GEBZE TECHNICAL UNIVERSITY

Computer Engineering Department

House Searching Platform

FINAL REPORT

GROUP BOOTSTRAPPERS - Members

Taner GEYHAN
Muhammet Melih KAVRAZ
Yağmur KARAMAN
Fatiha BEQİROVSKİ
Nilay KEVEN
Fatih KİRAZ

Advisor
Doç. Dr. Fatih Erdoğan SEVİLGEN

31/05/2018

1. TECHNICAL DESCRIPTION OF DATA STRUCTURES USED

At this stage of development we only used Array List, Stack and Queue. Our program only filters according to the price at the moment. Firstly the prices are stored in an Array List. Moreover they are sorted. Furthermore, Stack is used to sort the prices to descending from and Queue is used to sort the prices in ascending form. The data structures Stack and Queue’s properties; LIFO and FIFO are used according to the sort of array.

ArrayList

What is an ArrayList?

- There are multiple ways to obtain a data structure in Java. ArrayList is one of them.
- ArrayList objedir. ArrayList is an Object. It contains its own methods. Since it’s an object it’s in the heap part of the memory. It has an dynamic structure. There is no need to specify the size while declaring one.

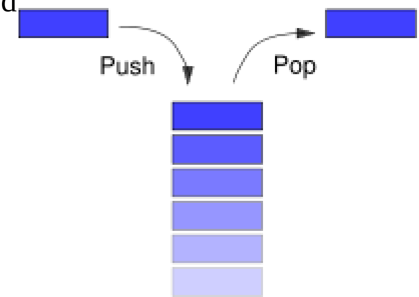
Where are we using the ArrayList?

- We use ArrayList to keep the users of the system and advertisements.

Stack

What is a Stack?

- Stack is a data structure that provides the objects to be stored with LIFO (Last In First Out) method.
- It is oftenly used in computer science.
- It’s logic is same as when putting the boxes one to another. The box to be added will be at the top.
- To take a box from the bottom or middle, one has to take the boxes from the top one by one until it reaches the desired box.



Where are we using the Stack?

- We use stack to provide option to sort the houses according to their prices in descending order.
- When we put sorted advertisements into the stack, the bottom of the stack will consist of the advertisement with the lowest price, and the top of the stack will hold the advertisement with the highest price.

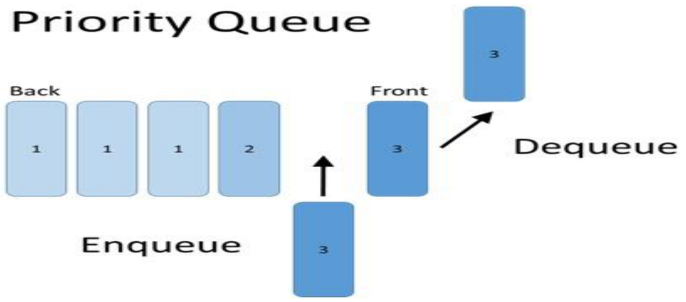
- By popping the elements from the stack we get the houses sorted in descending order according to their prices.

Priority Queue

What is a Priority Queue?

Priority Queue is an extension of queue with following properties:

- Every item has a priority associated with it.
- An element with high priority is dequeued before an element with low priority.
- If two elements have the same priority, they are served according to their order in the queue.



Where are we using the Priority Queue?

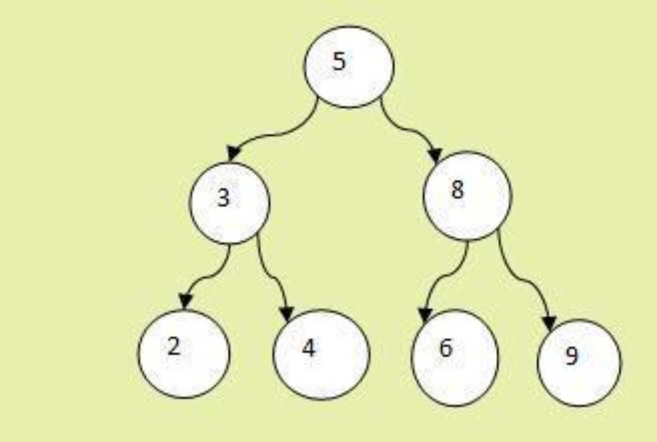
- We use queue while providing the user to sort the advertisements in ascending order according to their prices.
- When we poll an element from the priority queue it gives us the lowest one, so we get the houses in ascending order.

Binary Search Tree

What is a Binary Search Tree?

Binary Search Tree is a special kind of Binary Tree. Each element in the tree is in special order by having a relationship depending on it’s value (weather bigger or smaller). For instance, if we are to have a binary search tree of integer values, the values will have a bigger-smaller relationship.

The left subtree of a node contains only nodes with keys lesser than the node’s key. The right subtree of a node contains only nodes with keys greater than the node’s key. The left and right subtree each must also be a binary search tree. There must be no duplicate nodes.



- For instance picture given above represents a binary search tree. As we can see the values on the left side of the root are less than root, and the values on the right side are greater than root.

- In binary search tree each add remove or any other operation has to be performed according to this rule. After each performed operation the structure of the tree should be preserved.

Where are we using the Binary Search Tree?

We tend to use this data structure to provide the user more categories (filters). We kept the area information of the house in the binary search tree, which is of an integer type. With this functionality the user can filter the search by size of the houses aswell. The user can choose to filter the houses by their size in both ascending and descending order.

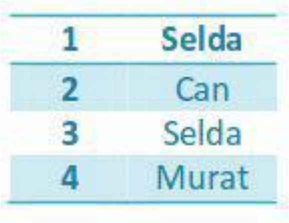
1. Descending: By traversing the implemented Binary Search Tree inorderly we get the size of the house and present it to the user in descending order.
2. Ascending: To accomplish this action we must be able to trace the size of the houses from smallest one to the biggest one. We take advantage of inorder traverse algorithm in this stage too. But not as orginal order left-root-right order, we arranged it in right-root-left order.

Map

What is Map?

Map is a member of Java Collections Framework.

Map pairs keys to it’s values. For instance in a list of names, if we are to give each name a row number we are to use a Map. In this operation each row number is a key while each name is a value. In order to prevent the list from a mix up a different row number is given to each name. The row numbers are held as a key, and the names as a value. The row numbers are different but the names can be same. For example, in the list given below there are two person named Selda. Each one of them has a different row number. There must be no duplicate row number entries. The same is not compulsory for names.



1	Selda
2	Can
3	Selda
4	Murat

Where are we using the Map data structure?

In the project, we use this data structure to provide the user to see total number of available houses in a certain district. By providing this functionality with no further filtering the user can be able to see the advertisement number in the desired district. In this functionality the number of rooms, the size of the house, price and other properties are ignored. Which means that when the number of houses in a certain district is calculated all the available houses are counted. In Map data structure district’s name is held as a key, and the total number of houses is held as a value.

Set

What is Set?

- The elements inside a set are not in a list format, it’s more like a bag full of elements. It comes from a mathematics set concept. A set with no elements is an empty set. A set can consist of one, two or more elements.
- In computer science the infinite elements cannot be declared, the infinite operations cannot be performed. Which is why like in lists, the set has a condition to consist of finite number of elements.
- A set is an abstract data type that can store certain values, without any particular order, and no repeated values.

Where are we using the Set data structure?

