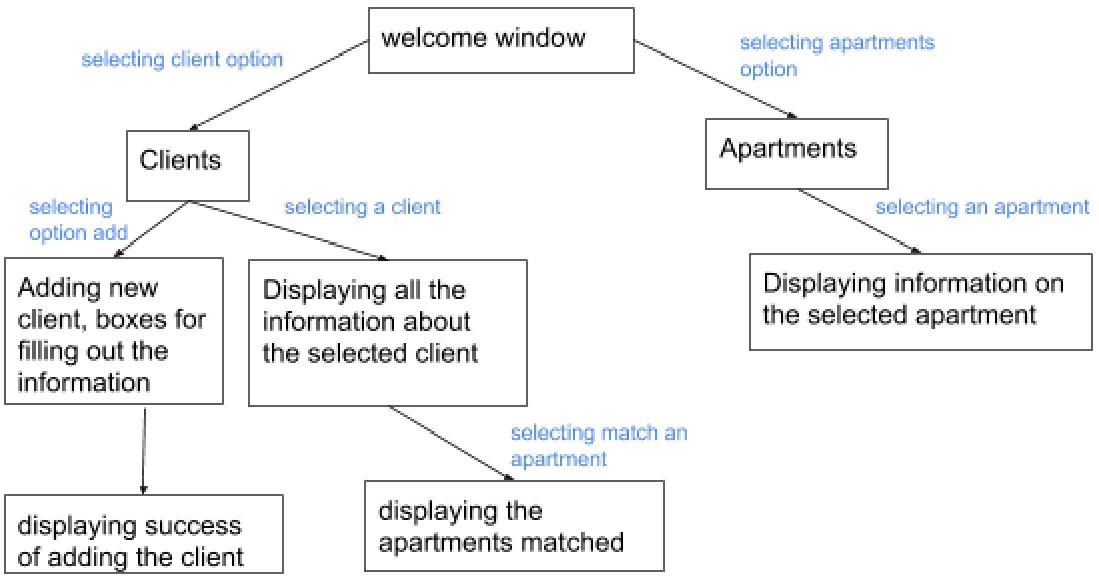
# Criterion B: Design

**An overview of the program**



welcome window
selecting client option
selecting apartments option
Clients
Apartments
selecting option add
selecting a client
selecting an apartment
Adding new client, boxes for filling out the information
Displaying all the information about the selected client
Displaying information on the selected apartment
selecting match an apartment
displaying success of adding the client
displaying the apartments matched

Fig. 1. Overview of the design of the program

Having a rough idea of how many windows I will need to use and what to include in them I proceeded to design them.

**Graphical visualisations of output**

The first thing that the client sees when running the program is the welcome window displayed below in figure 2.

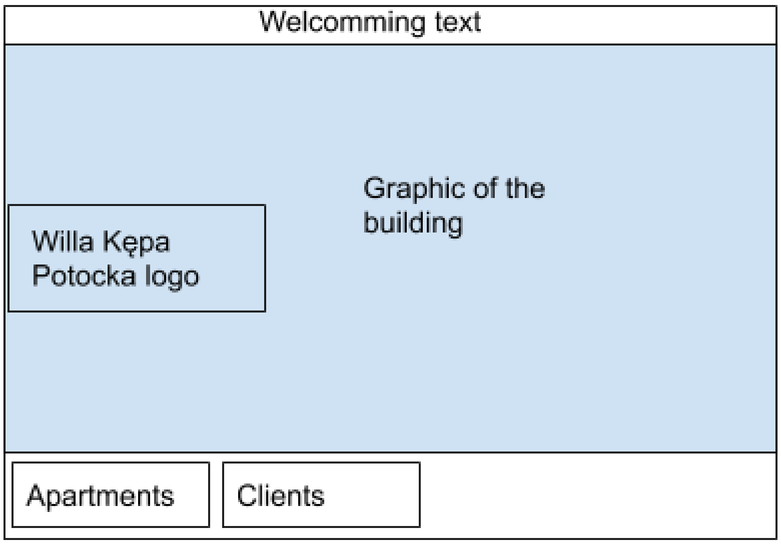


Fig. 2. Welcome window design

Selecting "APARTMENTS" in the menu in the welcome window results in a list of apartments showing.

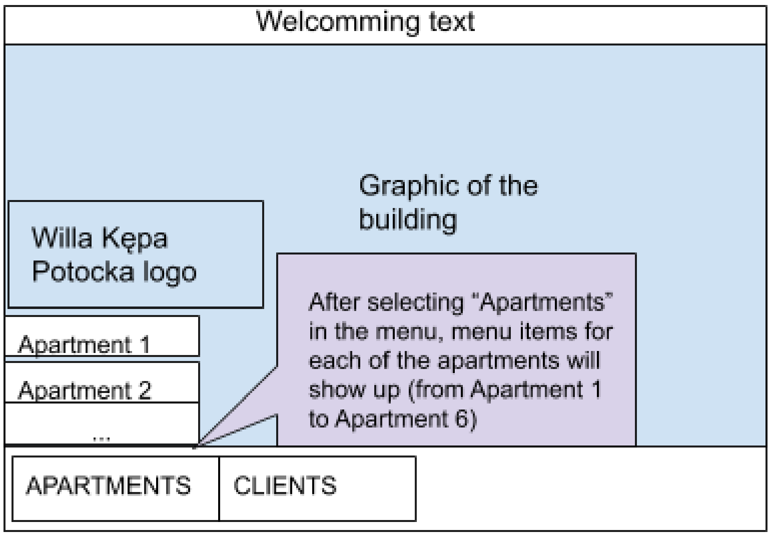

Welcomming text
Graphic of the building
Willa Kępa Potocka logo
Apartment 1
Apartment 2
...
After selecting "Apartments" in the menu, menu items for each of the apartments will show up (from Apartment 1 to Apartment 6)
APARTMENTS
CLIENTS

Fig. 3. Design of menu APARTMENTS with menu items for each apartment

The result of selecting "CLIENTS" in the menu in the welcome window is shown below in figure 4.

