing the connections correctly. It takes time and practice to grasp this concept.

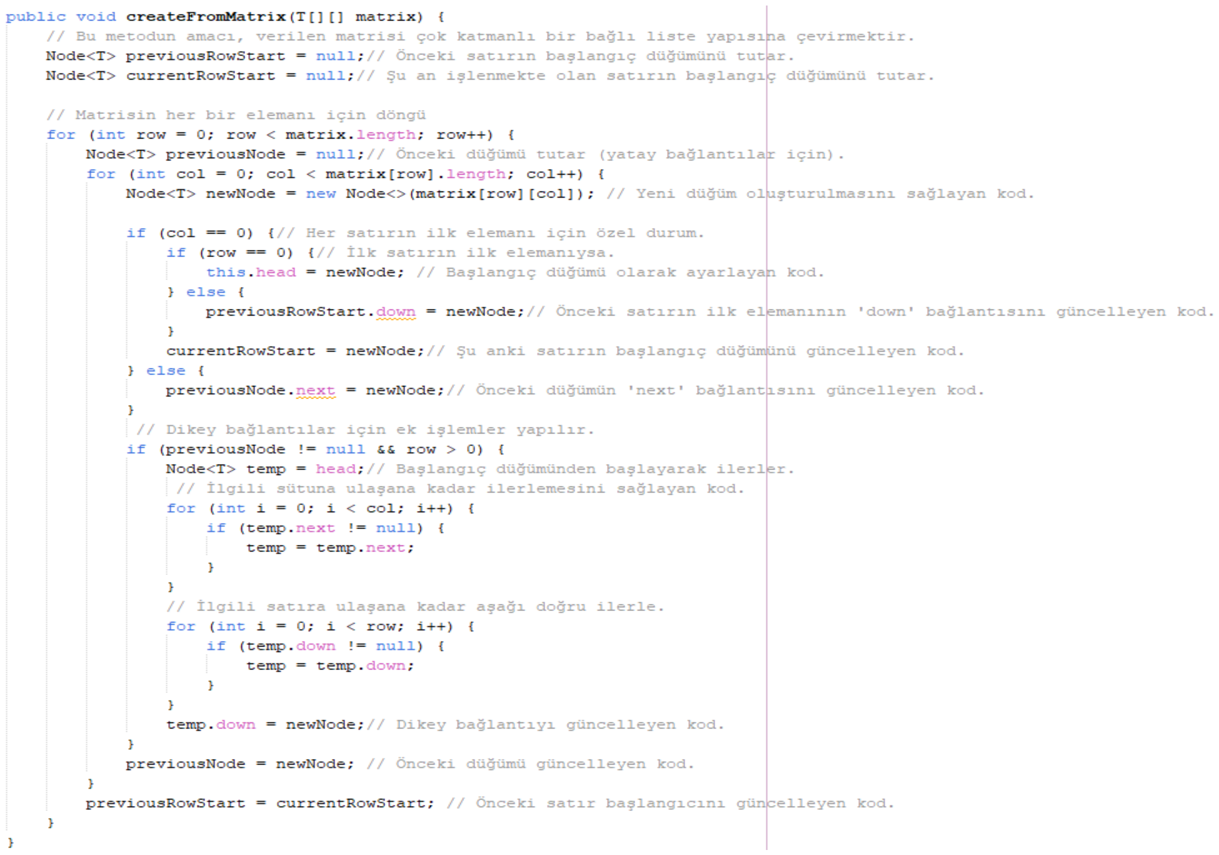
Debugging: When working with multi-layer linked lists, it is easy to make errors in inter-node connections. Finding and fixing these errors can be difficult, especially as the size of the project increases.

Algorithm Optimization: In some cases, finding and implementing the most effective algorithm can be challenging. Particularly in performance-critical situations, discovering and implementing more efficient solutions can be difficult.

Project Management: As you progress through the project, it can be difficult to maintain the order and readability of your code, especially as you add multiple components and features. You need a good project structure and coding practice to keep your code organized and manageable.

Design

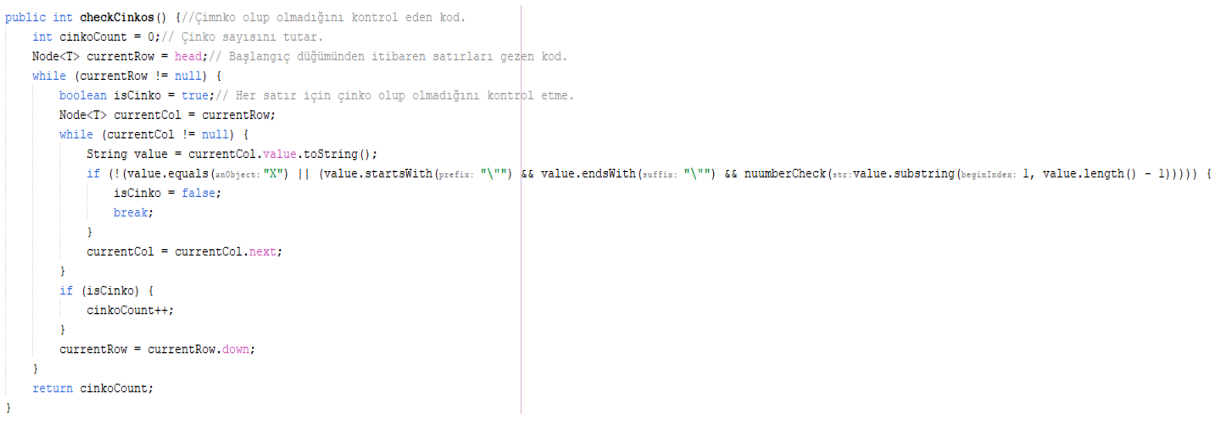
• Multi-Linked List



```java
public void createFromMatrix(T[][] matrix) {
    // Bu metodun amacı, verilen matrisi çok katmanlı bir bağlı liste yapısına çevirmektir.
    Node<T> previousRowStart = null;// Önceki satırın başlangıç düğümünü tutar.
    Node<T> currentRowStart = null;// Şu an işlenmekte olan satırın başlangıç düğümünü tutar.

    // Matrisin her bir elemanı için döngü
    for (int row = 0; row < matrix.length; row++) {
        Node<T> previousNode = null;// Önceki düğümü tutar (yatay bağlantılar için).
        for (int col = 0; col < matrix[row].length; col++) {
            Node<T> newNode = new Node<>(matrix[row][col]); // Yeni düğüm oluşturulmasını sağlayan kod.

            if (col == 0) {// Her satırın ilk elemanı için özel durum.
                if (row == 0) {// İlk satırın ilk elemanıysa.
                    this.head = newNode; // Başlangıç düğümü olarak ayarlayan kod.
                } else {
                    previousRowStart.down = newNode;// Önceki satırın ilk elemanının 'down' bağlantısını güncelleyen kod.
                }
                currentRowStart = newNode;// Şu anki satırın başlangıç düğümünü güncelleyen kod.
            } else {
                previousNode.next = newNode;// Önceki düğümün 'next' bağlantısını güncelleyen kod.
            }
             // Dikey bağlantılar için ek işlemler yapılır.
            if (previousNode != null && row > 0) {
                Node<T> temp = head;// Başlangıç düğümünden başlayarak ilerler.
                 // İlgili sütuna ulaşana kadar ilerlemesini sağlayan kod.
                for (int i = 0; i < col; i++) {
                    if (temp.next != null) {
                        temp = temp.next;
                    }
                }
                // İlgili satıra ulaşana kadar aşağı doğru ilerle.
                for (int i = 0; i < row; i++) {
                    if (temp.down != null) {
                        temp = temp.down;
                    }
                }
                temp.down = newNode;// Dikey bağlantıyı güncelleyen kod.
            }
            previousNode = newNode; // Önceki düğümü güncelleyen kod.
        }
        previousRowStart = currentRowStart; // Önceki satır başlangıcını güncelleyen kod.
    }
}
```