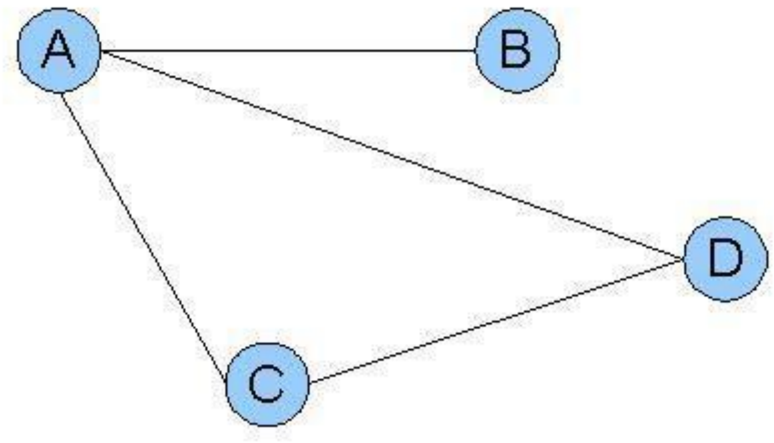
- The user who has successfully entered the system may wish to save an advertisement. The data structure is used to keep the favorite advertisements of the registered user.

- In our project that particular data structure is set. The reason we used set is to prevent the user to save the same advertisement more than once.
- By using set we saved from memory, and also we provided the user cleaner cluster of the desired advertisements.

Graph

What is a Graph?

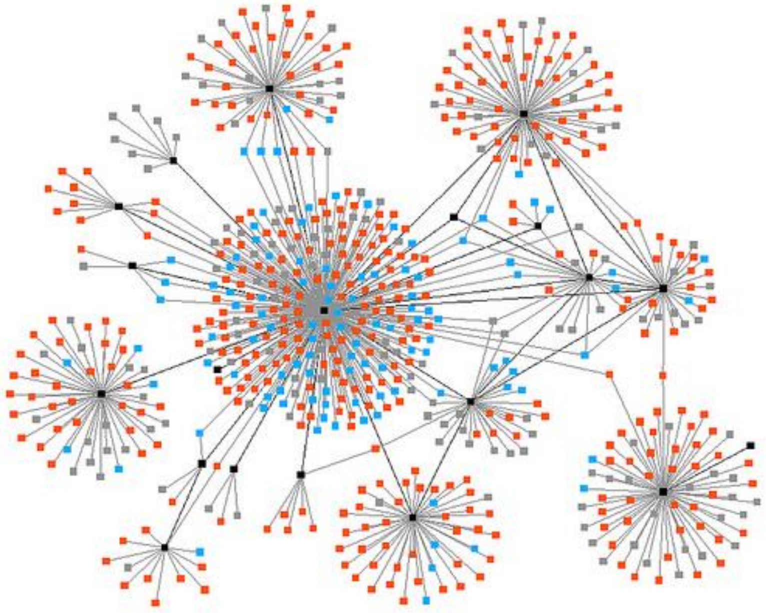
In the world of computing and also the real world graph is a way to represent various situations. For example using graph we can represent a computer network, the map of the characters or a decision tree. Computer science uses various mathematical and visual methods to express these constructs that are encountered in various applications. Accordingly, the entities in a graft are represented by nodes, and the relations between these entities are expressed by grafted edges. It is possible to divide graphs as directional graphs and non-directional graphs, depending on whether the edges are oriented or not. Also, according to the value of the edges, valuable graphs or names of worthless graphs can be given.



For example, the graph above shows a graph consisting of 4 nodes and 4 edges. This graph can be expressed as G = ({A, B, C, D}, {A, B}, (A, C), (C, D), (A, D)}) Therefore, the graphs are written in the form of G = (V, E), as well as nodes and edges.

Where are we using the Graph?

We Graph data structure to provide the user categorizing according to the distance. We keep individual graph for each district. In Graph's nodes are connected to the center, and in the edges we keep the distance to the center information. We also added several options which are available to user while filtering. To be precise; if user desires to see the houses according to their distance to the districts center, it has to choose the district firstly. At this point the center of the district is accepted as origin (0,0). Moreover user has to choose the distance. The distance chosen is accepted as radius. With the distance information added the houses are checked. The houses appropriate according to the distance chosen are firstly sorted with Merge Sort recursively. Finally the sorted list is presented to the user. The graph structure formed is similiar to the picture below.



3. SCENARIOS

User Scenario I

The person trying to enter the system is given two options; enter as an admin or as user.

Person chooses appropriate login option.

From the options menu it chooses rent/buy advertisement option.

From the filters provided it enters district/number of rooms/size of the house information.

The proper advertisements according to the filters are listed.

From advertisement's contact information it reaches the owner of the desired advertisement.

User can choose to go back and take a look at other advertisements or exit the system.

It chooses to go back.

The user is redirected to the main page.

From the filters provided it enters distance and district information.

The houses sorted according to the distance of districts center are listed.

User can choose to go back and take a look at other advertisements or exit the system.

It chooses to exit.

It exits the system.

User Scenario II

The person trying to enter the system is given two options; enter as an admin or as user.

Person chooses appropriate login option.

On the options menu it chooses to add an advertisement option.

In order to add the properties of the advertisement the form is shown.

The form is filled according to the advertisement's properties.

The advertisement is added.

User can choose to go back and take a look at other advertisements or exit the system.

It chooses to exit.

It exits the system.

Admin Scenario I

The person trying to enter the system is given two options; enter as an admin or as user.
Person chooses appropriate login option.
From options menu it chooses to add an advertisement option.
In order to add the properties of the advertisement the form is shown.
The form is filled according to the advertisement’s properties.
The advertisement is added.
Admin can choose to go back and take a look at other advertisements or exit the system.
It chooses to exit.
It exits the system.