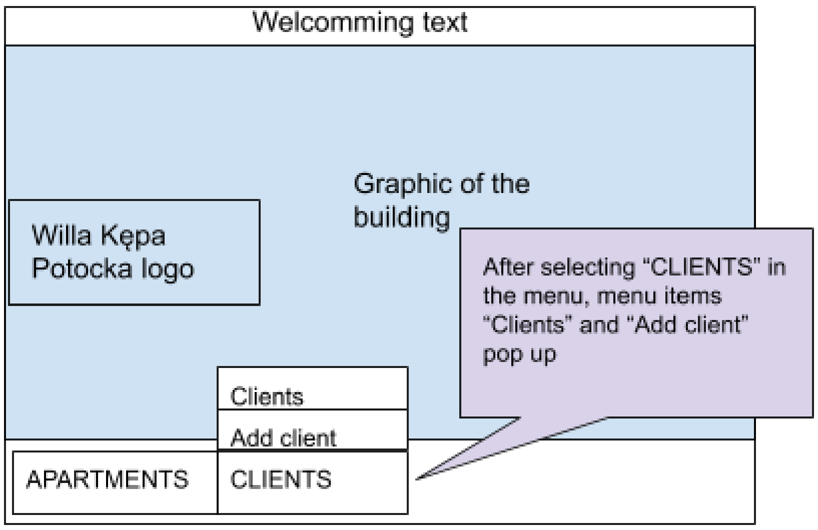


Fig. 4. Design of menu CLIENTS with menu items "Clients" and "Add client"

After selecting an apartment, for example "Apartment 1", the layout of the window of the chosen apartment is shown in figure 5.

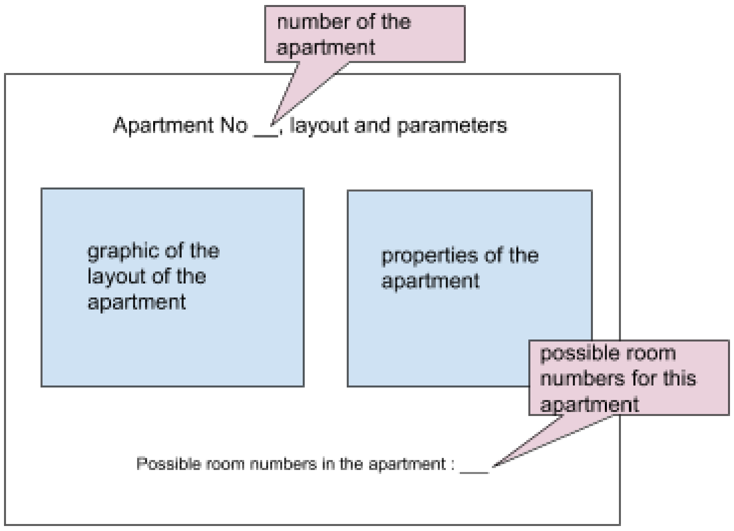

number of the apartment
Apartment No __, layout and parameters
graphic of the layout of the apartment
properties of the apartment
possible room numbers for this apartment
Possible room numbers in the apartment : ___

Fig. 5. Design of the window for displaying information about a selected apartment

After selecting "Add client" from the "CLIENTS" menu in the welcome window, the following one pops up.

