GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework #3 Report

Hasan Mutlu
1801042673

1. SYSTEM REQUIREMENTS

Changes and additions from/to homework1 are highlighted.

Functional Requirements:

- ➔ User must be able to create a street in an allowed length.
- ➔ User must be able to add a building/playground by entering building's side, position, length and other properties. If the desired position is available and the building's data are valid, building must be created.
- ➔ User must be able to delete a building/playground. To delete a building, a basic list of buildings on the street shall be shown to the user and user should choose the side and address of the building he/she wants to delete.
- ➔ User must be able to display the total remaining length of lands on the street. The remaining length of lands on each side should also be displayed to give user a better understanding.
- ➔ User must be able to display the list of buildings on the street. There should be two types of lists, a basic one and a detailed one. User should choose which kind of list he/she wants to see. In basic list, only the types of buildings in each side should be displayed. In detailed list, all the properties of all the buildings should be displayed.
- ➔ User must be able to display the number and ratio of lenth of playgrounds in the street.
- ➔ User must be able to calculate the total length of street occupied by the markets, houses or offices.
- ➔ User must be able to display the skyline silhouette of the street.
- ➔ User must be able to focus on a building. To focus on a building, a basic list of buildings should be displayed to the user and the user should select which building he/she wants to focus by entering side and address of the building.
- ➔ User should be able to navigate between functionalities in a menu. The navigation should be provided by entering the figure of the desired option.
- ➔ User should be able to choose whether he/she wants to use the program with the default street created in driver function or create a new empty street.
- ➔ User should be able to see basic and detailed list of deleted buildings. This functionality should be provided in view mode in the menu.

Non-Functional Requirements:

- ➔ Implementation: The program shall be implemented using VSCode, Ubuntu 18.04 WSL and Java 11. 3 different versions of the program shall be implemented.
- ➔ Compiling an Running: All 3 seperate programs should be compiled and run with following commands:

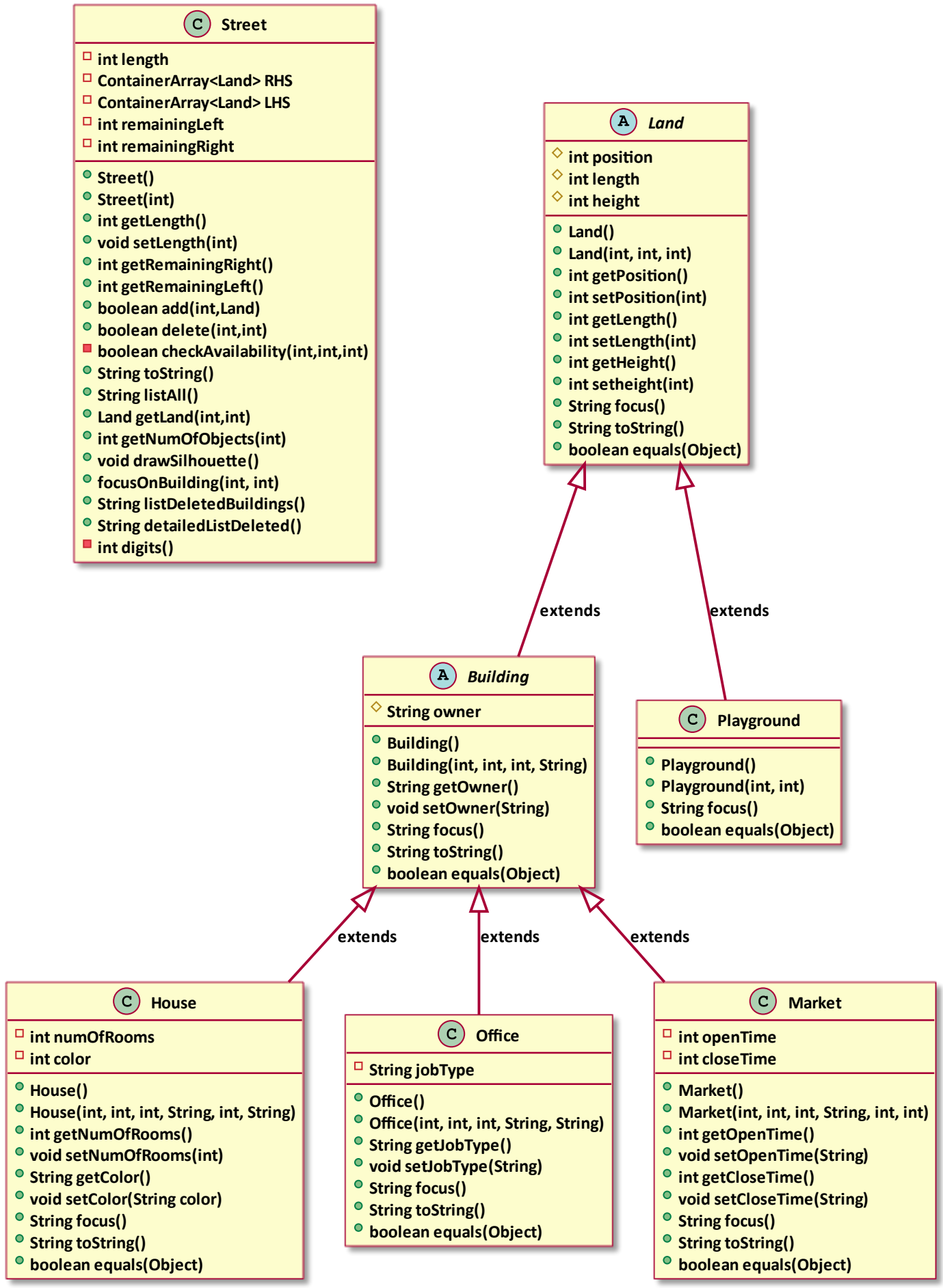
```
-$ javac *.java Main.java
-$ java Main
```