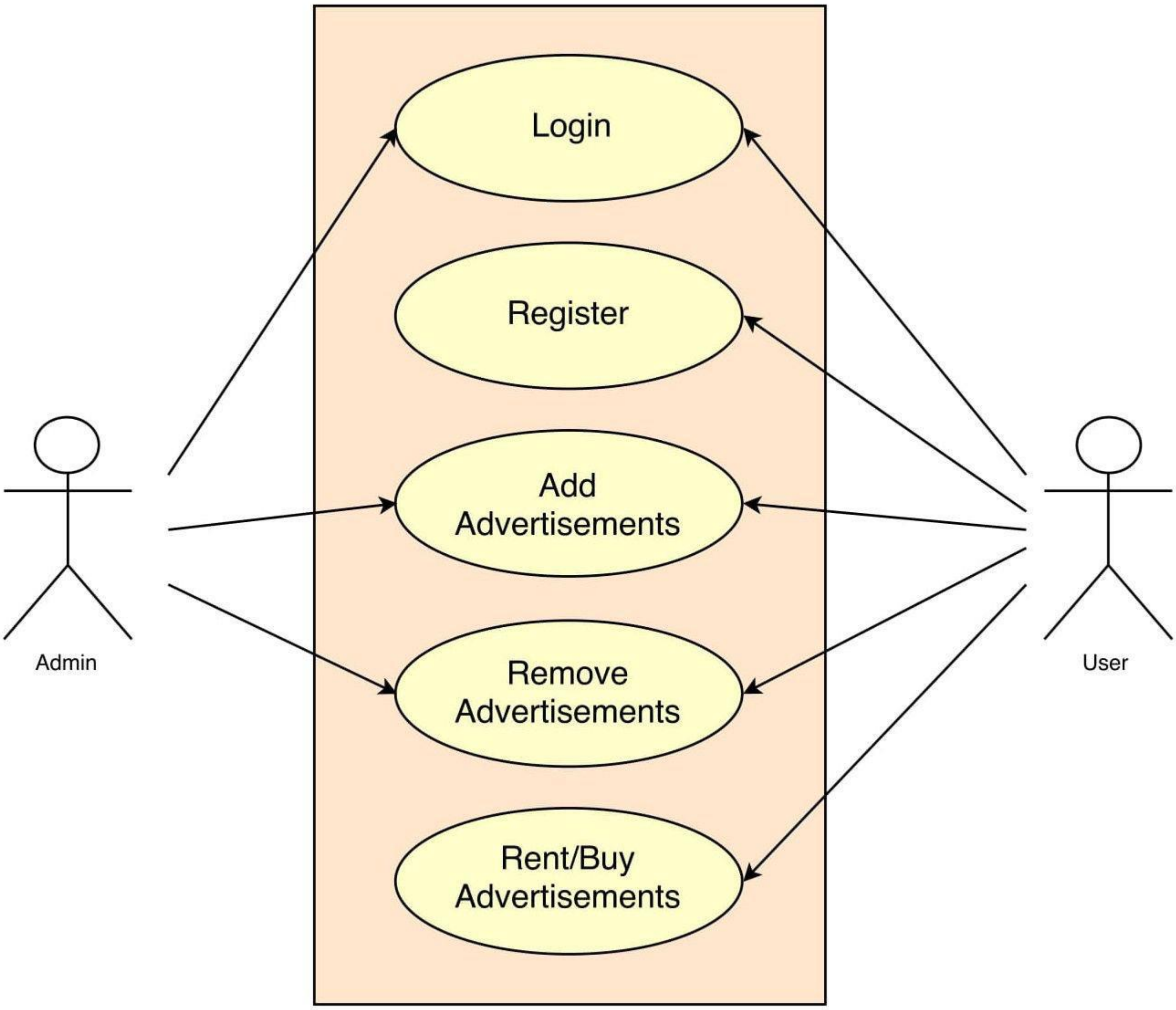
Admin Scenario II

The person trying to enter the system is given two options; enter as an admin or as user.
Person chooses appropriate login option.
From options menu it chooses to remove an advertisement option.
The advertisement to be removed is searched in database.
The advertisement is found and removed.
Admin can choose to go back and take a look at other advertisements or exit the system.
It chooses to go back and continue.
The admin is redirected to the options menu.

3. DIAGRAMS

The class diagram is under 5th sub-section.

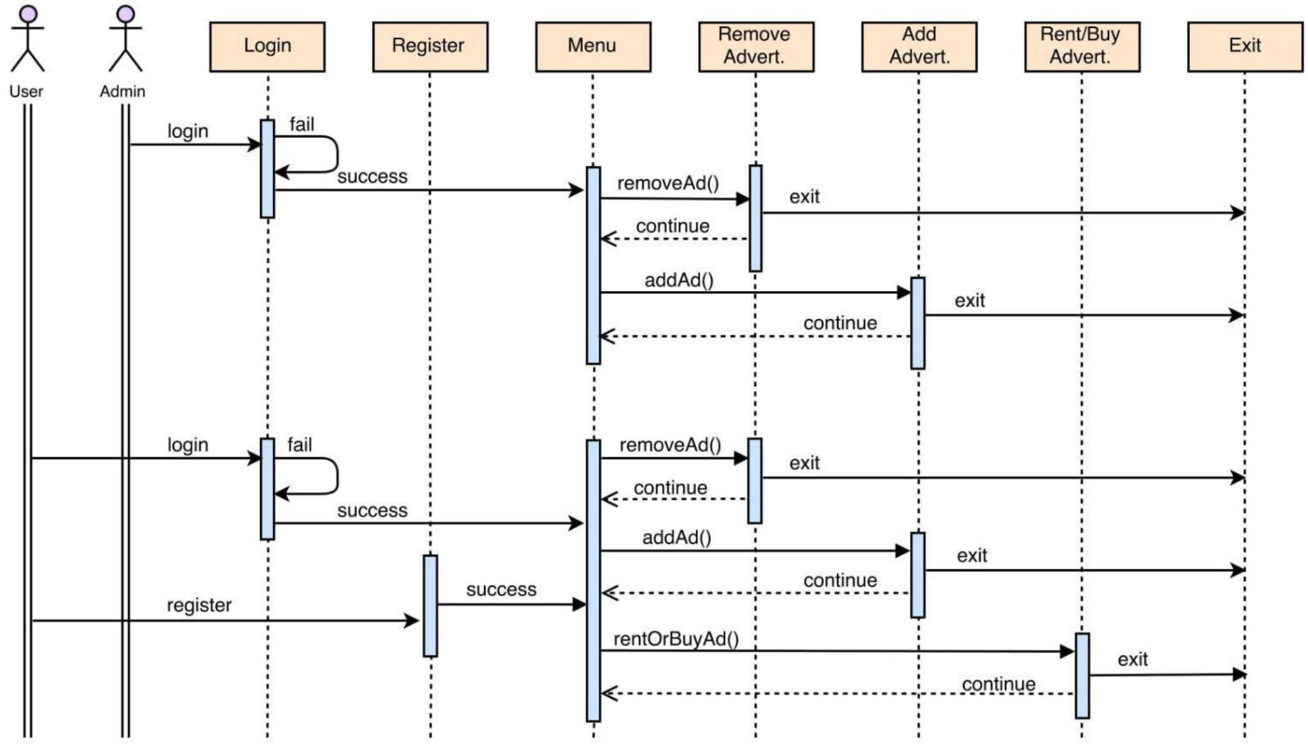
Use Case Diagram: Admin and User can perform a few operations. Admin can login into system, add and advertisement or remove one. User can login into system, register, add an advertisement or perform rent/buy actions.



Sequence Diagram: In this diagram we can see the actions users can perform more clear.

