

# Simulation and Modelling Project Report

**Language:** Java (NetBeans IDE 8.2)

**Promotion:** Tickets and combined tickets sales of a club playing 17 home matches will take place. Combined tickets will be sold first and then daily tickets will be sold. (There will be no combined ticket sales to the away tribune.) These sales will be made simultaneously from 10 box offices.

The stadium played by this club has a tribune consisting of 10 sections. Its total capacity is 50508 people. [\*: Combined Ticket Sales, \*\*: Daily Ticket Sales (For Derby Days), \*\*\*: Daily Ticket Sales]

Names	Capacity	Price*	Price**	Price***
East Top	5612	2000 ₺	750 ₺	75 ₺
East Middle	5612	3000 ₺	1000 ₺	100 ₺
East Bottom	5612	2000 ₺	750 ₺	75 ₺
West Top	5612	2000 ₺	750 ₺	75 ₺
West Middle	5612	3000 ₺	1000 ₺	100 ₺
West-Bot	5612	2000 ₺	750 ₺	75 ₺
North	8418	1000 ₺	250 ₺	25 ₺
South Left	2806	1500 ₺	500 ₺	50 ₺
South-Mid	2806	1500 ₺	500 ₺	50 ₺
Away	2806	-	250 ₺	25 ₺

## Default;

- *By default, the number of combined tickets is expected to be about 40000. ( $\pm 500$ )*
- *By default, 90 days (129600 minute) of combined ticket sales queue can be entered and all tickets offered for sale will be sold.*
- *By default, unit combined ticket sales process time will be between 1-30 minutes.*
- *By default, a fan who is on the ticket queue will be able to buy at least 1 combined ticket and a maximum of 5 combined tickets.*

## Working Principles;

- *All of the combined tickets bought by one person are in the same tribune.*
- *The value you entered (for number of sales combined) it is very unlikely to change.*
- *After the combined ticket sale is completed, the code sets the price and occupancy amount by assuming a 17-match match sale.*
- *While doing the fast simulation mentioned above, the program assumes that all unsold places are sold in derby matches in the 6th, 12th and 13th weeks.*
- *It performs fast simulation of other matches while selling unsold tickets between 50% and 100%.*
- *It writes the number of total tickets sold of each match and it writes the income according to the tribunes of each match on "Match" notebook.*
- *Everything about the combined sales simulation is written in the "combined sales" notebook.*

## A small reminder for possible variables;

- *Up to 47702 combined tickets can be offered for sale. Because this is the capacity of the stadium.*
- *A fan can buy up to 20 tickets. Because we don't want all the tickets on one fan, right?*
- *You can enter the number of combined tickets for sale and the number of times we want the total time. (Of course, for positive integers provided by integer. "+2147483647")*

## Coding;

*I used a lot of array lists and lots of global integer variables. I used some nested loops; I drew a road map using conditional situations. I wrote 8 method/function to reduce complexity. I created a small "JFrame Form". In order to get permanent data, I added/wrote it to the notepad instead of the "System.out.println" function. And they open when the program ends.*

## Scenario;

*As written in the promotion(introduction), the code sells combined tickets in detail. It then simulates the remaining tickets according to the match type and gives us a report.*

## Goals;

*By examining these results, we could comment on how much combined ticket should be available if we were men of influence in this club, and we would have a basis in making this comment.*

*My goal is to answer this question; When the combined tickets are sold at these prices, how much is right for the club to be sold?*

## System Definition

### a) Entities

- *Personnel that allows 10 box offices to work.*
- *Lots of Fans/Supporter*
- *Stadium*
- *10 tribunes*
- *10 Box-offices*

### b) Attributes

- *Capacities of the tribunes/stands*
- *Number of permitted ticket rights*
- *Total combined sales time*
- *Match Types (Derby/No Derby)*
- *Ticket prices*

### c) Activities

- *Ticket sale*
- *Total income/profit*

### d) Events


- *Pick-up of fare*
- *Tribune selection*
- *Ticket number selection*

### e) State Variables

- *Seats (If=1 Full / If=0 Empty),*
- *Income list from each person,*
- *Empty seats,*
- *A list of 10 items in which revenues from the tribunes were kept during the combined sale,*
- *The number of combined tickets sold,*