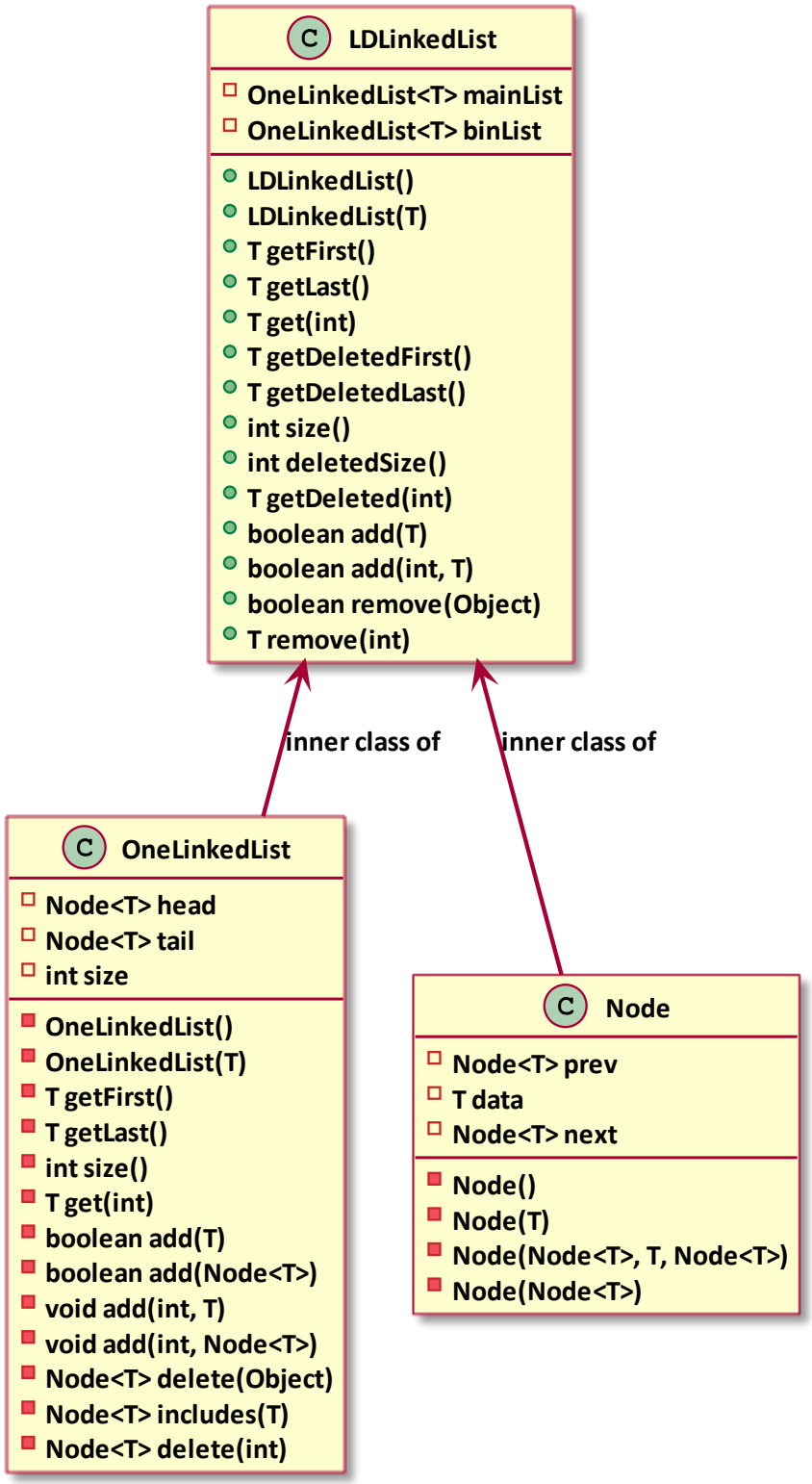
- ➔ Maintainability: The program should be maintainable in order to add new features in the future if necessary.
- ➔ Efficiency: The program should be efficient, should have high performance and shouldn't consume too much memory. It should allocate and deallocate memory when new buildings are added and deleted. For part 3, the LDLinkedList's functionalities should work efficiently.
- ➔ Reliable: The program should be reliable. It's mechanisms should work successfully in any combination of use. It should check the validity of inputs and when there is an unexpected situation, program should handle it instead of stopping.
- ➔ Compatibility: The program should be compatible with the facilities of the terminal interface. To achive this, some functionalities may be restricted. For example, the maximum length of the street should be limited in order to fit the skyline silhouette in the terminal.

2. USE CASE AND CLASS DIAGRAMS

Use Case Diagram:

Class Diagrams:





3. OTHER DIAGRAMS

4. PROBLEM SOLUTION APPROACH

First two parts of the homework were really easy for me. Since I have already implemented my own abstract data type, which is ContainerArray, in first homework, I only needed to replace its decleration with ArrayList and LinkedList. I also needed to change Street class' toString() method and that was it.

For third part, I struggled a little bit about identifying the problem. When I searched what lazy deletion is on the internet, the results were a little different than what was explained in the homework PDF and it confused me a little. But what is expected from me is the one in the PDF, of course. So I started implementing the LDLinkedList.

At first, I tried to keep two separate linked list heads inside LDLinkedList class, one as main list and other for keeping deleted items. But since these two lists have very same functionalities, it was leading me to a lot of code duplications. So I decided to implement an inner double linked list class, that would not only allow adding new objects but also would allow adding previously created nodes too. By this way, the deleted and re-added nodes would be transferred from one list to another. After implementing this inner class, which is named OneLinkedList, I created two OneLinkedLists as LDLinkedList fields. One is the main list and the other is list of deleted items. Then I implemented LDLinkedList's methods. I didn't implement every method from AbstractList as not all of them would be used in this homework's test scenario. I designed my classes maintainable so these methods can be implemented if needed in the future.