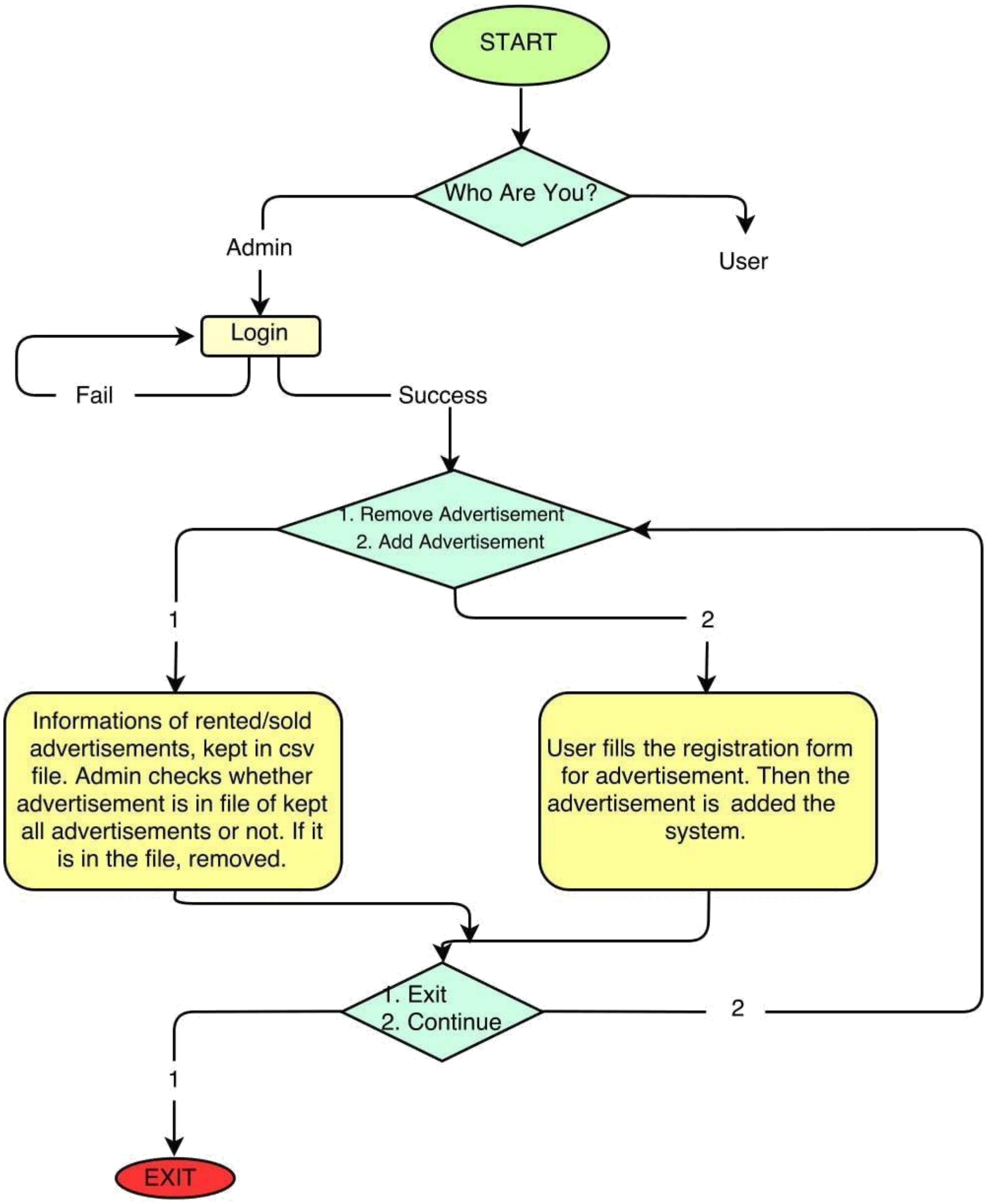
If admin logs into system successfully he/she will be directed to menu window. It may choose to add an advertisement or remove one. After each action performed, it will be redirected to menu, if admin doesn't wish to continue it can exit the system.

User can login into system or register. If user successfully performs any of those two it will be redirected to menu window. User can choose to add or remove an advertisement or rent/sale action. After each action performed, it will be redirected to menu, if user doesn't wish to continue it can exit the system.

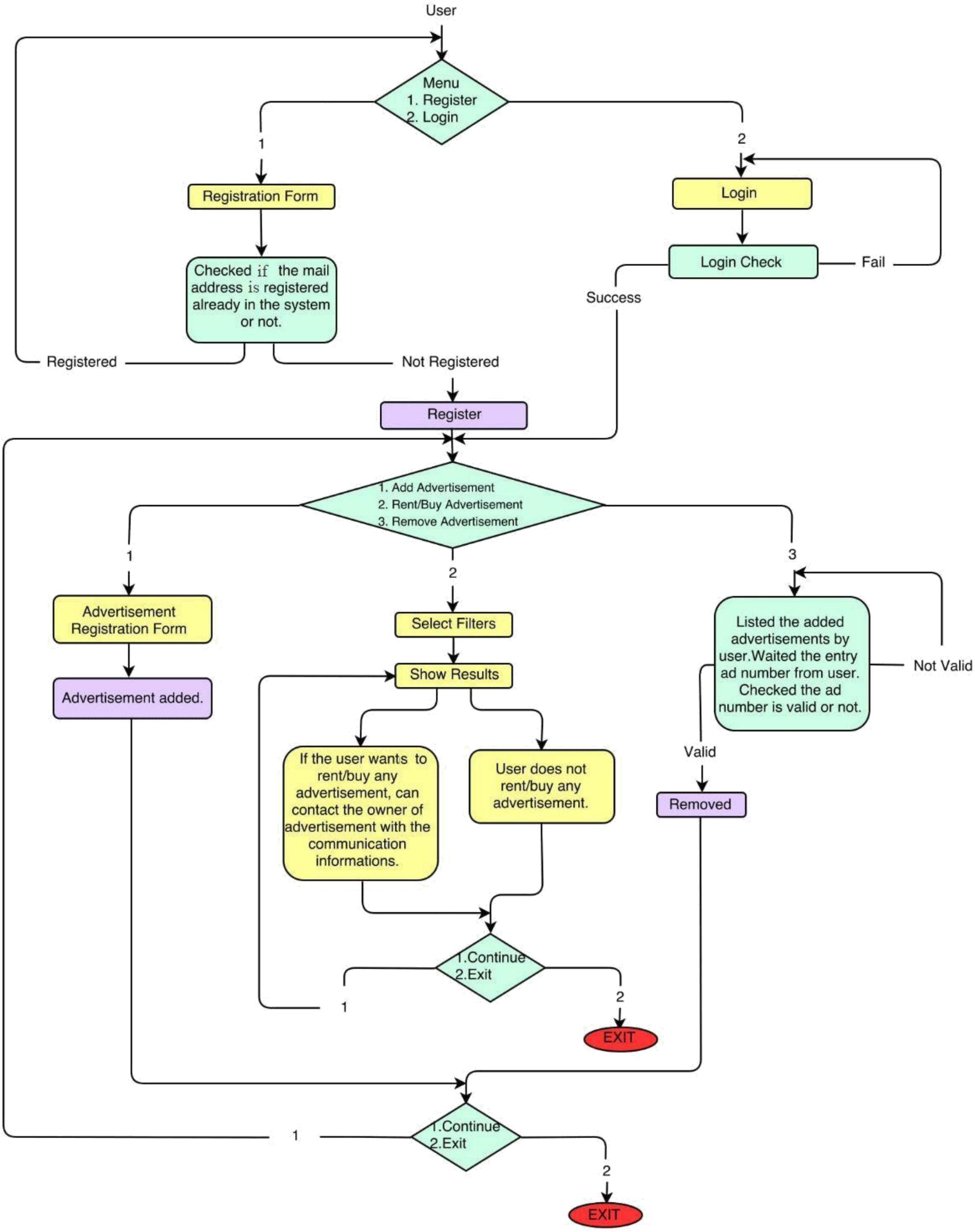


Flowchart: The first flowchart represents admin. The flowchart that represents user is added below admin’s flowchart. Before allowing any kind of usage of system the person trying to enter needs to determine whether it’s an admin or user. For admin there is only login operation which has been assigned by system and cannot register to the system. If the admin logs in to the system successfully it will face the menu window; showing him what actions can be performed. If admin wants to remove an advertisement, he/she needs to enter advertisement’s information. Moreover the information entered will be searched in advertisement database. If found it will be deleted and the action will be done. Afterwards, admin will be asked whether it wants to exit or not. If he/she chooses to continue the menu window will be shown again. Otherwise he will exit the system. If admin wants to add an advertisement, he/she will have to enter advertisement information. After the information is entered the advertisement is added to system and action is terminated. Afterwards, admin will be asked whether it wants to exit or not. If he/she chooses to continue the menu window will be shown again. Otherwise he/she will exit the system.