#1.1 This code describes a method for converting a given matrix into a multi-layered linked list structure. This structure treats each element of the matrix as a node, where nodes are connected both horizontally (elements within a row) and vertically (across columns), forming a list

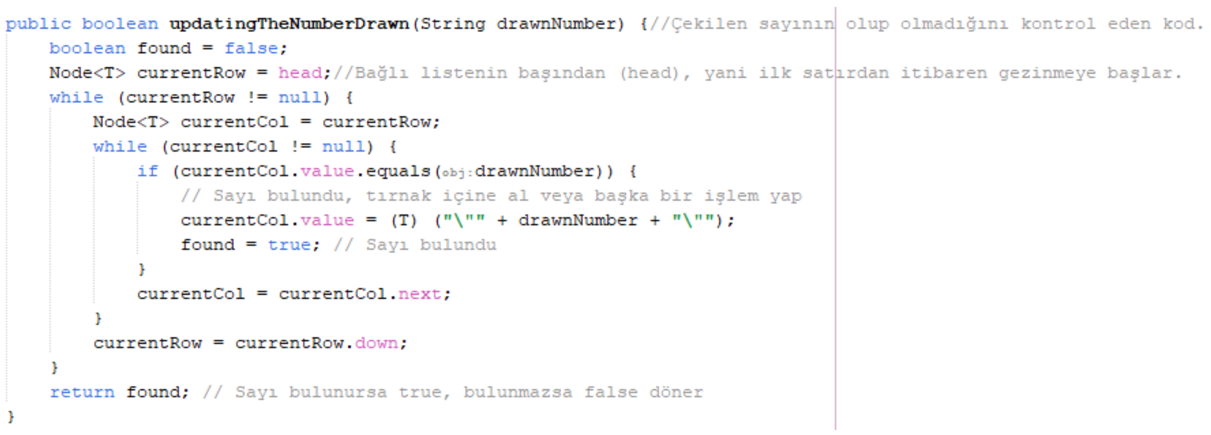
---***----------------------------------***---------------------------------



```java
public int checkCinkos() {//Çimnko olup olmadığını kontrol eden kod.
    int cinkoCount = 0;// Çinko sayısını tutar.
    Node<T> currentRow = head;// Başlangıç düğümünden itibaren satırları gezen kod.
    while (currentRow != null) {
        boolean isCinko = true;// Her satır için çinko olup olmadığını kontrol etme.
        Node<T> currentCol = currentRow;
        while (currentCol != null) {
            String value = currentCol.value.toString();
            if (!(value.equals(anObject: "X") || (value.startsWith(prefix: "\"") && value.endsWith(suffix: "\"") && nuumberCheck(str:value.substring(beginIndex: 1, value.length() - 1))))) {
                isCinko = false;
                break;
            }
            currentCol = currentCol.next;
        }
        if (isCinko) {
            cinkoCount++;
        }
        currentRow = currentRow.down;
    }
    return cinkoCount;
}
```

#1.2 checkCinkos Method: This method checks for rows in a bingo-like game that have been completed, known as "cinkos." It iterates through each row of the linked list, examining if all values are marked (denoted by "X" or surrounded by quotes which could represent a drawn number that's been marked).

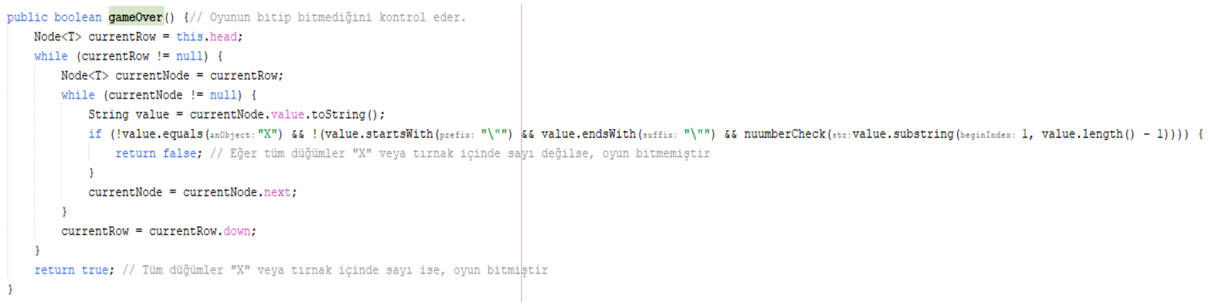
---***----------------------------------***---------------------------------



```java
public boolean updatingTheNumberDrawn(String drawnNumber) {//Çekilen sayının olup olmadığını kontrol eden kod.
    boolean found = false;
    Node<T> currentRow = head;//Bağlı listenin başından (head), yani ilk satırdan itibaren gezinmeye başlar.
    while (currentRow != null) {
        Node<T> currentCol = currentRow;
        while (currentCol != null) {
            if (currentCol.value.equals(obj:drawnNumber)) {
                // Sayı bulundu, tırnak içine al veya başka bir işlem yap
                currentCol.value = (T) ("\"" + drawnNumber + "\"");
                found = true; // Sayı bulundu
            }
            currentCol = currentCol.next;
        }
        currentRow = currentRow.down;
    }
    return found; // Sayı bulunursa true, bulunmazsa false döner
}
```

#1.3updatingTheNumberDrawn Method: This function searches the linked list for a specified number (the one recently drawn in the game). Starting from the beginning of the list, it traverses each row and column. If the drawn number is found, it marks the number by enclosing it in quotes (to indicate that this number has been drawn and acknowledged).

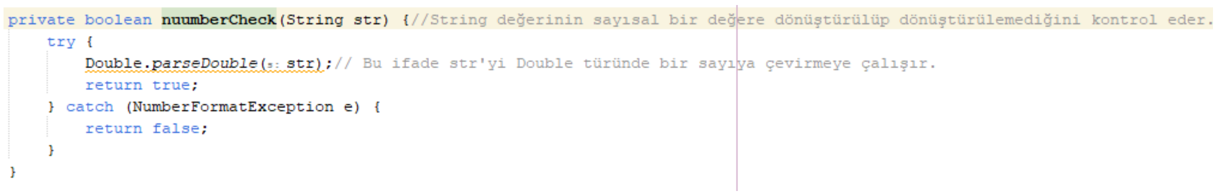
---***----------------------------------***---------------------------------



```java
public boolean gameOver() {// Oyunun bitip bitmediğini kontrol eder.
    Node<T> currentRow = this.head;
    while (currentRow != null) {
        Node<T> currentNode = currentRow;
        while (currentNode != null) {
            String value = currentNode.value.toString();
            if (!value.equals(anObject: "X") && !(value.startsWith(prefix: "\"") && value.endsWith(suffix: "\"") && nuumberCheck(str:value.substring(beginIndex: 1, value.length() - 1)))) {
                return false; // Eğer tüm düğümler "X" veya tırnak içinde sayı değilse, oyun bitmemiştir
            }
            currentNode = currentNode.next;
        }
        currentRow = currentRow.down;
    }
    return true; // Tüm düğümler "X" veya tırnak içinde sayı ise, oyun bitmiştir
}
```

#1.4 gameOver Method: This method assesses whether a game (similar to bingo) has concluded by checking if all items in a data structure (like a board) are marked as "X" or are numbers encapsulated in quotes.