- *lists which store which ticket is sold at which box office,*
- *Lists arrival time,*
- *Lists service time,*
- *Lists name of tribunes,*
- *Lists number of tickets,*
- *Lists fee,*
- *Lists service begins,*
- *Lists service end,*
- *Lists waiting time,*
- *Lists spend time,*
- *Lists idle time,*
- *And, different lists that I keep to use different data later.*

## System Behaviour

 *System or this piece of code / program is running once.*

## Working

*Firstly, a list with a total capacity is created in the program. Then the capacities of the tribunes are determined. The lists are duplicated so that their dimensions are the same.*

*These lists will be used for other transactions. First of all, all elements of all lists are 0. The 0s mean that list of information has not been sold in some lists yet. (For lists with total tribune capacity and capacities of the tribunes)*

*“Boyutlar” ArrayList maintains information on empty stands that have not yet been sold. Combined ticket prices are stored on the basis of “KombineFiyatları” ArrayList. The names of the tribune are also stored in “Tribün\_İsimleri” ArrayList.*

*With a random number generation process, I aimed to add reality to the number of combined tickets sold. ( $\pm 500$ ) I realized the combined ticket sales with a repetitive while loop, reducing the number of combined tickets for sale with the number of sold back tickets.*

*While the combined ticket is being sold, if a person buys more than one ticket, the information from the combined tickets purchased is synchronized to the location on the smallest seat. Such as in the İşlemSüreleri ArrayList, GelişSüreleri ArrayList, Gişeler ArrayList, BiletSayısı ArrayList, Tribünler ArrayList.*

*After that, Kombineli\_Yerler ArrayList is being updated. And, using count and delete methods; the number of combined tickets sold and empty seat index spaces are cleaned for convenience. With the sorting method, all lists are arranged synchronously according to the arrival times. (Sorted) (İşlemSüreleri, Gişeler, BiletSayısı, Tribünler, Ücret Array lists)*

*In order, While the full list is looking at the box office numbers processed, 10 different lists are stored in the box office index numbers. And, running sort method. This method performs sorting based on transaction times and arrival times.*

*Then, thanks to the for loops, waiting times and transaction termination times are calculated. Two different "Idle time" is calculated with two different methods. These are freetime and freetime2 methods. The freetime method finds idle time one by one. The freetime2 method finds idle time box-office by box-office.*

*At the end, with a fast simulation, an estimated match sale per game is realized. Two different prices are determined according to the match type. Occupancy rate estimation is made over unsold tickets. And ticket sales are made and income is obtained. And it is written to the file by program.*



The image shows a Windows-style application window titled "Stadium Box-Office Simulation". The window has a standard title bar with minimize, maximize, and close buttons. The main content area has a light gray background. At the top, there is an orange header bar with the title "Stadium Box-Office Simulation" in white text. Below the header, there are four input fields with labels to their left: "Number of sale combined :", "Total sales time for combined :", "Unit ticket processing time :", and "How many can buy number of tickets?". Each label is in a dark gray, italicized font. The input fields are white rectangles with thin black borders. At the bottom of the window, there are two large, light blue buttons with black text: "RUN" on the left and "DEFAULT" on the right.

*Photo0: Design button screen while the program is running.*

```

static void sorting(ArrayList<Integer> array) {
    int hold, min_index;
    for (i = 0; i < array.size() - 1; i++) {
        min_index = i;
        for (int j = i + 1; j < array.size(); j++) {
            if (array.get(j) < array.get(min_index)) {
                min_index = j;
            }
        }
        hold = array.get(i);
        array.set(i, array.get(min_index));
        array.set(min_index, hold);

        hold = İşlemSüreleri.get(i);
        İşlemSüreleri.set(i, İşlemSüreleri.get(min_index));
        İşlemSüreleri.set(min_index, hold);

        hold = Gişeler.get(i);
        Gişeler.set(i, Gişeler.get(min_index));
        Gişeler.set(min_index, hold);

        hold = BiletSayısı.get(i);
        BiletSayısı.set(i, BiletSayısı.get(min_index));
        BiletSayısı.set(min_index, hold);

        hold = Tribünler.get(i);
        Tribünler.set(i, Tribünler.get(min_index));
        Tribünler.set(min_index, hold);

        hold = Ücret.get(i);
        Ücret.set(i, Ücret.get(min_index));
        Ücret.set(min_index, hold);
        //-----
    }
}

```

Photo1: Sorting method image from the main class.