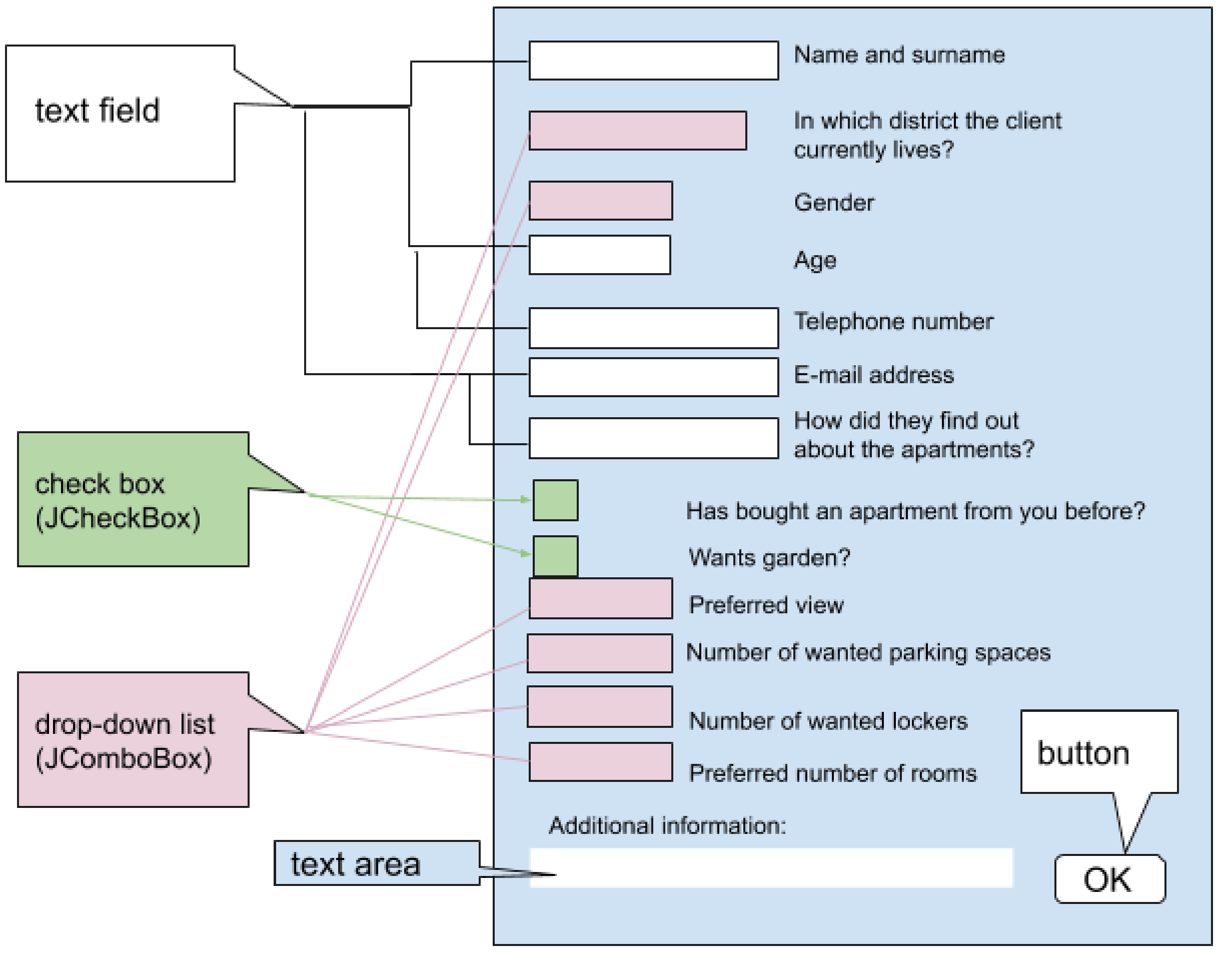

text field
check box (JCheckBox)
drop-down list (JComboBox)
text area
Name and surname
In which district the client currently lives?
Gender
Age
Telephone number
E-mail address
How did they find out about the apartments?
Has bought an apartment from you before?
Wants garden?
Preferred view
Number of wanted parking spaces
Number of wanted lockers
Preferred number of rooms
button
Additional information:
OK

Fig. 6. Design of the window for adding a client

After adding the client by clicking “OK”, if adding was successful the following window pops up.

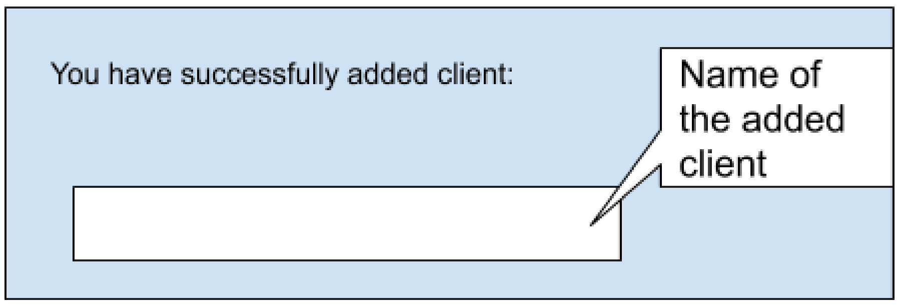


Fig. 7. Design of a window for  the confirmation of success in adding a client to the database

After selecting “Clients” from the “CLIENTS” menu, the window displayed in figure 8 shows up.

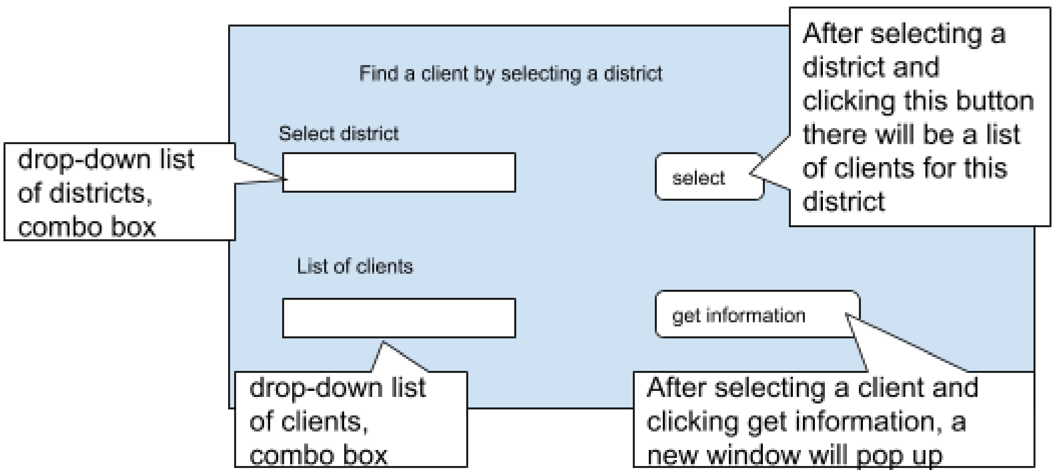

Find a client by selecting a district
Select district
select
List of clients
get information
drop-down list of districts, combo box
After selecting a district and clicking this button there will be a list of clients for this district
drop-down list of clients, combo box
After selecting a client and clicking get information, a new window will pop up

Fig. 8. Design of a window for listing clients in a selected district

After clicking the button “get information” in this window, all the information on the selected client will be displayed as illustrated below.