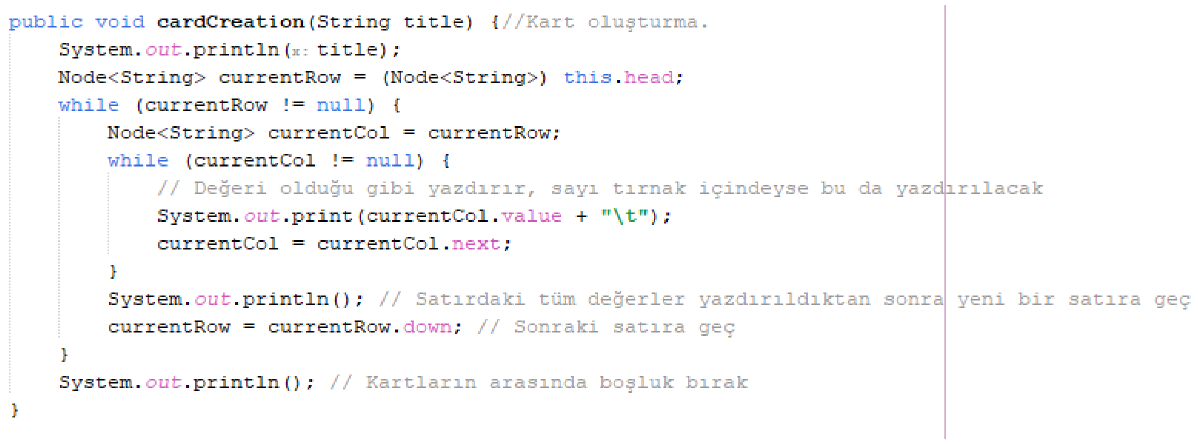
---***----------------------------------***---------------------------------



```java
private boolean nuumberCheck(String str) {//String değerinin sayısal bir değere dönüştürülüp dönüştürülemediğini kontrol eder.
    try {
        Double.parseDouble(s: str);// Bu ifade str'yi Double türünde bir sayıya çevirmeye çalışır.
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}
```

#1.5nuumberCheck Method: This auxiliary function validates whether a given string can be converted into a numerical value, specifically a Double. It attempts to parse the string into a Double, and if successful, returns true, indicating the string represents a valid number.

--***---------------------------------***--------------------------------

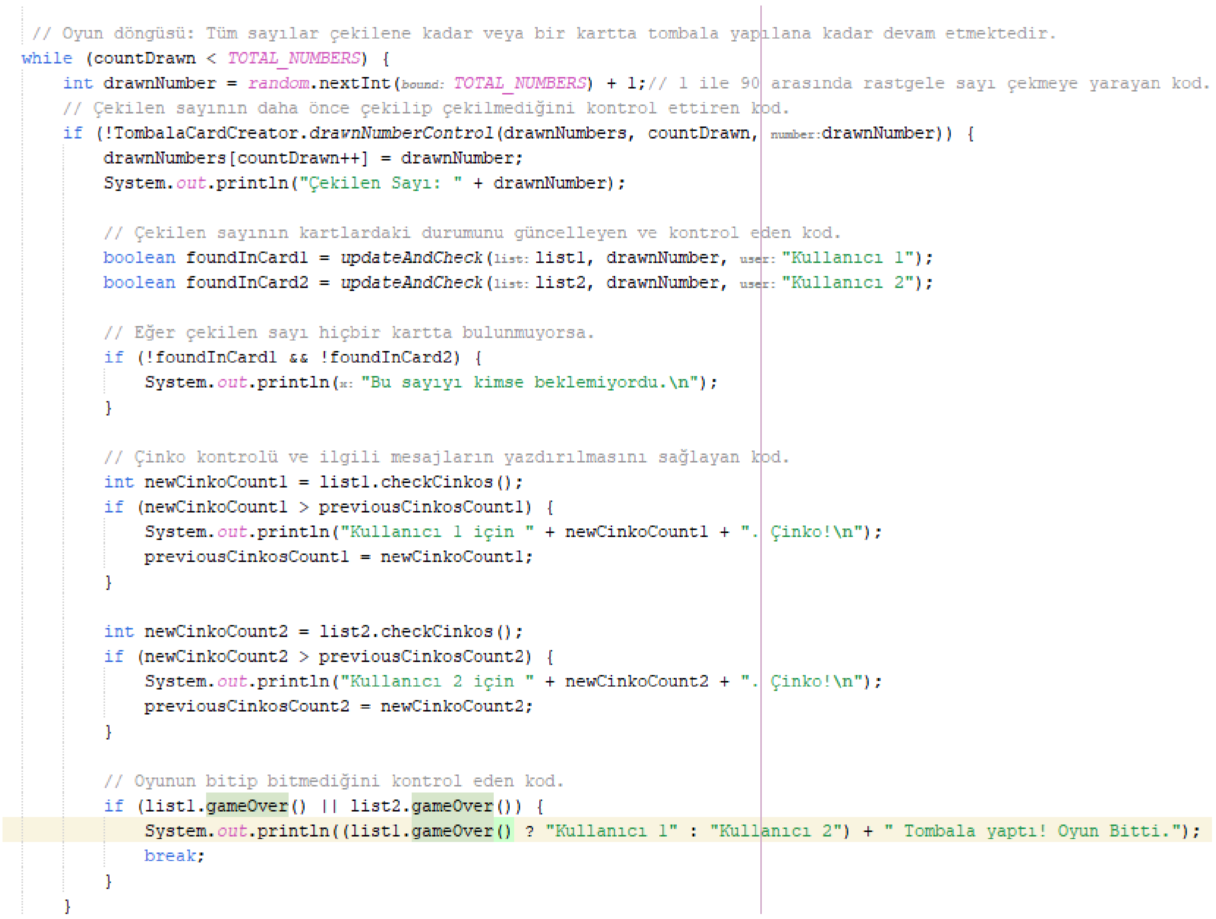


```java
public void cardCreation(String title) {//Kart oluşturma.
    System.out.println(x: title);
    Node<String> currentRow = (Node<String>) this.head;
    while (currentRow != null) {
        Node<String> currentCol = currentRow;
        while (currentCol != null) {
            // Değeri olduğu gibi yazdırır, sayı tırnak içindeyse bu da yazdırılacak
            System.out.print(currentCol.value + "\t");
            currentCol = currentCol.next;
        }
        System.out.println(); // Satırdaki tüm değerler yazdırıldıktan sonra yeni bir satıra geç
        currentRow = currentRow.down; // Sonraki satıra geç
    }
    System.out.println(); // Kartların arasında boşluk bırak
}
```

#1.6 cardCreation Method: This code creates a "card" using the multilayer linked list and prints that card to the console. Each row and column represents a node of the connected listen. These nodes will be joined both Extreme (next) and vertical (down). After writing the title, the function loops through each node in each row and writes its values.