5. TEST CASES

First 30 test cases are same with homework 1. Since the homework has 3 parts, I only included third part's test cases.

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Create street	Length: 100	No error messages	As Expected	Pass
2	Add Playground	Side: 1 (Left) Position: 0 Length: 20	"New Playground is added" message	As Expected	Pass
3	Add House	Side: 2 (Right) Position: 0 Length: 15 Height: 20 Owner: "Ali Bey" Number of Rooms: 5 Color: "Green"	"New House is added." message	As Expected	Pass
4	Add Office	Side: 1 (Left) Position: 25 Length: 25 Height: 25 Owner: "Veli Bey" Job Type: "Law Firm"	"New Office is added." message	As Expected	Pass

5	Add Market	Side: 2 (Right) Position: 15 Length: 10 Height: 10 Owner: Zeynep Hanim Opening Time: 10 Closing Time: 21	"New Market is added." message	As Expected	Pass
6	Add Playground	Side: 1 (Left) Position: 55 Length: 20	"New Playground is added" message	As Expected	Pass
7	Add House	Side: 1 (Left) Position: 80 Length: 20 Height: 25 Owner: Ahmet Bey Number of Rooms: 7 Color: Blue	"New House is added." message	As Expected	Pass
8	Add Office	Side: 2 (Right) Position: 35 Length: 25 Height: 40 Owner: Ayse Hanim Job Type: Technology Company	"New Office is added." message	As Expected	Pass
9	Add Market	Side: 2 (Right) Position: 65 Length: 20 Height: 15 Owner: Selim Bey Opening Time: 8 Closing Time: 16	"New Market is added." message	As Expected	Pass
10	Add Playground	Side: Left Position: 10 Length: 20 Error: This position is already occupied. This place is already occupied.	"This place is already occupied." message	As Expected	Pass
11	Add House on an already occupied land.	Side: Left Position: 50 Length: 20 Height: 25 Owner: Noone Number of Rooms: 7 Color: Blue	"This place is already occupied." message	As Expected	Pass
12	Add Invalid Playground: Goes beyond the street.	Side: Right Position: 85 Length: 30	"Error: The building is out of bounds. This place is out of bounds." messages	As Expected	Pass
13	Add Playground that starting position is out of the street.	Side: Left Position: 110 Length: 10	"Error: The building is out of bounds. This place is out of bounds." messages	As Expected	Pass

14	Delete Playground	Side: 1(Left) Address: 3	"Building removed." Message	As Expected	Pass
15	Delete Market	Side: 2 Address: 2	"Building removed." Message	As Expected	Pass
16	Display street before and after deletion	Existing Street	Deleted buildings doesnt show up	As Expected	Pass
17	Deleting non-existing building	Side: 2(Right) Address: 5	"Error: Address info is invalid. Invalid building address." messages	As Expected	Pass
18	Deleting non-existing building	Side: 1 (Left) Address: 0	"Error: Address info is invalid. Invalid building address." messages	As Expected	Pass
19	Displaying the total remaining length of lands on the street	Existing Street	Data are 100, 15, 30, 45	As Expected	Pass
20	Basic listing of the buildings	Existing Street	Buildings are listed correctly	As Expected	Pass
21	Detailed listing of the buildings	Existing Street	Buildings are listed correctly	As Expected	Pass
22	Displaying the number and ratio of lenth of playgrounds on the street	Existing Street	Data are 1, 20, 0.1	As Expected	Pass
23	Calculating the total length of street occupied by the markets, houses or offices	Existing Street	Data are 35, 50, 20	As Expected	Pass
24	Draw skyline silhouette	Existing Street	Silhouette is drawen correctly	As Expected	Pass
25	Focusing on a building	All side and address combination of buildings on the street (2x3)	The data are given correctly according to building types	As Expected	Pass
26	Focusing with invalid info	Side: 1(Left) Address: 0	"Error: Invalid building info." message	As Expected	Pass
27	Focusing with invalid info	Side: 1(Left) Address: 5	"Error: Invalid building info." message	As Expected	Pass
28	Navigate in all menu options	Menu selections	Navigate without issue	As Expected	Pass
29	Test all functionalities above in menu as well.	Menu selections and inputs	Functionalities work without issue	As Expected	Pass
30	Enter invalid string type inputs In main menu	string	Program acts if 0 is entered	As Expected	Pass
31	Display basic list of deleted buildings	Existing Street	Basic list of deleted buildings is displayed	As Expected	Pass
32	Display detailed list of deleted buildings	Existing Street	Detailed list of deleted buildings is displayed	As Expected	Pass
33	Add same exact buildings from deleted buildings list	Building data from detailed lists	Buildings are added	As Expected	Pass
34	Display basic list of deleted buildings again to see if they are transferred to the main list.	Existing Street	Deleted buildings list is empty	As Expected	Pass