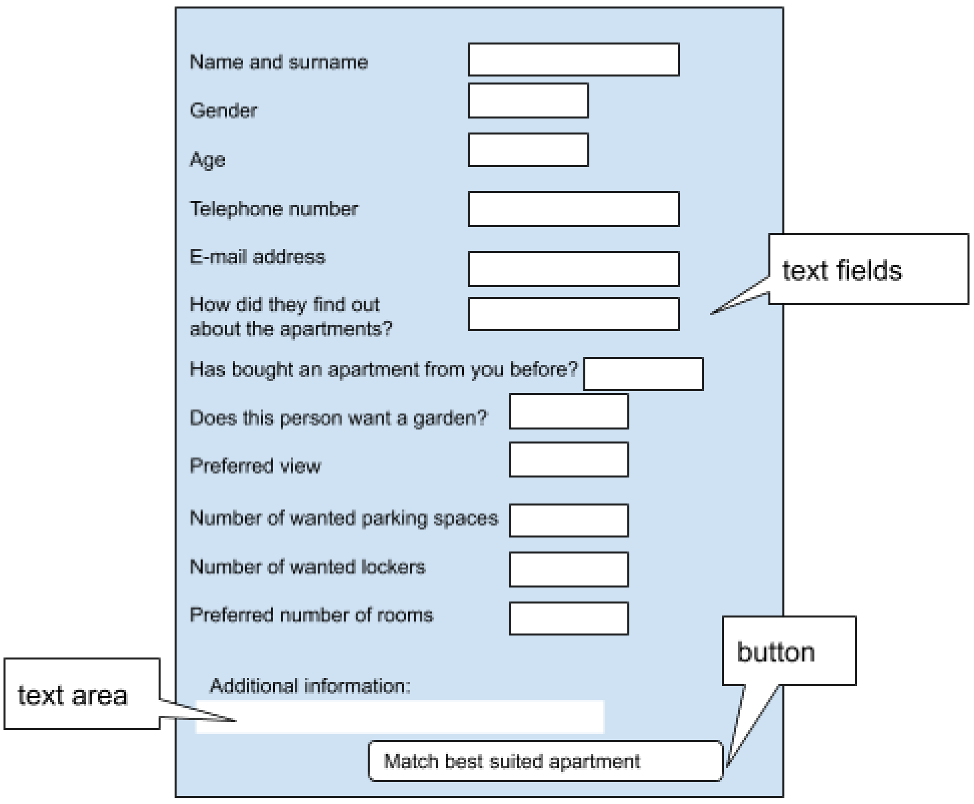

Name and surname
Gender
Age
Telephone number
E-mail address
How did they find out about the apartments?
Has bought an apartment from you before?
Does this person want a garden?
Preferred view
Number of wanted parking spaces
Number of wanted lockers
Preferred number of rooms
Additional information:
Match best suited apartment
text fields
text area
button

Fig. 9. Design of the window for displaying information on a selected client

After pressing the “Match best suited apartment” button in this window, the apartment(s) that is/are the best match for customer’s requirements/preferences is displayed in a window illustrated below.

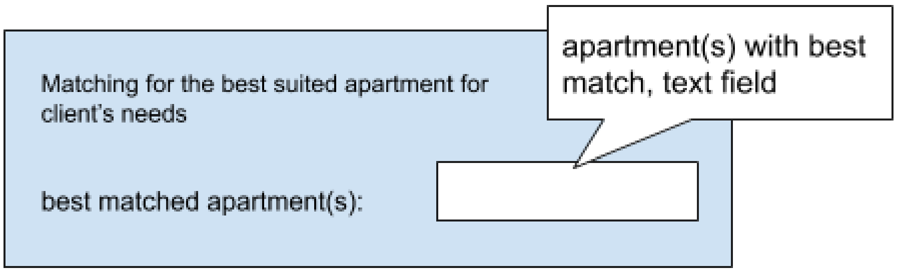
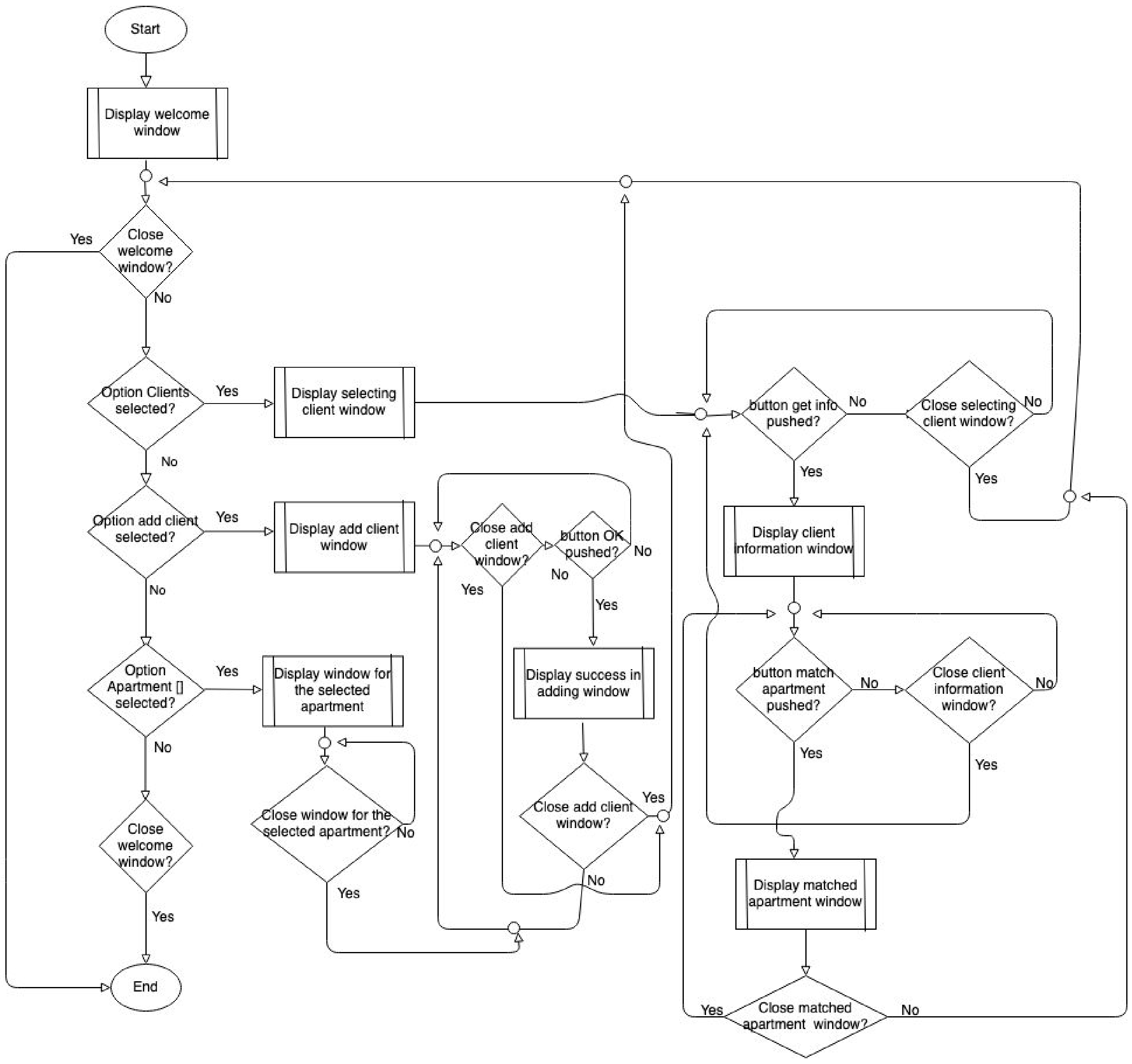