Game

- Main:



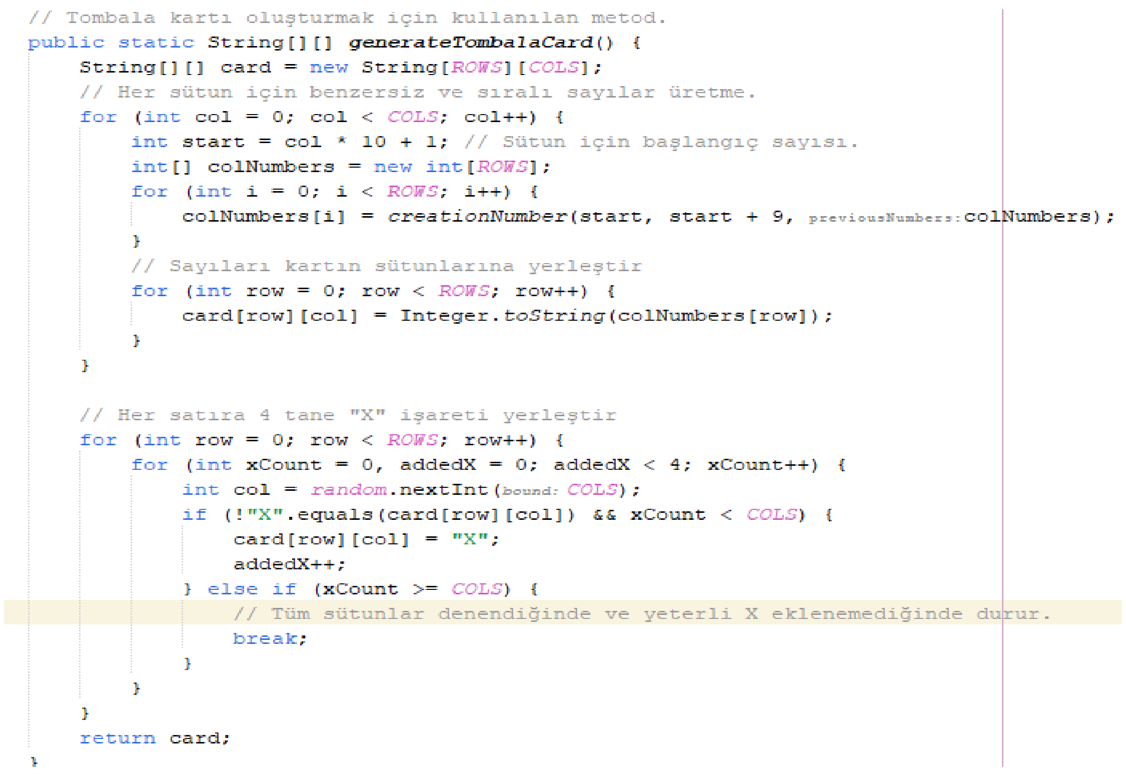
```
// Oyun döngüsü: Tüm sayılar çekilene kadar veya bir kartta tombala yapılana kadar devam etmektedir.
while (countDrawn < TOTAL_NUMBERS) {
    int drawnNumber = random.nextInt(bound: TOTAL_NUMBERS) + 1;// 1 ile 90 arasında rastgele sayı çekmeye yarayan kod.
    // Çekilen sayının daha önce çekilip çekilmediğini kontrol ettiren kod.
    if (!TombalaCardCreator.drawnNumberControl(drawnNumbers, countDrawn, number:drawnNumber)) {
        drawnNumbers[countDrawn++] = drawnNumber;
        System.out.println("Çekilen Sayı: " + drawnNumber);

        // Çekilen sayının kartlardaki durumunu güncelleyen ve kontrol eden kod.
        boolean foundInCard1 = updateAndCheck(list: list1, drawnNumber, user: "Kullanıcı 1");
        boolean foundInCard2 = updateAndCheck(list: list2, drawnNumber, user: "Kullanıcı 2");

        // Eğer çekilen sayı hiçbir kartta bulunmuyorsa.
        if (!foundInCard1 && !foundInCard2) {
            System.out.println(x: "Bu sayıyı kimse beklemiyordu.\n");
        }

        // Çinko kontrolü ve ilgili mesajların yazdırılmasını sağlayan kod.
        int newCinkoCount1 = list1.checkCinkos();
        if (newCinkoCount1 > previousCinkosCount1) {
            System.out.println("Kullanıcı 1 için " + newCinkoCount1 + ". Çinko!\n");
            previousCinkosCount1 = newCinkoCount1;
        }

        int newCinkoCount2 = list2.checkCinkos();
        if (newCinkoCount2 > previousCinkosCount2) {
            System.out.println("Kullanıcı 2 için " + newCinkoCount2 + ". Çinko!\n");
            previousCinkosCount2 = newCinkoCount2;
        }

        // Oyunun bitip bitmediğini kontrol eden kod.
        if (list1.gameOver() || list2.gameOver()) {
            System.out.println((list1.gameOver() ? "Kullanıcı 1" : "Kullanıcı 2") + " Tombala yaptı! Oyun Bitti.");
            break;
        }
    }
```

# 2.1This code snippet orchestrates the main game loop for a bingo-like game, where numbers are randomly drawn one at a time, checked against two players' cards for matches, and marked if found.

The loop continues until all possible numbers are drawn or when a player achieves a bingo, ending the game

It incorporates mechanisms to avoid repeating numbers, updates each player's card with the drawn number, checks for winning conditions, and announces the game's conclusion along with the winner.

- TombalaCardCreator:



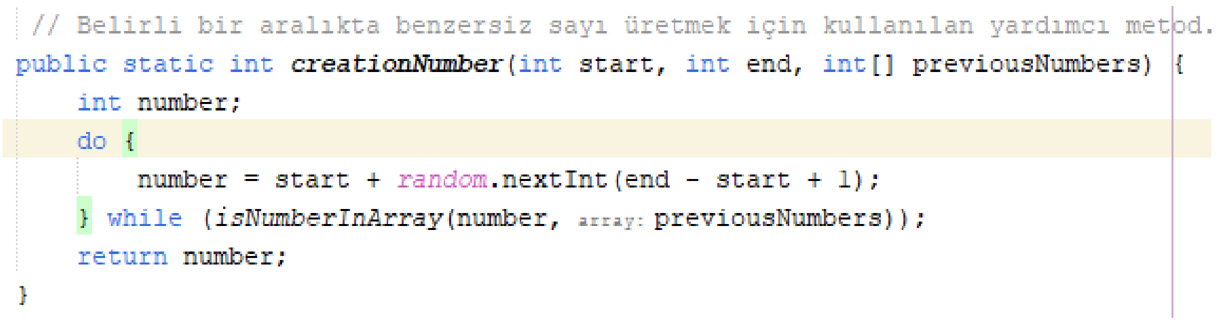
```
// Tombala kartı oluşturmak için kullanılan metod.
public static String[][] generateTombalaCard() {
    String[][] card = new String[ROWS][COLS];
    // Her sütun için benzersiz ve sıralı sayılar üretme.
    for (int col = 0; col < COLS; col++) {
        int start = col * 10 + 1; // Sütun için başlangıç sayısı.
        int[] colNumbers = new int[ROWS];
        for (int i = 0; i < ROWS; i++) {
            colNumbers[i] = creationNumber(start, start + 9, previousNumbers: colNumbers);
        }
        // Sayıları kartın sütunlarına yerleştir
        for (int row = 0; row < ROWS; row++) {
            card[row][col] = Integer.toString(colNumbers[row]);
        }
    }

    // Her satıra 4 tane "X" işareti yerleştir
    for (int row = 0; row < ROWS; row++) {
        for (int xCount = 0, addedX = 0; addedX < 4; xCount++) {
            int col = random.nextInt( bound: COLS);
            if (!"X".equals(card[row][col]) && xCount < COLS) {
                card[row][col] = "X";
                addedX++;
            } else if (xCount >= COLS) {
                // Tüm sütunlar denendiğinde ve yeterli X eklenemediğinde durur.
                break;
            }
        }
    }
    return card;
}
```

#3.1 generateTombalaCard Method: Each card contains a designated number of rows (ROWS) and columns (COLS). In this example the card has 3 rows and 9 columns.