6. RUNNING AND RESULTS

Since this homework is about data structures, I didn't added the Street class specific results in this part. They are same with previous homework anyway. I only added add and delete related results. Results in these page are same in all versions of the homework.

```
HW3 Driver Function
Test setting street size.
Entered street size: 100

--------------------------------
Test Adding Buildings
Add Playground
Side: Left
Position: 0
Length: 20
New Playground is added.

Add House
Side: Right
Position: 0
Length: 15
Height: 20
Owner: Ali Bey
Number of Rooms: 5
Color: Green
New House is added.

Add Office
Side: Left
Position: 25
Length: 25
Height: 25
Owner: Veli Bey
Job Type: Law Firm
New Office is added.

Add Market
Side: Right
Position: 15
Length: 10
Height: 10
Owner: Zeynep Hanim
Opening Time: 10
Closing Time: 21
New Market is added.
```

```
Add Playground
Side: Left
Position: 55
Length: 20
New Playground is added.

Add House
Side: Left
Position: 80
Length: 20
Height: 25
Owner: Ahmet Bey
Number of Rooms: 7
Color: Blue
New House is added.

Add Office
Side: Right
Position: 35
Length: 25
Height: 40
Owner: Ayse Hanim
Job Type: Technology Company
New Office is added.

Add Market
Side: Right
Position: 65
Length: 20
Height: 15
Owner: Selim Bey
Opening Time: 8
Closing Time: 16
New Market is added.
--------------------------------
```

```
Detailed Listing:

Left side of the street:
--Playground--
Position: 0
Length: 20
Default height: 2

--Office--
Position: 25
Length: 25
Height: 25
Owner: Veli Bey
Job Type: Law Firm

--House--
Position: 80
Length: 20
Height: 25
Owner: Ahmet Bey
Number of Rooms: 7
Color: Blue
```

```
Right side of the street:
--House--
Position: 0
Length: 15
Height: 20
Owner: Ali Bey
Number of Rooms: 5
Color: Green

--Office--
Position: 35
Length: 25
Height: 40
Owner: Ayse Hanim
Job Type: Technology Company

--Market--
Position: 65
Length: 20
Height: 15
Owner: Selim Bey
Opening Time: 8
Closing Time: 16
```

```
--------------------------------
Test Deleting Building

Left side of the street:
1: Playground
2: Office
3: Playground
4: House

Right side of the street:
1: House
2: Market
3: Office
4: Market


Remove Playground(3) on left.
Building removed.
Remove Market(2) on right
Building removed.

After deletion:

Left side of the street:
1: Playground
2: Office
3: House

Right side of the street:
1: House
2: Office
3: Market

--------------------------------
```