Matching for the best suited apartment for client’s needs
best matched apartment(s):
apartment(s) with best match, text field

Fig. 10. Design of the window for displaying the best apartment(s) for client’s requirements

**Overview of workings of the program in a flowchart**

Keeping in mind my design for the GUI interface I developed a flowchart for the overview of the workings of this program.



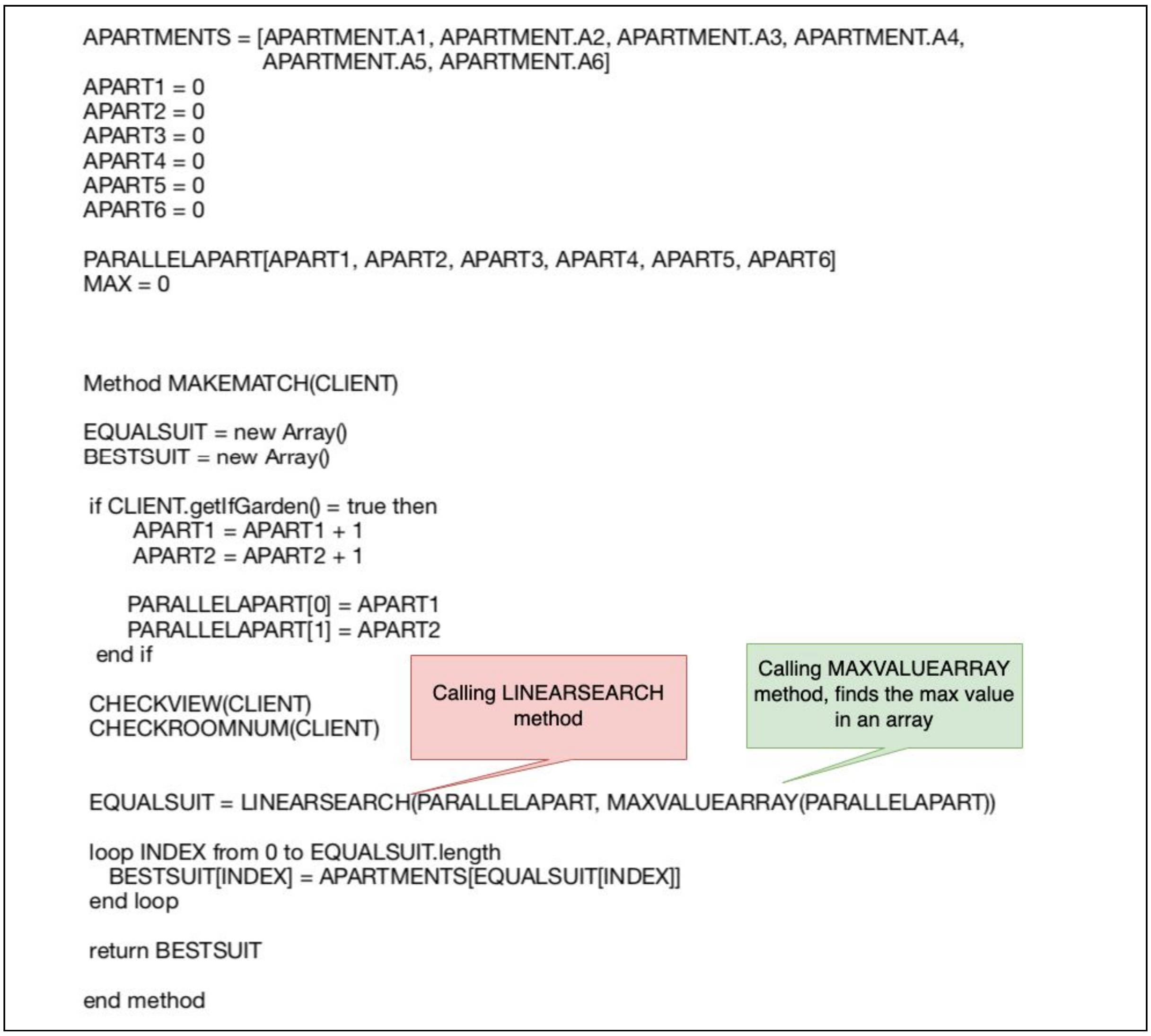
Start
Display welcome window
Close welcome window?
Yes
No
Option Clients selected?
Display selecting client window
Option add client selected?
Display add client window
Option Apartment [] selected?
Display window for the selected apartment
Close window for the selected apartment?
Close welcome window?
End
Close add client window?
button OK pushed?
Display success in adding window
Close add client window?
button get info pushed?
Close selecting client window?
Display client information window
button match apartment pushed?
Close client information window?
Display matched apartment window
Close matched apartment window?