- Generating Unique and Sequential Numbers: For each column, unique numbers are selected from a range of numbers specific to that column. For example, numbers 1 to 10 can be selected for the first column and 11 to 20 for the second column. This process is done using the creationNumber method. This method produces a number within the specified range (start and start + 9) that is different from the previous selected numbers.
- Placing Numbers on the Card: The produced numbers are placed in the relevant columns of the card. Numbers are stored by converting them to String type because the card matrix (String[][] card) is defined as String type.
- Placement of "X" Marks: Some fields are usually left blank on bingo cards. These empty areas are as important as the filled areas during the game. For each row, an "X" mark is placed in 4 randomly selected columns, indicating that that field is empty. Using random.nextInt(COLS) a random column is selected and an "X" is placed in that column. If a column already contains "X" or the designated number of "X" (4 in this example) is placed for each row, this step is skipped or the loop ends.

This process explains how to create a Bingo card with both numbers and blank spaces ("X" signs). The result is a Bingo card ready to be used during the game.

--***--------------------------------***-------------------------------



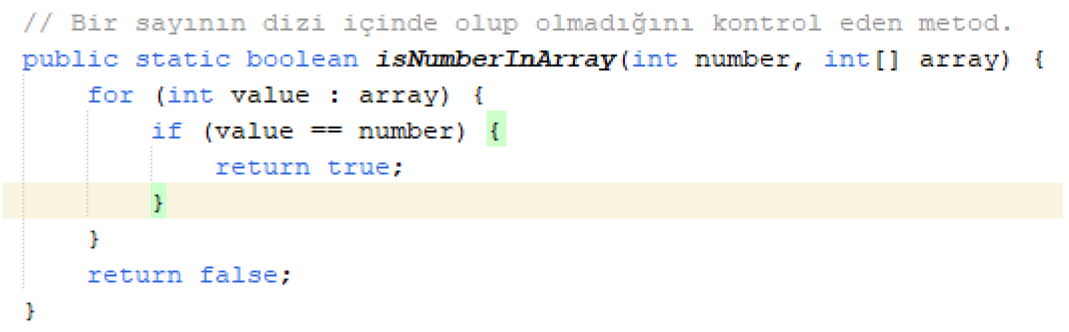
```java
// Belirli bir aralıkta benzersiz sayı üretmek için kullanılan yardımcı metod.
public static int creationNumber(int start, int end, int[] previousNumbers) {
    int number;
    do {
        number = start + random.nextInt(end - start + 1);
    } while (isNumberInArray(number, array: previousNumbers));
    return number;
}
```

#3.2 creationNumber Method : This piece of code is used to generate a number within a specified range (between start and end) and different from previously selected numbers (previousNumbers array). The purpose of the function is to generate random and unique numbers for columns on a Bingo card.

- number = start + random.nextInt(end - start + 1); line generates a random number in the range start and end. The random.nextInt(n) method generates a random integer from 0 (inclusive) to n-1 (not included). So, with the expression end - start + 1, the range is expanded to include all possible numbers between start and end.
- Calling the isNumberInArray(number, previousNumbers) method checks whether number is in the previousNumbers array. Returns true if the number is present in the array, false otherwise.

As a result, this method is used to generate a random number within a certain range, which differs from the pre-selected numbers for each column. In this way, each column of the Bingo card contains unique and range-compliant numbers.

--***--------------------------------***-------------------------------



```java
// Bir sayının dizi içinde olup olmadığını kontrol eden metod.
public static boolean isNumberInArray(int number, int[] array) {
    for (int value : array) {
        if (value == number) {
            return true;
        }
    }
    return false;
}
```

#3.3 isNumberInArray Method: This piece of code checks whether a number exists in an array. It returns true if the searched number is in the array, otherwise false. This functionality is used to query the existence of a particular value in an array.

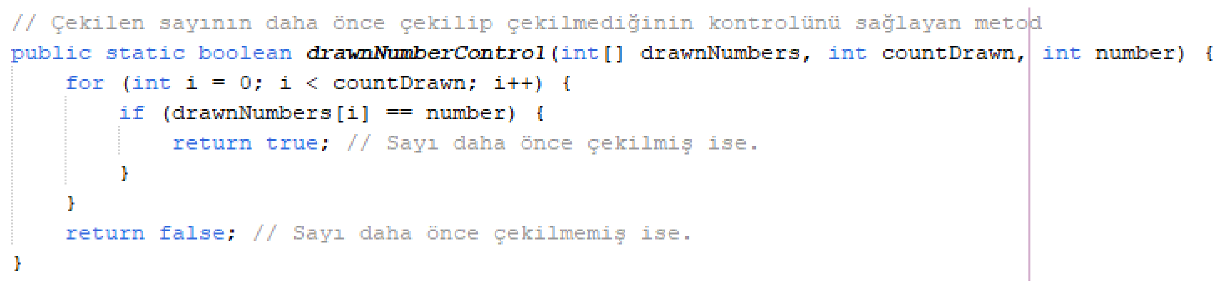
- The for (int value : array) { ... } loop loops through each element (value) in the array named array one by one.
- The if (value == number) { ... } statement compares each value considered during the loop with the searched value number.

- If value and number are equal, that is, if the searched number is found in the array, the method returns true and the search ends.

When the loop completes and the number is not found in the array, the method returns false, indicating that the searched number is not in the array.

This method is often used to check whether a number has been selected before.

---***----------------------------------***----------------------------------

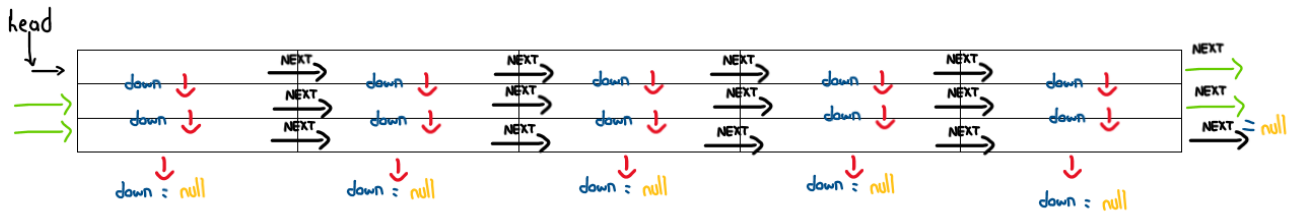


```java
// Çekilen sayının daha önce çekilip çekilmediğinin kontrolünü sağlayan metod
public static boolean drawnNumberControl(int[] drawnNumbers, int countDrawn, int number) {
    for (int i = 0; i < countDrawn; i++) {
        if (drawnNumbers[i] == number) {
            return true; // Sayı daha önce çekilmiş ise.
        }
    }
    return false; // Sayı daha önce çekilmemiş ise.
}
```

#3.4 drawnNumberControl Method : This piece of code contains the definition of a method that checks whether a number (number) has been drawn before. In a bingo game or similar situation, there is this method of knowing whether a certain number has been drawn or not.

- The DrawNumbers series stores parts that are detailed up to that point.
- countDrawn specifies how many numbers are stored in the drawnNumbers array.
- number is the number to be checked.

DESIGN



The MultiLinkedList structure represents each element of a Bingo card and the relationships of these elements to each other. In this structure, each node symbolizes a box on the Bingo card. Nodes can be connected to each other in two different directions, both horizontally and vertically. Horizontal links connect boxes in a row, while vertical links connect boxes one under the other in the same column. In this way, the two-dimensional structure of a Bingo card can be represented in one dimension using a multi-layer linked list.