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    if (jTextField1.getText().trim().isEmpty()) {
        jTextField1.setText("40000");
    }
    if (jTextField4.getText().trim().isEmpty()) {
        jTextField4.setText("129601");//minumum 1 minute
    }
    if (jTextField5.getText().trim().isEmpty()) {
        jTextField5.setText("30");
    }
    if (jTextField3.getText().trim().isEmpty()) {
        jTextField3.setText("5");
    }
    if (Integer.parseInt(jTextField1.getText()) > 47702) {
        jTextField1.setText("47702");//max 47.702
    }
    if (Integer.parseInt(jTextField3.getText()) > 20) {
        jTextField3.setText("20");//max 20
    }
    Main.tahmin_edilen_kombine_sayisi = Integer.parseInt(jTextField1.getText());
    Main.k0_rand = Integer.parseInt(jTextField4.getText());
    Main.k_rand = Integer.parseInt(jTextField5.getText());
    Main.c_rand = Integer.parseInt(jTextField3.getText());
    try {
        Main.run();
    } catch (IOException ex) {
        Logger.getLogger(Form.class.getName()).log(Level.SEVERE, null, ex);
    }
    System.exit(0);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Form().setVisible(true);
        }
    });
}

```

Photo2: JFrame image 1



```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Main.run();
    } catch (IOException ex) {
        Logger.getLogger(Form.class.getName()).log(Level.SEVERE, null, ex);
    }
    System.exit(0);
}

private void jTextField1KeyPressed(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (!Character.isDigit(c) && c != '\b') {
        jTextField1.setEditable(false);
        JOptionPane.showMessageDialog(null, "Please enter number only");
    } else {
        jTextField1.setEditable(true);
    }
}

private void jTextField4KeyPressed(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (!Character.isDigit(c) && c != '\b') {
        jTextField4.setEditable(false);
        JOptionPane.showMessageDialog(null, "Please enter number only");
    } else {
        jTextField4.setEditable(true);
    }
}

private void jTextField5KeyPressed(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (!Character.isDigit(c) && c != '\b') {
        jTextField5.setEditable(false);
        JOptionPane.showMessageDialog(null, "Please enter number only");
    } else {
        jTextField5.setEditable(true);
    }
}