Fig. 11. Flowchart of the program

**Explanation of key algorithms**

By doing the flowchart of the program I knew what some of the key algorithms will be. The key algorithm for my program is one for matching the best apartments(s) for customer's requirements. For this a provided pseudocode and a flowchart.

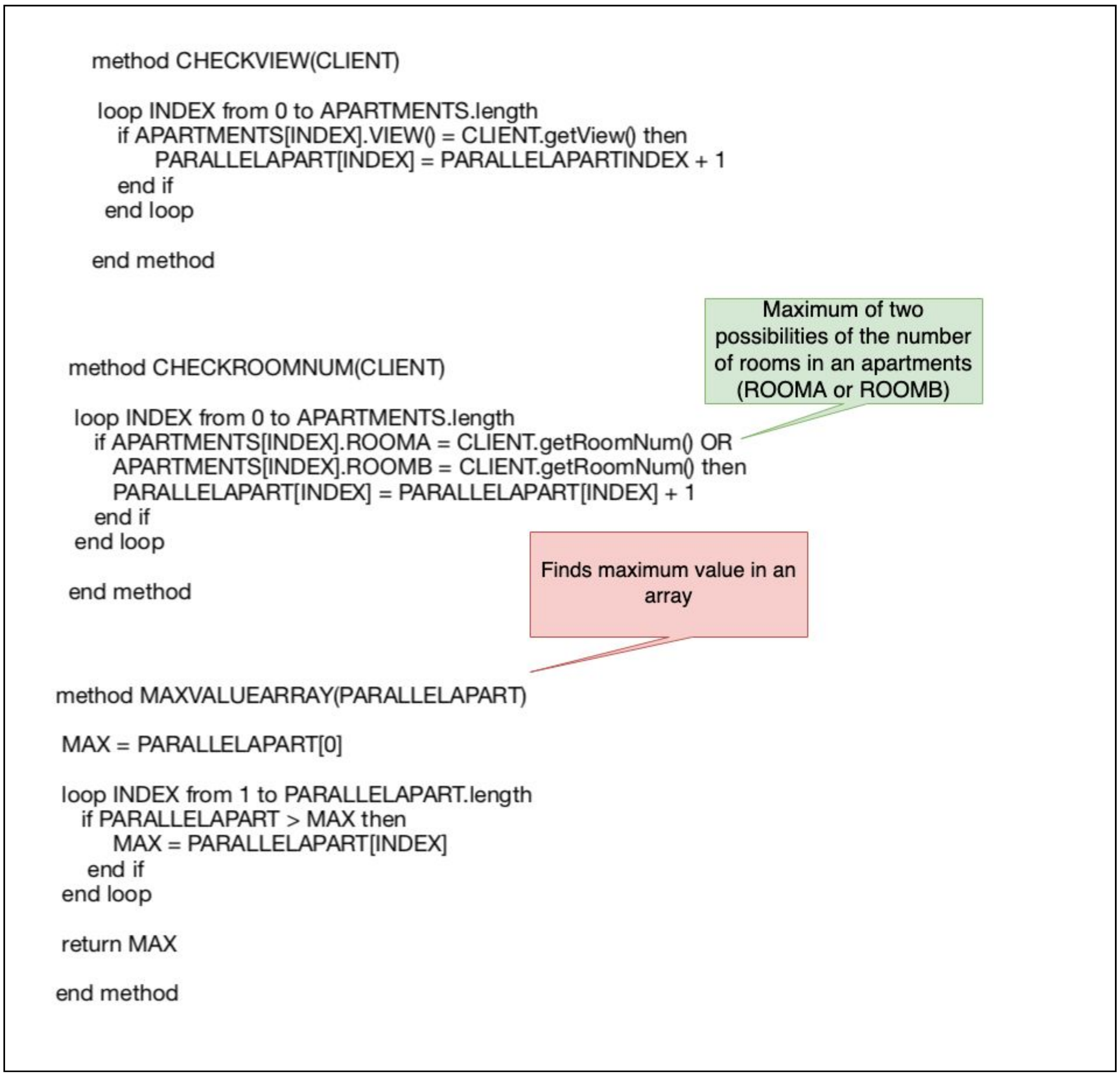
Workings of methods for matching the best apartment(s) for a selected client. Assuming provided accessor/mutator methods for already constructed objects CLIENT and APARTMENTS.



```
APARTMENTS = [APARTMENT.A1, APARTMENT.A2, APARTMENT.A3, APARTMENT.A4,
              APARTMENT.A5, APARTMENT.A6]
APART1 = 0
APART2 = 0
APART3 = 0
APART4 = 0
APART5 = 0
APART6 = 0

PARALLELAPART[APART1, APART2, APART3, APART4, APART5, APART6]
MAX = 0


Method MAKEMATCH(CLIENT)

EQUALSUIT = new Array()
BESTSUIT = new Array()

 if CLIENT.getIfGarden() = true then
     APART1 = APART1 + 1
     APART2 = APART2 + 1

     PARALLELAPART[0] = APART1
     PARALLELAPART[1] = APART2
  end if

 CHECKVIEW(CLIENT)
 CHECKROOMNUM(CLIENT)

 EQUALSUIT = LINEARSEARCH(PARALLELAPART, MAXVALUEARRAY(PARALLELAPART))

 loop INDEX from 0 to EQUALSUIT.length
    BESTSUIT[INDEX] = APARTMENTS[EQUALSUIT[INDEX]]
 end loop

 return BESTSUIT

end method
```