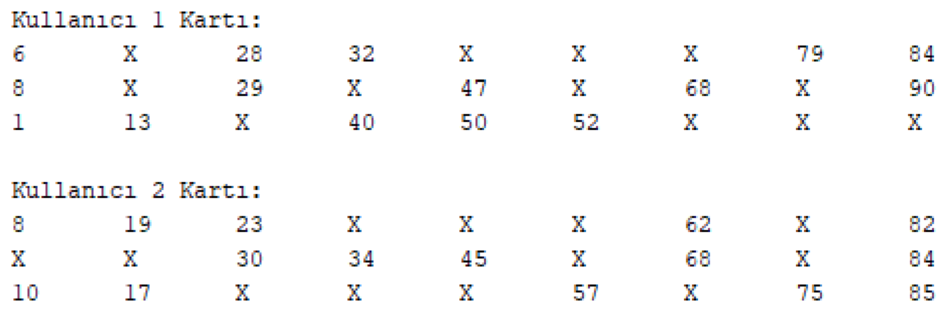
Each Node class has two main properties: value and connections (next and down). The value property holds the data that the node stores (in this case, a number or an "X" sign).

The next property refers to the link to the next node in the same row, and the down attribute refers to the link to the node below in the same column.

Here are some basic functions in the MultiLinkedList structure:

- createFromMatrix(T[][] matrix): This method creates a multilayer linked list from a given matrix (Bingo card). Each matrix element is converted into a Node node. Horizontal and vertical links are adjusted based on their position in the matrix.
- checkCinkos(): This method checks each line on the card and calculates the number of lines (zincs) completely filled with the "X" sign.
- updatingTheNumberDrawn(String drawnNumber): This method finds the equivalent of a drawn number on the card and updates it by replacing the found number with "X". This method returns true if the number is present on the card.
- gameOver(): This method checks whether the game is over or not. If all numbers on the card have been replaced with "X" (i.e. bingo has been made), the value true is returned.
- cardCreation(String title): This method prints the current status of the card on the screen. It cycles through each row and column of the card, showing the values (numbers and "X" signs) within it.

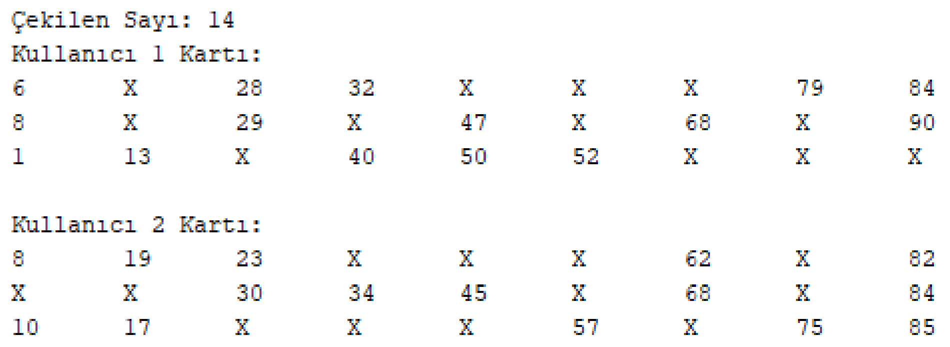
AN EXEMPLARY SCENARIO



```
Kullanıcı 1 Kartı:
6       X       28      32      X       X       X       79      84
8       X       29      X       47      X       68      X       90
1       13      X       40      50      52      X       X       X

Kullanıcı 2 Kartı:
8       19      23      X       X       X       62      X       82
X       X       30      34      45      X       68      X       84
10      17      X       X       X       57      X       75      85
```

- First, two cards are randomly created. The created card consists of 3 rows and 9 columns. Numbers 1-9 in the first column, numbers 10-19 in the second column, numbers 20-39 in the third column... Numbers 80-99 in the ninth column are randomly generated and there are 4 empty (X) boxes in each row.

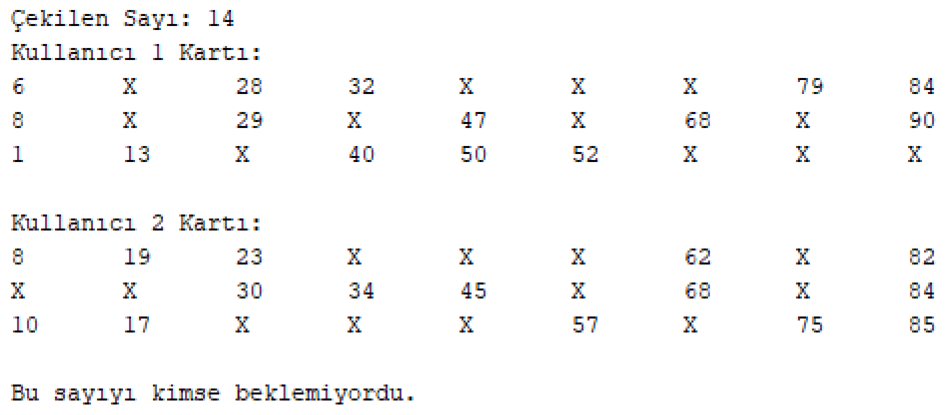
---\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 14
Kullanıcı 1 Kartı:
6       X       28      32      X       X       X       79      84
8       X       29      X       47      X       68      X       90
1       13      X       40      50      52      X       X       X

Kullanıcı 2 Kartı:
8       19      23      X       X       X       62      X       82
X       X       30      34      45      X       68      X       84
10      17      X       X       X       57      X       75      85
```

- In the second stage, random numbers are drawn and control is achieved through bingo cards. Different situations arise if the number drawn is on the card or not.
-

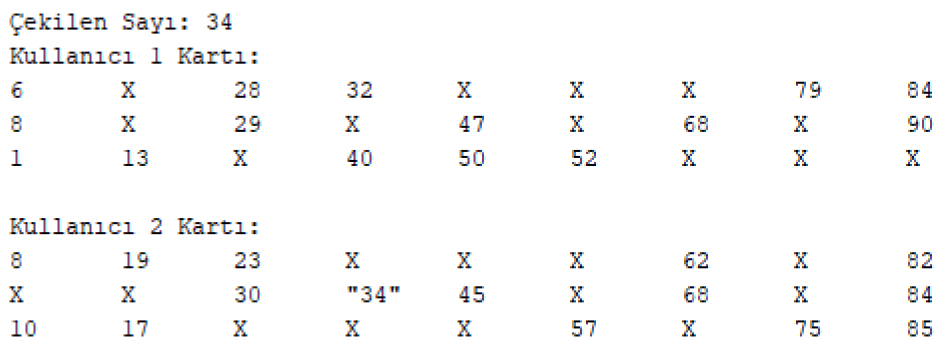
--\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 14
Kullanıcı 1 Kartı:
6       X       28      32      X       X       X       79      84
8       X       29      X       47      X       68      X       90
1       13      X       40      50      52      X       X       X

Kullanıcı 2 Kartı:
8       19      23      X       X       X       62      X       82
X       X       30      34      45      X       68      X       84
10      17      X       X       X       57      X       75      85

Bu sayıyı kimse beklemiyordu.
```

- The randomly drawn number is checked within two bingo cards. If the number drawn is not on either card, the user is warned and given a message saying "Bu sayıyı kimse beklemiyordu". This warning indicates that this number is not on either card.