Second flowchart represent user: The user will be given two options; whether to login or register.
If the user chooses to login, it will be asked to enter log in informations. If the login is performed successfully the menu of actions available will be shown. If the user chooses to register, it will be asked to enter registartion information, furthermore the info entered will be checked in User Database; whether the user has already been registered or not. If not the user will be registered and menu window will be shown.
If user wants to add an advertisement, it will have to enter advertisement information. After the information is entered the advertisement is added to system and the action is terminated. Afterwards, user will be asked whether it wants to exit or not. If he/she chooses to continue the menu window will be shown again. Otherwise he/she will exit the system. If user wants to perform rent/buy action advertisement categorizing window will be shown and the user can choose the advertisements properties. After user has choosen the properties, if he/she likes an advertisement it may reach the advertisement owner by getting the contact information, or only view the advertisement. After the operation is done, user will be asked whether it wants to exit or not. If he/she chooses to continue the menu window will be shown again. Otherwise he/she will exit the system. If user wants to remove an advertisement, it has to enter advertisement’s information. Moreover, the information entered will be searched in Advertisement Database. If found the advertisement will be removed and the action will be terminated. Once again user will be asked whether it wants to exit or not. If he/she chooses to continue the menu window will be shown again. Otherwise he/she will exit the system.


START
Who Are You?
Admin
User
Login
Fail
Success
1. Remove Advertisement
2. Add Advertisement
1
2
Informations of rented/sold advertisements, kept in csv file. Admin checks whether advertisement is in file of kept all advertisements or not. If it is in the file, removed.
User fills the registration form for advertisement. Then the advertisement is added the system.
1. Exit
2. Continue
2
1
EXIT

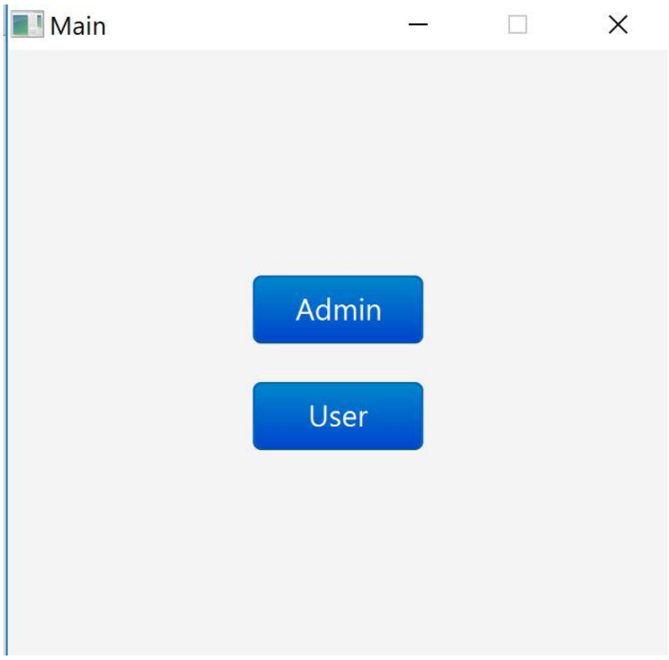
31/05/2018


User
Menu
1. Register
2. Login
1
2
Registration Form
Login
Login Check
Fail
Success
Checked if the mail address is registered already in the system or not.
Registered
Not Registered
Register
1. Add Advertisement
2. Rent/Buy Advertisement
3. Remove Advertisement
1
2
3
Advertisement Registration Form
Advertisement added.
Select Filters
Show Results
Listed the added advertisements by user.Waited the entry ad number from user. Checked the ad number is valid or not.
Not Valid
Valid
Removed
If the user wants to rent/buy any advertisement, can contact the owner of advertisement with the communication informations.
User does not rent/buy any advertisement.
1.Continue
2.Exit
1
2
EXIT
1.Continue
2.Exit
1
2
EXIT

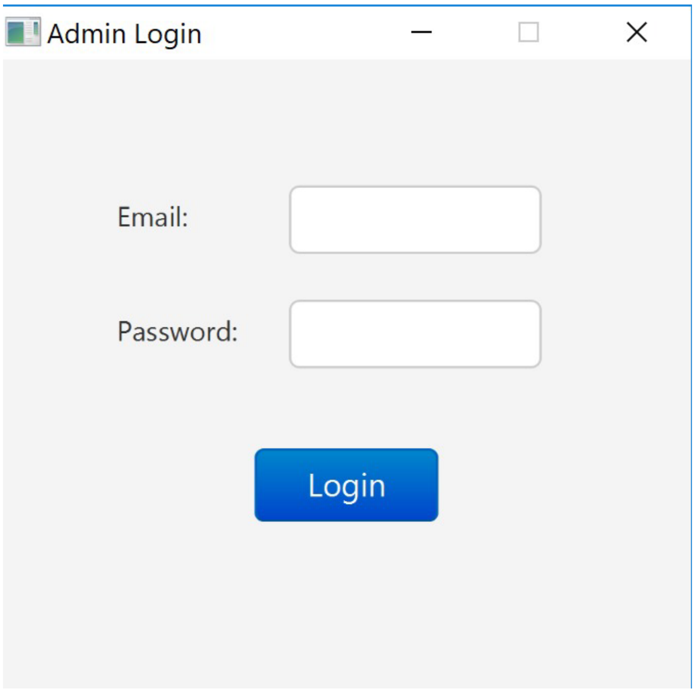
31/05/2018

4. GUI PROPERTIES

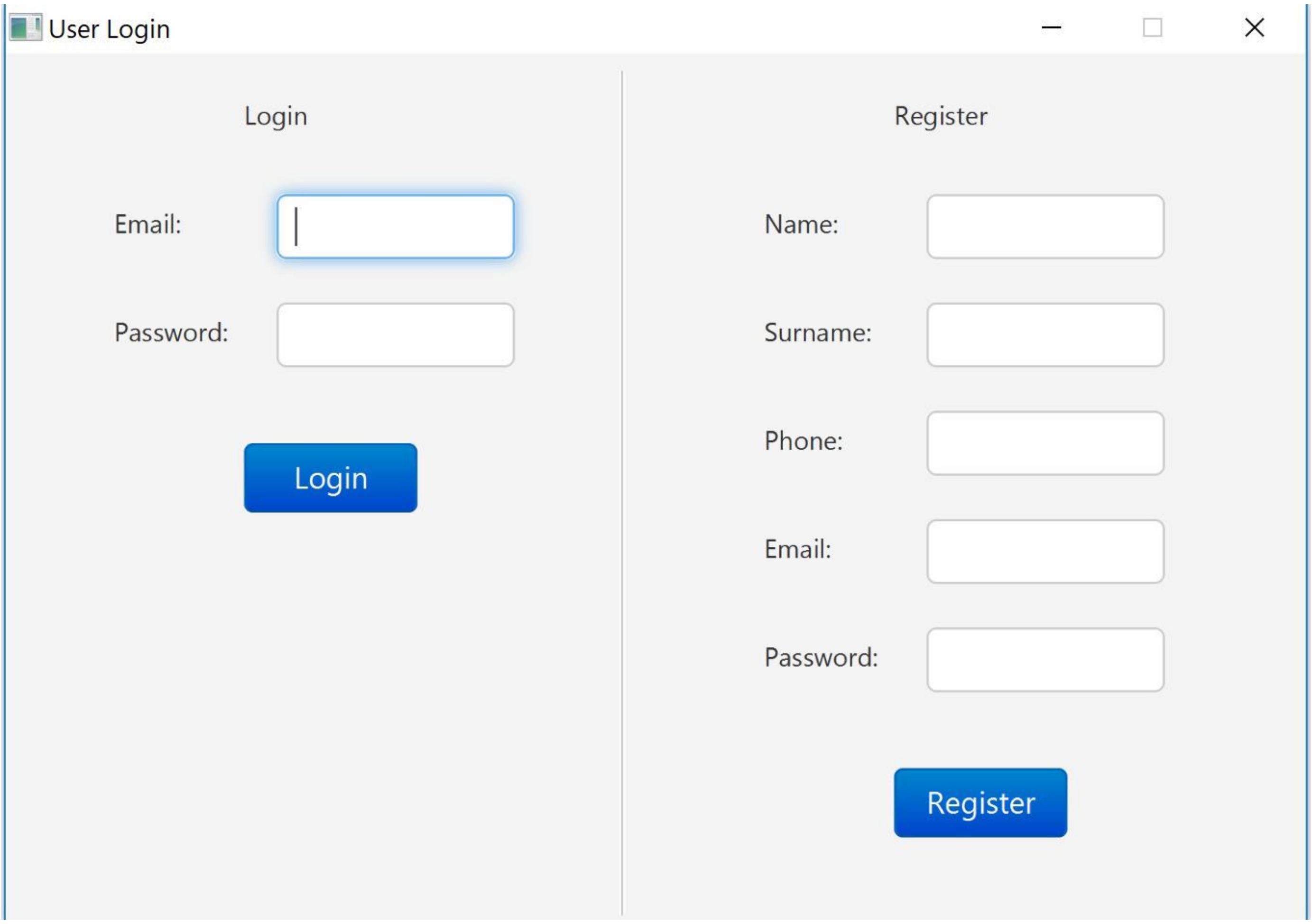
User enters either as an Admin or User.