Calling LINEARSEARCH method
Calling MAXVALUEARRAY method, finds the max value in an array

Fig. 12. Pseudocode for MAKEMATCH method

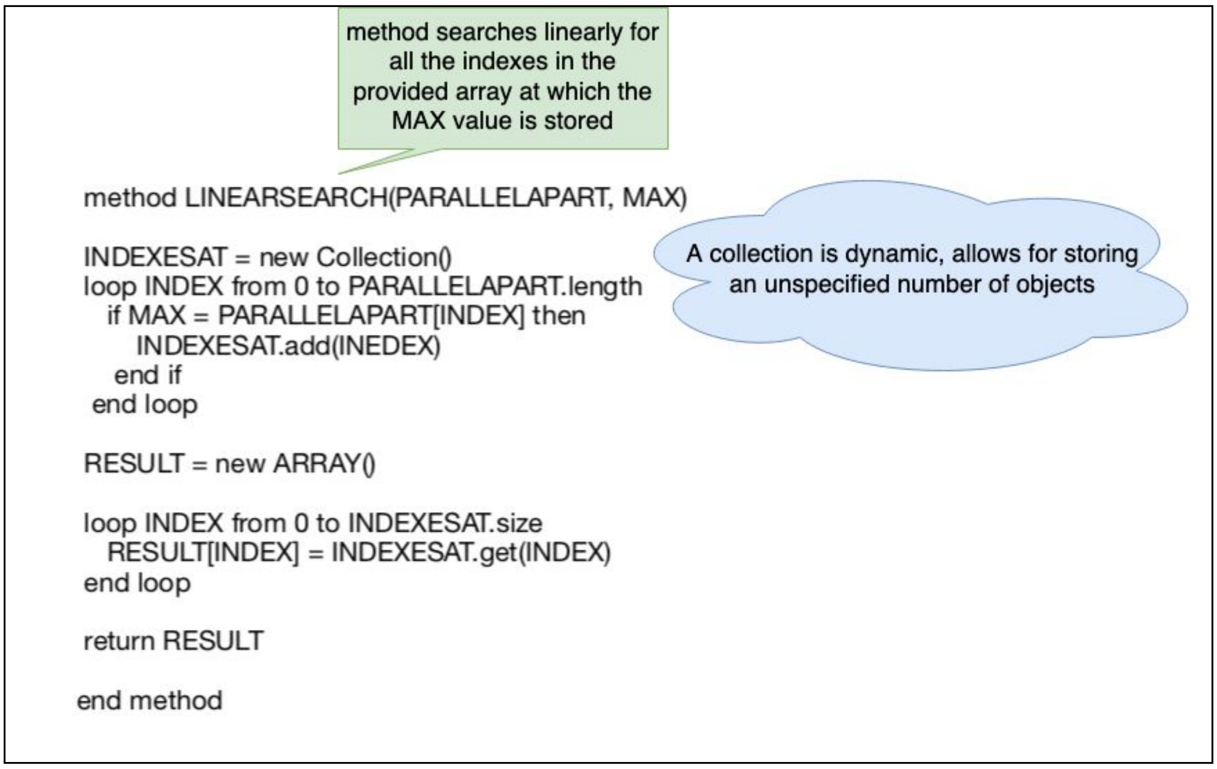


```
method CHECKVIEW(CLIENT)

 loop INDEX from 0 to APARTMENTS.length
    if APARTMENTS[INDEX].VIEW() = CLIENT.getView() then
         PARALLELAPART[INDEX] = PARALLELAPARTINDEX + 1
    end if
  end loop

end method


method CHECKROOMNUM(CLIENT)

 loop INDEX from 0 to APARTMENTS.length
   if APARTMENTS[INDEX].ROOMA = CLIENT.getRoomNum() OR
      APARTMENTS[INDEX].ROOMB = CLIENT.getRoomNum() then
      PARALLELAPART[INDEX] = PARALLELAPART[INDEX] + 1
   end if
 end loop

end method


method MAXVALUEARRAY(PARALLELAPART)

 MAX = PARALLELAPART[0]

 loop INDEX from 1 to PARALLELAPART.length
   if PARALLELAPART > MAX then
      MAX = PARALLELAPART[INDEX]
    end if
 end loop

 return MAX

end method
```

Maximum of two possibilities of the number of rooms in an apartments (ROOMA or ROOMB)

Finds maximum value in an array

Fig. 13. Pseudocode for methods for checking for the same qualities in an apartment as the client wants and finding max value in an array



```
method LINEARSEARCH(PARALLELAPART, MAX)

INDEXESAT = new Collection()
loop INDEX from 0 to PARALLELAPART.length
  if MAX = PARALLELAPART[INDEX] then
    INDEXESAT.add(INEDEX)
  end if
end loop

RESULT = new ARRAY()

loop INDEX from 0 to INDEXESAT.size
  RESULT[INDEX] = INDEXESAT.get(INDEX)
end loop

return RESULT

end method
```

method searches linearly for all the indexes in the provided array at which the MAX value is stored
A collection is dynamic, allows for storing an unspecified number of objects

Fig. 14. Pseudocode of method LINEARSEARCH

The flowchart for the MAKEMATCH(CLIENT) method is displayed in figure 15.