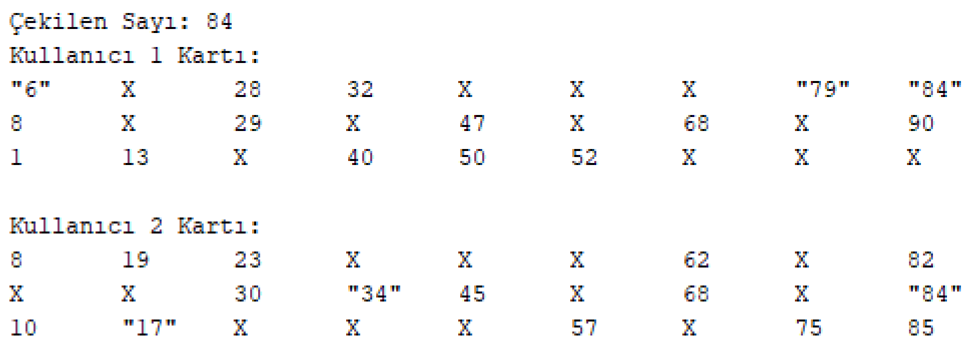
--\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 34
Kullanıcı 1 Kartı:
6        X        28       32       X        X        X        79       84
8        X        29       X        47       X        68       X        90
1        13       X        40       50       52       X        X        X

Kullanıcı 2 Kartı:
8        19       23       X        X        X        62       X        82
X        X        30       "34"     45       X        68       X        84
10       17       X        X        X        57       X        75       85
```

- Another randomly drawn number is checked again on both cards. If the drawn number is on either of the two cards, the value of the drawn number is written as "number" in its place on the card and updated. Thus, that number is now marked.

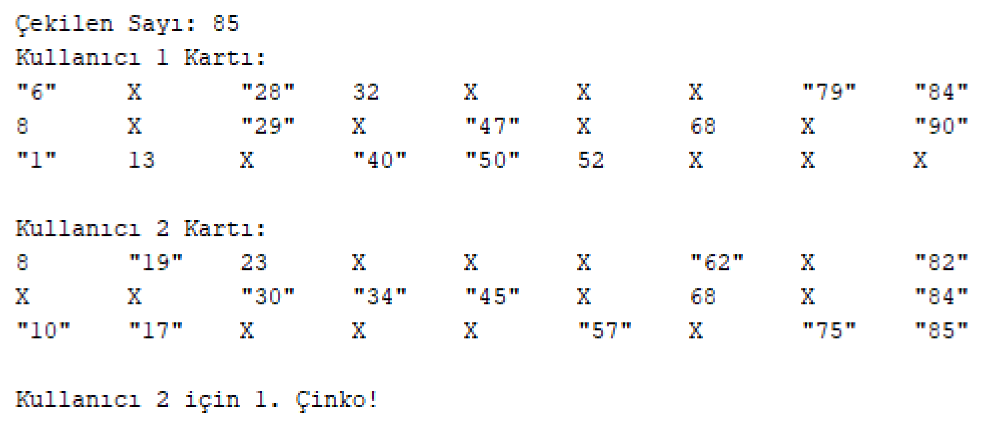
--\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 84
Kullanıcı 1 Kartı:
"6"      X        28       32       X        X        X        "79"     "84"
8        X        29       X        47       X        68       X        90
1        13       X        40       50       52       X        X        X

Kullanıcı 2 Kartı:
8        19       23       X        X        X        62       X        82
X        X        30       "34"     45       X        68       X        "84"
10       "17"     X        X        X        57       X        75       85
```

- The randomly drawn number is checked on two cards. If the drawn number is on both cards, it is written as "number" on both cards. Then, "number" is written on both cards and the cards are updated. It continues with the next card draw with the number marked on the two cards.

--\\\---------------------------------///--------------------------------

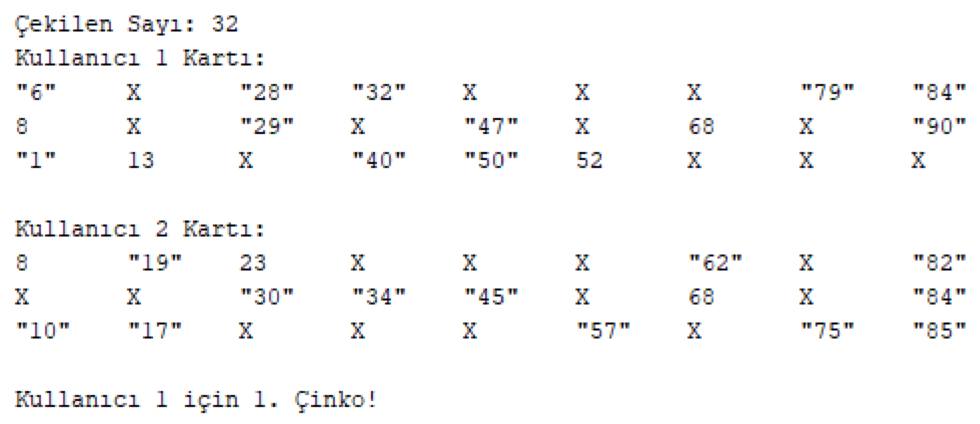


```
Çekilen Sayı: 85
Kullanıcı 1 Kartı:
"6"     X       "28"    32      X       X       X       "79"    "84"
8       X       "29"    X       "47"    X       68      X       "90"
"1"     13      X       "40"    "50"    52      X       X       X

Kullanıcı 2 Kartı:
8       "19"    23      X       X       X       "62"    X       "82"
X       X       "30"    "34"    "45"    X       68      X       "84"
"10"    "17"    X       X       X       "57"    X       "75"    "85"

Kullanıcı 2 için 1. Çinko!
```

!! (What is seen on the screen is user 2 making 1st zinc)!!

- The randomly drawn number is constantly checked for two cards, and if the randomly drawn number is on any of the cards, it is written as "number". If all of the numbers on the cards in the user's hand (card1 or card2) overlap in the form of X or "number", the user will have made the 1st zinc. and the system gives a message to the user as (1-2) 1st zinc and the program continues to run. (What is seen on the screen is user 2 making 1st zinc)

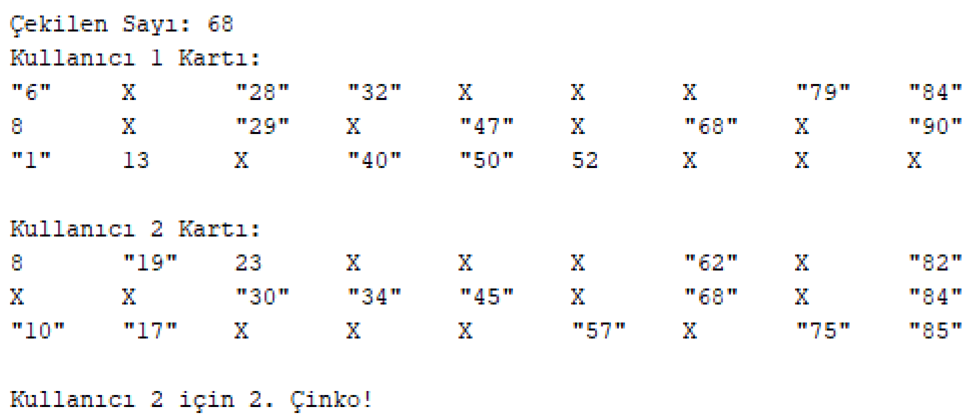
--\\\\\\-----------------------------------///--------------------------------



```
Çekilen Sayı: 32
Kullanıcı 1 Kartı:
"6"     X       "28"    "32"    X       X       X       "79"    "84"
8       X       "29"    X       "47"    X       68      X       "90"
"1"     13      X       "40"    "50"    52      X       X       X

Kullanıcı 2 Kartı:
8       "19"    23      X       X       X       "62"    X       "82"
X       X       "30"    "34"    "45"    X       68      X       "84"
"10"    "17"    X       X       X       "57"    X       "75"    "85"

Kullanıcı 1 için 1. Çinko!
```

!! In the picture, user 1 made his 1st zinc!!

- User 2's first zinc does not prevent user 1's 1st zinc (both cards may have 1st zinc).