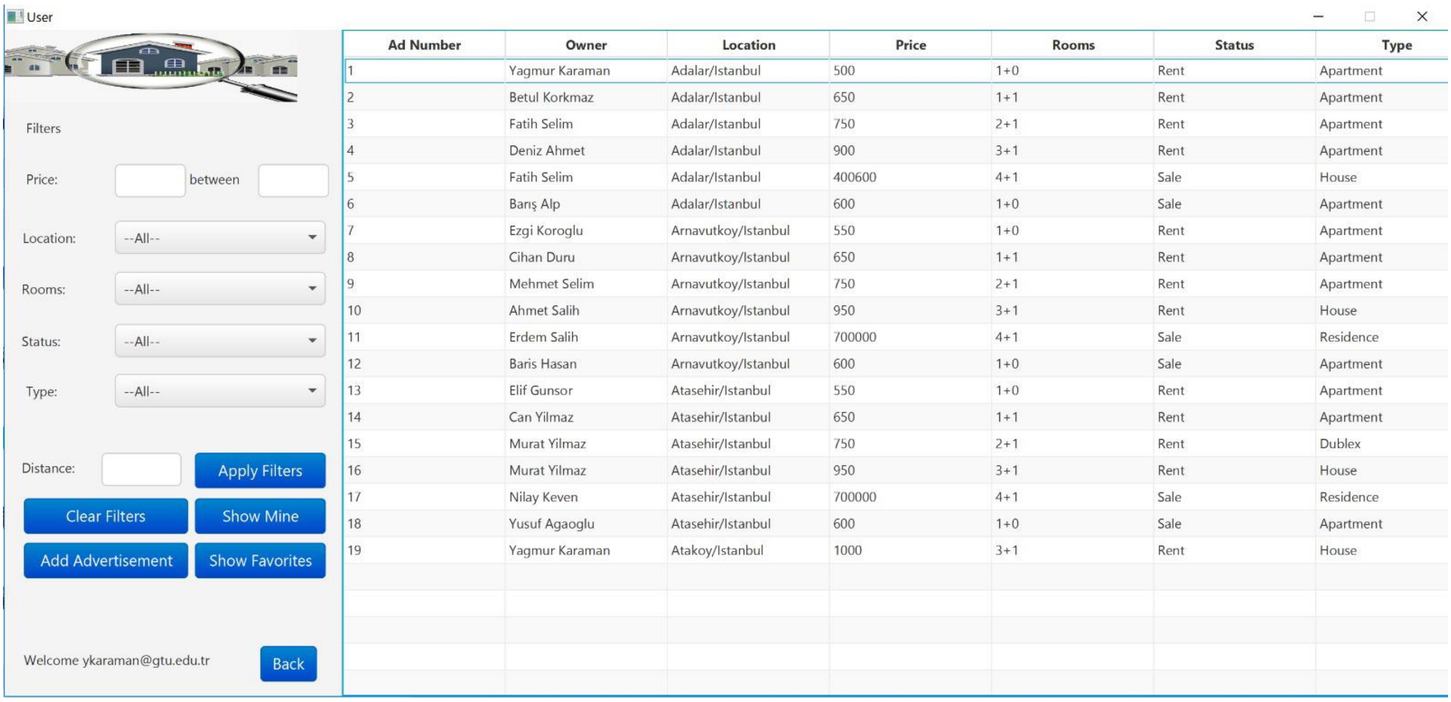
If the user chooses to enter as admin it will have to enter email and password.



If the user chooses to enter as an User, Register and Login charts will be shown to the user. In this charts user can register by entering username, password and e-mail; or login by entering username and password. Same name and e-mail cannot be registered more than once.



User who successfully enters the system can see the houses for sale/for rent, can categorize them and sort them, and can also set up an new advertisement or remove its own old one.



Price: between

Location: --All--

Rooms: --All--

Status: --All--

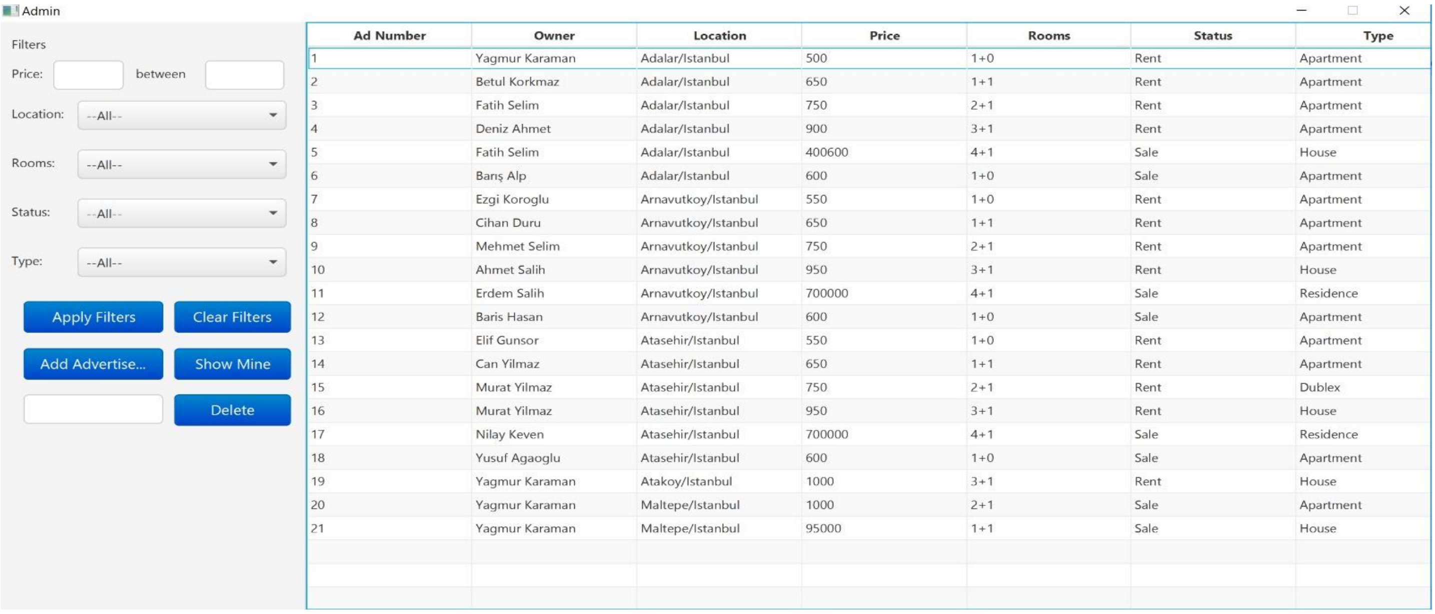
Type: --All--

Distance:

Welcome ykaraman@gtu.edu.tr

Ad Number	Owner	Location	Price	Rooms	Status	Type
1	Yagmur Karaman	Adalar/Istanbul	500	1+0	Rent	Apartment
2	Betul Korkmaz	Adalar/Istanbul	650	1+1	Rent	Apartment
3	Fatih Selim	Adalar/Istanbul	750	2+1	Rent	Apartment
4	Deniz Ahmet	Adalar/Istanbul	900	3+1	Rent	Apartment
5	Fatih Selim	Adalar/Istanbul	400600	4+1	Sale	House
6	Barış Alp	Adalar/Istanbul	600	1+0	Sale	Apartment
7	Ezgi Koroglu	Arnavutkoy/Istanbul	550	1+0	Rent	Apartment
8	Cihan Duru	Arnavutkoy/Istanbul	650	1+1	Rent	Apartment
9	Mehmet Selim	Arnavutkoy/Istanbul	750	2+1	Rent	Apartment
10	Ahmet Salih	Arnavutkoy/Istanbul	950	3+1	Rent	House
11	Erdem Salih	Arnavutkoy/Istanbul	700000	4+1	Sale	Residence
12	Baris Hasan	Arnavutkoy/Istanbul	600	1+0	Sale	Apartment
13	Elif Gunsor	Atasehir/Istanbul	550	1+0	Rent	Apartment
14	Can Yilmaz	Atasehir/Istanbul	650	1+1	Rent	Apartment
15	Murat Yilmaz	Atasehir/Istanbul	750	2+1	Rent	Dublex
16	Murat Yilmaz	Atasehir/Istanbul	950	3+1	Rent	House
17	Nilay Keven	Atasehir/Istanbul	700000	4+1	Sale	Residence
18	Yusuf Agaoglu	Atasehir/Istanbul	600	1+0	Sale	Apartment
19	Yagmur Karaman	Atakoy/Istanbul	1000	3+1	Rent	House

User who successfully enters the system as admin can see the houses for sale/for rent, can categorize them and sort them, and can also set up an new advertisement or remove an old one.



Filters

Price: between

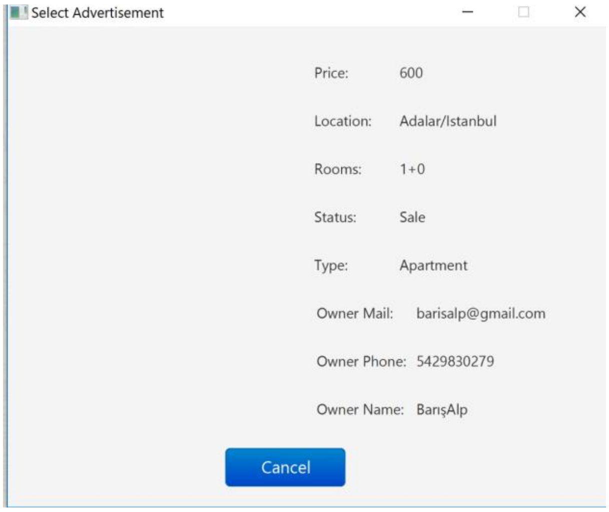
Location: --All--

Rooms: --All--

Status: --All--

Type: --All--

Ad Number	Owner	Location	Price	Rooms	Status	Type
1	Yagmur Karaman	Adalar/Istanbul	500	1+0	Rent	Apartment
2	Betul Korkmaz	Adalar/Istanbul	650	1+1	Rent	Apartment
3	Fatih Selim	Adalar/Istanbul	750	2+1	Rent	Apartment
4	Deniz Ahmet	Adalar/Istanbul	900	3+1	Rent	Apartment
5	Fatih Selim	Adalar/Istanbul	400600	4+1	Sale	House
6	Barış Alp	Adalar/Istanbul	600	1+0	Sale	Apartment
7	Ezgi Koroglu	Arnavutkoy/Istanbul	550	1+0	Rent	Apartment
8	Cihan Duru	Arnavutkoy/Istanbul	650	1+1	Rent	Apartment
9	Mehmet Selim	Arnavutkoy/Istanbul	750	2+1	Rent	Apartment
10	Ahmet Salih	Arnavutkoy/Istanbul	950	3+1	Rent	House
11	Erdem Salih	Arnavutkoy/Istanbul	700000	4+1	Sale	Residence
12	Baris Hasan	Arnavutkoy/Istanbul	600	1+0	Sale	Apartment
13	Elif Gunsor	Atasehir/Istanbul	550	1+0	Rent	Apartment
14	Can Yilmaz	Atasehir/Istanbul	650	1+1	Rent	Apartment
15	Murat Yilmaz	Atasehir/Istanbul	750	2+1	Rent	Dublex
16	Murat Yilmaz	Atasehir/Istanbul	950	3+1	Rent	House
17	Nilay Keven	Atasehir/Istanbul	700000	4+1	Sale	Residence
18	Yusuf Agaoglu	Atasehir/Istanbul	600	1+0	Sale	Apartment
19	Yagmur Karaman	Atakoy/Istanbul	1000	3+1	Rent	House
20	Yagmur Karaman	Maltepe/Istanbul	1000	2+1	Sale	Apartment
21	Yagmur Karaman	Maltepe/Istanbul	95000	1+1	Sale	House