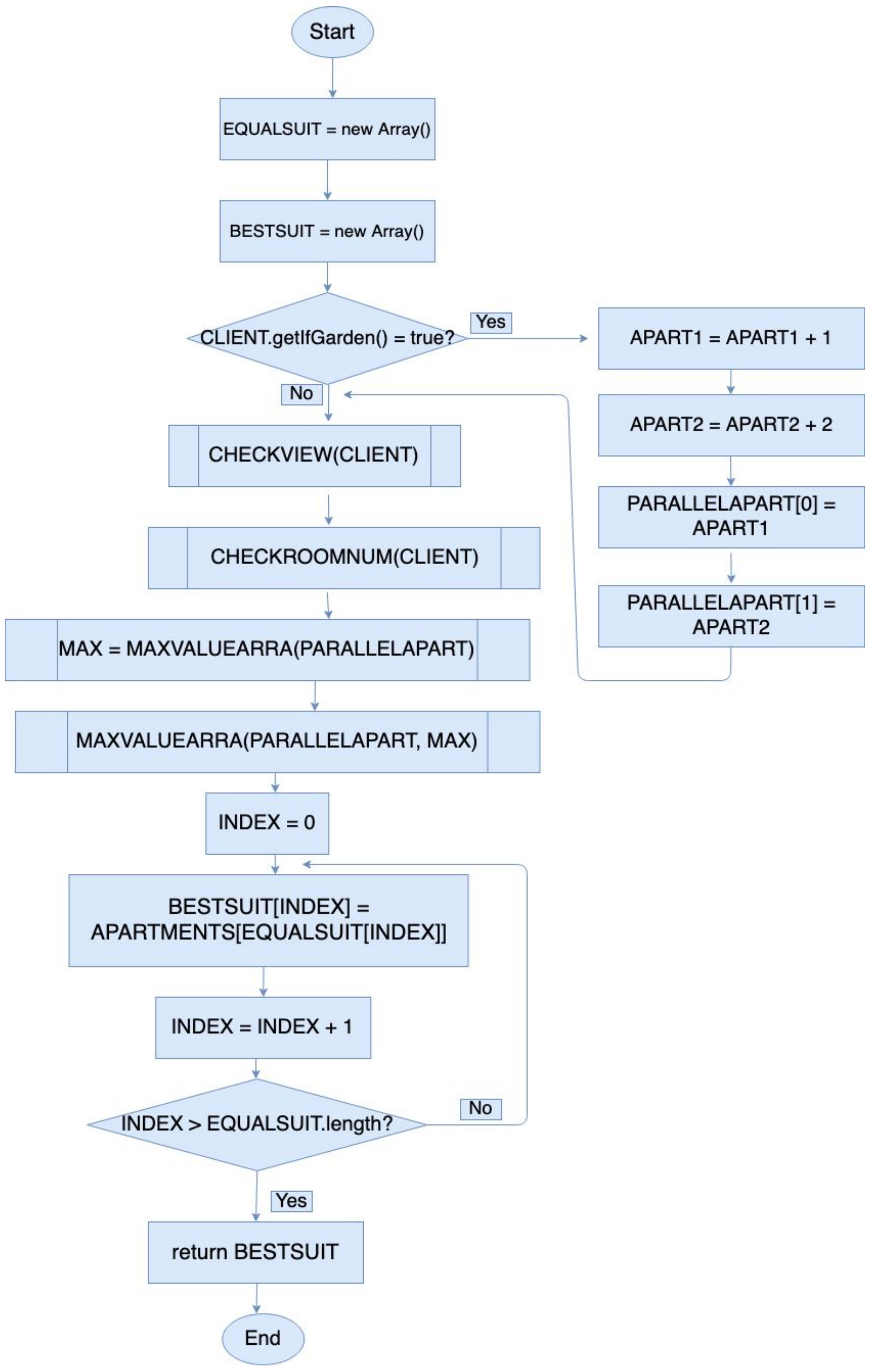

Start
EQUALSUIT = new Array()
BESTSUIT = new Array()
CLIENT.getIfGarden() = true?
Yes
APART1 = APART1 + 1
APART2 = APART2 + 2
PARALLELAPART[0] = APART1
PARALLELAPART[1] = APART2
No
CHECKVIEW(CLIENT)
CHECKROOMNUM(CLIENT)
MAX = MAXVALUEARRA(PARALLELAPART)
MAXVALUEARRA(PARALLELAPART, MAX)
INDEX = 0
BESTSUIT[INDEX] = APARTMENTS[EQUALSUIT[INDEX]]
INDEX = INDEX + 1
INDEX > EQUALSUIT.length?
No
Yes
return BESTSUIT
End

Fig. 15. Flowchart for the MAKEMATCH(CLIENT) method

**word count: 67**

**Test plan:**

| Actions tested | Nature of test | result |
|---|---|---|
| Welcome window with menu and graphic of the building | right-clicking on the class Gui, selecting method void Gui(String [] args) | welcome window pops up |
| Showing information on a selected apartment with a graphic of its layout | selecting an apartment from clicking on “APARTMENTS” in menu | A new window with the information on the selected apartment pops up |
| Adding a client into the database | selecting “CLIENTS” in the menu, then clicking on “Add client”, providing the information on the client | the client with all the provided information is saved into the database, written to the txt file |
| searching for a client in a selected district | selecting “CLIENTS” in the menu and then “Clients”. Selecting a district | listing all the clients from this district |
| displaying information on a selected client | After selecting the district, selecting the client from the list of clients names from this district | displaying the information on the selected client, read from the txt file |
| suggesting best-suited apartment(s) for clients requirements | when displaying the information on a selected client, pressing the button “Match best suited apartment” | a new window pops up listing the best suited apartment(s) |