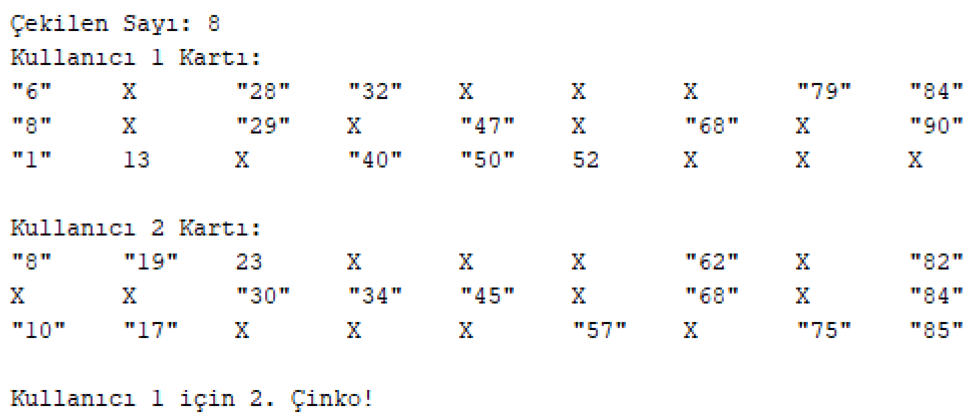


```
Çekilen Sayı: 68
Kullanıcı 1 Kartı:
"6"     X       "28"    "32"    X       X       X       "79"    "84"
8       X       "29"    X       "47"    X       "68"    X       "90"
"1"     13      X       "40"    "50"    52      X       X       X

Kullanıcı 2 Kartı:
8       "19"    23      X       X       X       "62"    X       "82"
X       X       "30"    "34"    "45"    X       "68"    X       "84"
"10"    "17"    X       X       X       "57"    X       "75"    "85"

Kullanıcı 2 için 2. Çinko!
```

!! (What is seen on the screen is user 2 making 2st zinc)!!

- The randomly drawn number is constantly checked for two cards, and if the randomly drawn number is on any of the cards, it is written as "number". If all of the numbers on the cards in the user's hand (card1 or card2) overlap in the form of X or "number", the user will have made the 2st zinc. and the system gives a message to the user as (1-2) 1st zinc and the program continues to run.

---\\\---------------------------------///--------------------------------

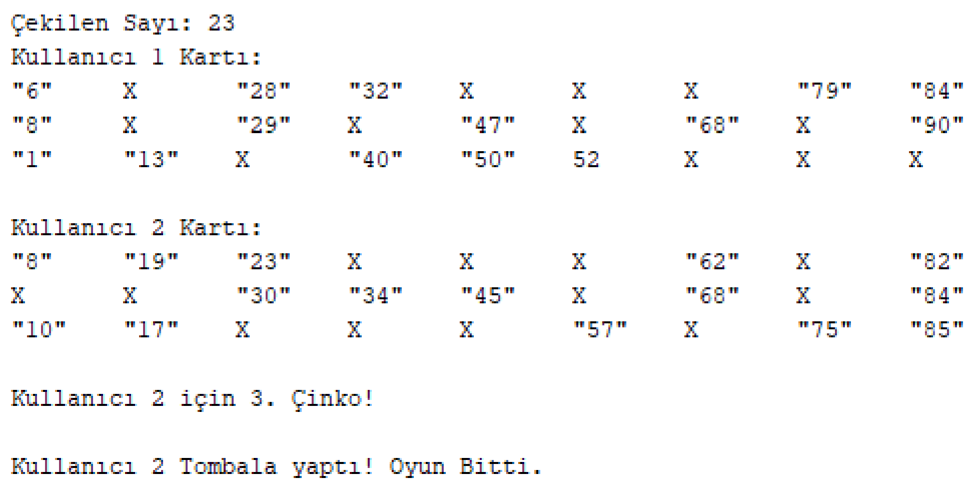


```
Çekilen Sayı: 8
Kullanıcı 1 Kartı:
"6"     X       "28"    "32"    X       X       X       "79"    "84"
"8"     X       "29"    X       "47"    X       "68"    X       "90"
"1"     13      X       "40"    "50"    52      X       X       X

Kullanıcı 2 Kartı:
"8"     "19"    23      X       X       X       "62"    X       "82"
X       X       "30"    "34"    "45"    X       "68"    X       "84"
"10"    "17"    X       X       X       "57"    X       "75"    "85"

Kullanıcı 1 için 2. Çinko!
```

!! In the picture, user 1 made his 2nd zinc!!

- User 2's second zinc does not prevent user 1's 2nd zinc (both cards may have 2nd zinc).

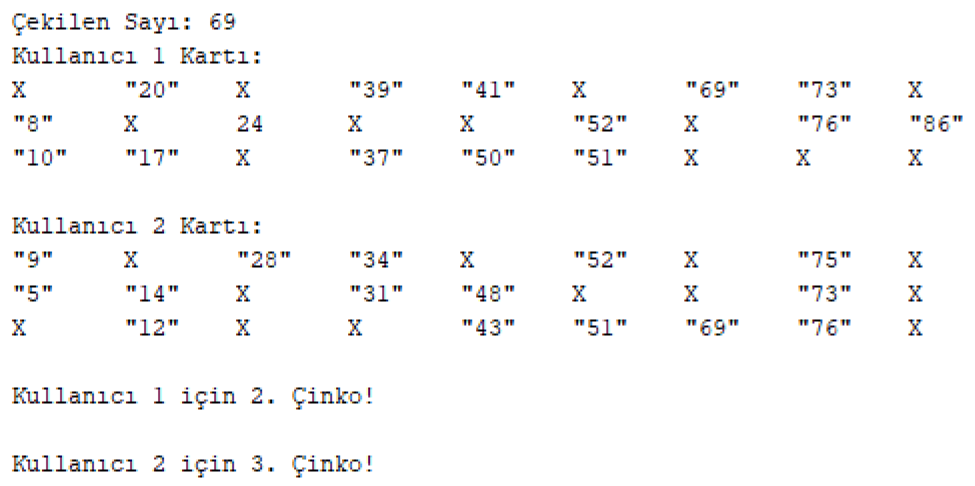
--\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 23
Kullanıcı 1 Kartı:
"6"     X       "28"    "32"    X       X       X       "79"    "84"
"8"     X       "29"    X       "47"    X       "68"    X       "90"
"1"     "13"    X       "40"    "50"    52      X       X       X

Kullanıcı 2 Kartı:
"8"     "19"    "23"    X       X       X       "62"    X       "82"
X       X       "30"    "34"    "45"    X       "68"    X       "84"
"10"    "17"    X       X       X       "57"    X       "75"    "85"

Kullanıcı 2 için 3. Çinko!

Kullanıcı 2 Tombala yaptı! Oyun Bitti.
```

• The randomly drawn number is constantly checked for two cards, and if the randomly drawn number is found on any of the cards, it is written as a "number". If all the numbers on the cards in the user's hand (card1 or card2) overlap in the form of X or "number", the user will have made the 3rd zinc. The system gives the user a message in the form of (1-2) 3rd zinc and the program continues to run. 3. Zinc means bingo, if the game is bingo, the game is over.

--\\\---------------------------------///--------------------------------



```
Çekilen Sayı: 69
Kullanıcı 1 Kartı:
X       "20"    X       "39"    "41"    X       "69"    "73"    X
"8"     X       24      X       X       "52"    X       "76"    "86"
"10"    "17"    X       "37"    "50"    "51"    X       X       X

Kullanıcı 2 Kartı:
"9"     X       "28"    "34"    X       "52"    X       "75"    X
"5"     "14"    X       "31"    "48"    X       X       "73"    X
X       "12"    X       X       "43"    "51"    "69"    "76"    X

Kullanıcı 1 için 2. Çinko!

Kullanıcı 2 için 3. Çinko!
```

!! Taken from different cards than the examples above!!

- After some random numbers are drawn, there may be zinc status on both cards, accordingly a message will be given on behalf of both users.