private void jTextField3KeyPressed(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (!Character.isDigit(c) && c != '\b') {
        jTextField3.setEditable(false);
        JOptionPane.showMessageDialog(null, "Please enter number only");
    } else {
        jTextField3.setEditable(true);
    }
}

```

Photo3: JFrame image 2



## System Analysis

*The output values in the system are only long and complex, suitable for evaluation by a specialist.*

## My Comment

*By doing fewer cycles, I could speed things up. In addition, in order not to lose the index loss due to splitting in the lists, editing could be done. The chance to change in the 50508-tribune capacity, which is the full multiple of nine, could be given.*

*In general, the question I am looking for; Can I realize the annual combined tickets sale? And after doing this, can I sell the remaining these tickets before the matches? And I think I can find the answers. I think that real-time combined tickets sale is also realized.*

COMBINED SALES - Notebook

Display: Name Right Selection: Year

TOTAL NUMBER OF COMBINED TICKETS SOLD: 38624

NUMBER OF COMBINED SOLD BY TRIBUNES

NAME OF TRIBUNES	WEST TOP	EAST MIDDLE	EAST BOTTOM	WEST TOP	WEST MIDDLE	WEST BOTTOM	NORTH	SOUTH LEFT	SOUTH MIDDLE	AWAY
NUMBER OF COMBINED TICKETS SOLD	4881	4748	4892	4889	8020	4887	4917	2808	2808	(0) 0

TOTAL INCOME FROM COMBINED TICKETS SALES: 81101000K

SALES FROM COMBINED TICKETS SALES BY TRIBUNES

	EAST TOP	EAST MIDDLE	EAST BOTTOM	WEST TOP	WEST MIDDLE	WEST BOTTOM	NORTH	SOUTH LEFT	SOUTH MIDDLE
	9982000K	1428000K	984000K	987000K	3500000K	9174000K	4917000K	4229000K	4209000K

SOLD TIME OF TICKETS (HOURS)

	56723	69098	70350	71000	70954	71257	69004	67005	68919	62971
TOTAL SOLD TIME: 68895										

Details Below

RANKING	ARRIVAL TIME(MIN)	SERVICE TIME(MIN)	NAME OF TRIBUNES	TOLL BOOTHS	NUMBER OF TICKETS	UNIT PRICE(K)	PRICE(K)	SERVICE BEGIN(MIN)
1	2	32	EAST TOP	8	2	2000	4000	2
2	24	32	EAST TOP	7	2	2000	4000	14
3	24	32	SOUTH MIDDLE	2	2	2000	4000	14
4	17	21	WEST TOP	4	1	2000	2000	18
5	17	23	NORTH	4	1	1000	1000	18
6	18	40	WEST MIDDLE	1	1	2000	2000	18
7	23	28	NORTH	5	1	1000	1000	25
8	45	20	WEST TOP	5	1	2000	2000	62
9	92	40	WEST MIDDLE	5	1	2000	2000	71
10	84	11	SOUTH MIDDLE	6	1	2000	2000	113
11	88	27	WEST MIDDLE	10	1	2000	2000	88
12	87	40	SOUTH LEFT	2	1	1000	1000	97
13	100	70	NORTH	10	1	1000	1000	112
14	105	108	EAST TOP	2	1	2000	2000	117
15	130	88	WEST TOP	4	1	2000	2000	126
16	113	81	WEST MIDDLE	8	1	2000	2000	118
17	118	6	WEST BOTTOM	6	1	1000	1000	142
18	118	20	NORTH	10	1	1000	1000	187
19	121	50	WEST MIDDLE	7	1	2000	2000	121
20	122	6	SOUTH LEFT	4	1	1000	1000	121
21	122	112	WEST BOTTOM	9	1	2000	2000	122
22	128	34	NORTH	4	1	1000	1000	128
23	149	6	EAST BOTTOM	10	1	2000	2000	207
24	159	1	WEST BOTTOM	6	1	1000	1000	154
25	162	2	EAST MIDDLE	1	2	3000	6000	162
26	168	7	WEST MIDDLE	7	1	2000	2000	177
27	169	40	EAST MIDDLE	2	4	2000	2000	169
28	179	24	EAST TOP	4	1	2000	2000	149
29	182	84	WEST TOP	7	1	2000	2000	202
30	187	55	WEST TOP	1	1	1000	1000	187
31	194	14	WEST TOP	1	2	2000	4000	232
32	207	52	SOUTH MIDDLE	6	1	1000	1000	207

Photo4: COMBINED SALES notebook image

MATCH - Notebook

Display: Name Right Selection: Year

NUMBER OF SEATS THAT CAN BE SOLD: 10584

Details Below

ONLY MATCH TICKET SALES

1 MATCH	NUMBER OF TICKETS SOLD: 5601
2 MATCH	NUMBER OF TICKETS SOLD: 7459
3 MATCH	NUMBER OF TICKETS SOLD: 6150
4 MATCH	NUMBER OF TICKETS SOLD: 9835
5 MATCH	NUMBER OF TICKETS SOLD: 7459
6 MATCH (DEBT)	NUMBER OF TICKETS SOLD: 10194
7 MATCH	NUMBER OF TICKETS SOLD: 6919
8 MATCH	NUMBER OF TICKETS SOLD: 10194
9 MATCH	NUMBER OF TICKETS SOLD: 10454
10 MATCH	NUMBER OF TICKETS SOLD: 6930
11 MATCH	NUMBER OF TICKETS SOLD: 10584
12 MATCH (DEBT)	NUMBER OF TICKETS SOLD: 10194
13 MATCH (DEBT)	NUMBER OF TICKETS SOLD: 10584
14 MATCH	NUMBER OF TICKETS SOLD: 9776
15 MATCH	NUMBER OF TICKETS SOLD: 6919
16 MATCH	

Photo5: MATCH notebook image1