Price: 600

Location: Adalar/Istanbul

Rooms: 1+0

Status: Sale

Type: Apartment

Owner Mail: barisalp@gmail.com

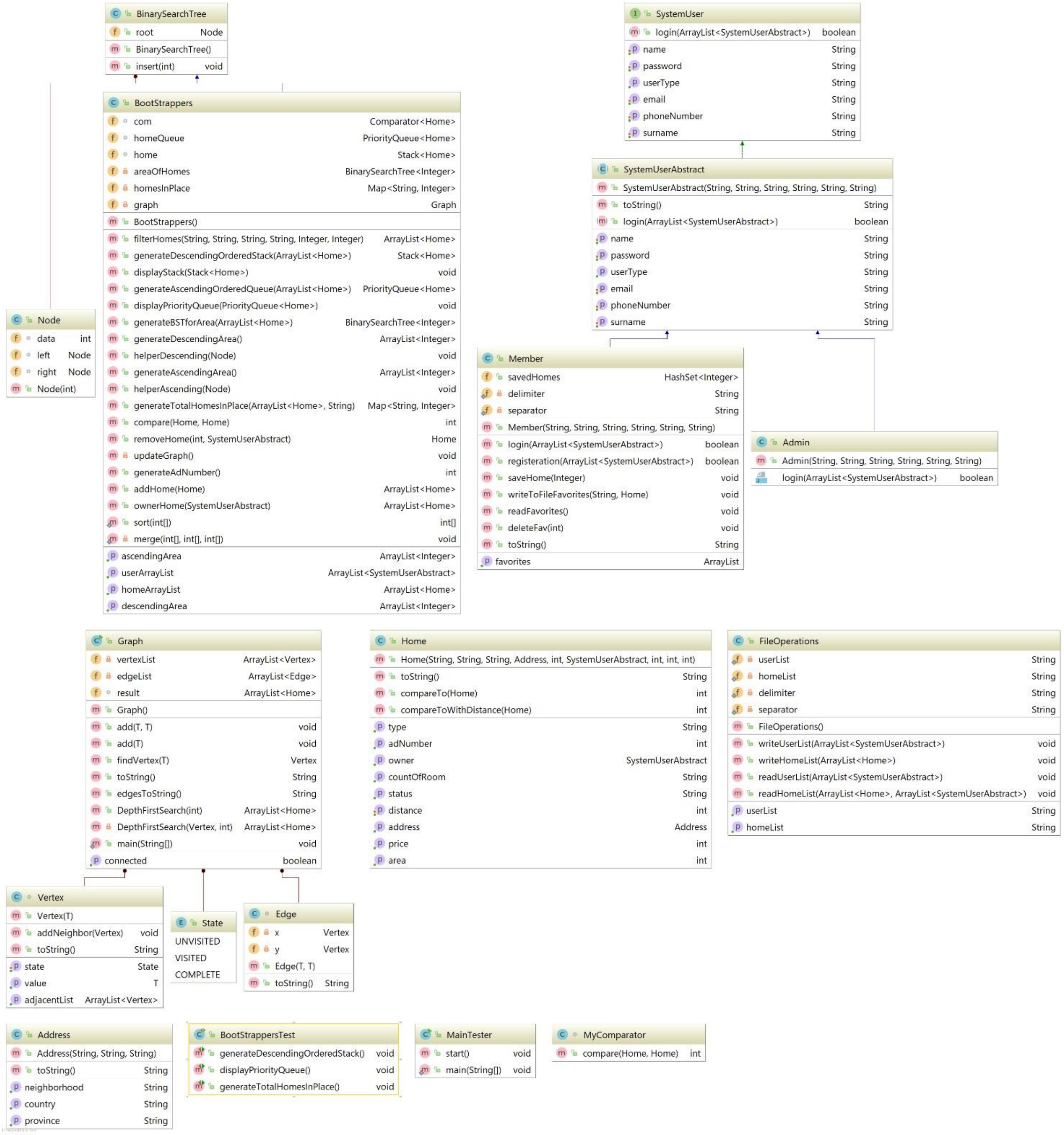
Owner Phone: 5429830279

Owner Name: BarışAlp

5. BRIEF DESCRIPTION OF HOW OOP CONCEPTS ARE IMPLEMENTED

For user groups two class have been declared. One for Member and one for Admin. Thoses classes extend SystemUserAbstract class. While SystemUserAbstract class implements SystemUser interface. Class Address has been declared, which stores address information. Class Home has been declared, which stores houses information. In order to determine house address Home class contains 1 Address object, in order to determine the house owner it contains 1 Member object. For file reading/writing operations File Operations class has been declared. FileOperations has a Member and Home. BootStrappers class also has a Member and Home.

To store the users of system ArrayList data structure is used. The login/register operation searches are to be performed on this list. The advertisment information is kept in ArrayList. The user can filter the desired features, and choose to sort them in descending or ascending order. The price information will be used in order to perform those actions. When sorted in ascending order (according to the price), categorized advertisement price information is listed with PriorityQueue and by using FIFO feature, is presented to the user. When sorted in descending order (according to the price), categorized advertisement price information is listed with Stak data structure and by using LIFO feature, is presented to the user.


BinarySearchTree
root Node
BinarySearchTree()
insert(int) void
BootStrappers
com Comparator<Home>
homeQueue PriorityQueue<Home>
home Stack<Home>
areaOfHomes BinarySearchTree<Integer>
homesInPlace Map<String, Integer>
graph Graph
BootStrappers()
filterHomes(String, String, String, String, Integer, Integer) ArrayList<Home>
generateDescendingOrderedStack(ArrayList<Home>) Stack<Home>
displayStack(Stack<Home>) void
generateAscendingOrderedQueue(ArrayList<Home>) PriorityQueue<Home>
displayPriorityQueue(PriorityQueue<Home>) void
generateBSTforArea(ArrayList<Home>) BinarySearchTree<Integer>
generateDescendingArea() ArrayList<Integer>
helperDescending(Node) void
generateAscendingArea() ArrayList<Integer>
helperAscending(Node) void
generateTotalHomesInPlace(ArrayList<Home>, String) Map<String, Integer>
compare(Home, Home) int
removeHome(int, SystemUserAbstract) Home
updateGraph() void
generateAdNumber() int
addHome(Home) ArrayList<Home>
ownerHome(SystemUserAbstract) ArrayList<Home>
sort(int[]) int[]
merge(int[], int[], int[]) void
ascendingArea ArrayList<Integer>
userArrayList ArrayList<SystemUserAbstract>
homeArrayList ArrayList<Home>
descendingArea ArrayList<Integer>
Node
data int
left Node
right Node
Node(int)
SystemUser
login(ArrayList<SystemUserAbstract>) boolean
name String
password String
userType String
email String
phoneNumber String
surname String
SystemUserAbstract
SystemUserAbstract(String, String, String, String, String, String)
toString() String
login(ArrayList<SystemUserAbstract>) boolean
name String
password String
userType String
email String
phoneNumber String
surname String
Member
savedHomes HashSet<Integer>
delimiter String
separator String
Member(String, String, String, String, String, String)
login(ArrayList<SystemUserAbstract>) boolean
registeration(ArrayList<SystemUserAbstract>) boolean
saveHome(Integer) void
writeToFileFavorites(String, Home) void
readFavorites() void
deleteFav(int) void
toString() String
favorites ArrayList
Admin
Admin(String, String, String, String, String, String)
login(ArrayList<SystemUserAbstract>) boolean
Graph
vertexList ArrayList<Vertex>
edgeList ArrayList<Edge>
result ArrayList<Home>
Graph()
add(T, T) void
add(T) void
findVertex(T) Vertex
toString() String
edgesToString() String
DepthFirstSearch(int) ArrayList<Home>
DepthFirstSearch(Vertex, int) ArrayList<Home>
main(String[]) void
connected boolean
Home
Home(String, String, String, Address, int, SystemUserAbstract, int, int, int)
toString() String
compareTo(Home) int
compareToWithDistance(Home) int
type String
adNumber int
owner SystemUserAbstract
countOfRoom String
status String
distance int
address Address
price int
area int
FileOperations
userList String
homeList String
delimiter String
separator String
FileOperations()
writeUserList(ArrayList<SystemUserAbstract>) void
writeHomeList(ArrayList<Home>) void
readUserList(ArrayList<SystemUserAbstract>) void
readHomeList(ArrayList<Home>, ArrayList<SystemUserAbstract>) void
userList String
homeList String
Vertex
Vertex(T)
addNeighbor(Vertex) void
toString() String
state State
value T
adjacentList ArrayList<Vertex>
State
UNVISITED
VISITED
COMPLETE
Edge
x Vertex
y Vertex
Edge(T, T)
toString() String
Address
Address(String, String, String)
toString() String
neighborhood String
country String
province String
BootStrappersTest
generateDescendingOrderedStack() void
displayPriorityQueue() void
generateTotalHomesInPlace() void
MainTester
start() void
main(String[]) void
MyComparator
compare(Home, Home) int