Next results are only from third part of the homework. They are to show that deleted objects are kept in a separate linked list and when these objects are re-added to the main list, they aren't created again, they are transferred from deleted list to main list.

```
------------------------------------------------
List of deleted buildings:

Buildings removed from left side of the street:
1: Playground

Buildings removed from right side of the street:
1: Market

------------------------------------------------
```

```
------------------------------------------------
Detailed list of deleted buildings:

Buildings deleted from left side of the street:
--Playground--
Position: 55
Length: 20
Default height: 2

Buildings deleted from right side of the street:
--Market--
Position: 15
Length: 10
Height: 10
Owner: Zeynep Hanim
Opening Time: 10
Closing Time: 21
```

You can see deleted buildings here.

```
------------------------------------------------------------
Add deleted buildings again and list again:
Add the deleted Market(15, 10, 10, "Zeynep Hanim", 10, 21)
Add the deleted playground Playground(55, 20)
Detailed list of street:

Left side of the street:
--Playground--
Position: 0
Length: 20
Default height: 2

--Office--
Position: 25
Length: 25
Height: 25
Owner: Veli Bey
Job Type: Law Firm

--House--
Position: 80
Length: 20
Height: 25
Owner: Ahmet Bey
Number of Rooms: 7
Color: Blue

--Playground--
Position: 55
Length: 20
Default height: 2
```

```
Right side of the street:
--House--
Position: 0
Length: 15
Height: 20
Owner: Ali Bey
Number of Rooms: 5
Color: Green

--Office--
Position: 35
Length: 25
Height: 40
Owner: Ayse Hanim
Job Type: Technology Company

--Market--
Position: 65
Length: 20
Height: 15
Owner: Selim Bey
Opening Time: 8
Closing Time: 16

--Market--
Position: 15
Length: 10
Height: 10
Owner: Zeynep Hanim
Opening Time: 10
Closing Time: 21
```

Here you can see the deleted buildings are re-added to the list.

```
Detailed list of deleted buildings:

Buildings deleted from left side of the street:
Buildings deleted from right side of the street:

------------------------------------------------
```

And list of deleted objects is emptied.

7. TIME COMPLEXITY ANALYSIS

1.) Homework 1 (ContainerArray) Version Analysis

```
public void add(T obj){
    T[] temp = (T[]) new Object[++size];

    for(int i = 0; i < size - 1; i++){
        temp[i] = arr[i];
    }

    temp[size-1] = obj;
    arr = (T[]) new Object[size];

    for(int i = 0; i < size; i++){
        arr[i] = temp[i];
    }
}
```

Adding New Element = O(n)

```
public boolean delete(int ind){
    if(ind < 0 || ind > size - 1 || size < 0){
        System.out.println("Error: Incorrect index.");
        return false;
    }
    for(int i = ind; i < size - 1; i++){
        arr[i] = arr[i+1];
    }
    arr[--size] = null;
    return true;
}
```

Deleting Element = O(n)

```
public T get(int i){
    return this.arr[i];
}
```

Access Element = O(1)

2.) ArrayList Version Analysis

Adding New Element = O(n)

Deleting Element = O(n)

Accessing Element = O(1)

3.) LinkedList Version Analysis

Adding New Element = O(1)

Deleting Element = O(1)

Accessing Element = O(n)

4.) LDLinkedList Version

```
private void add(int index, T obj){
    if(index < 0 || index > size){
        throw new IndexOutOfBoundsException(Integer.toString(index));
    }
    if(index == 0){
        head.prev = new Node<T>(null, obj, head);
        head = head.prev;
        size++;
        if(size == 0){
            head.next = null;
            tail = head;
        }
        return;
    }
    if(index == size){
        tail.next = new Node<T>(tail, obj, null);
        tail = tail.next;
        size++;
        return;
    }
    Node<T> iter = head;
    for(int i = 0; i < index && iter != null; i++){
        iter = iter.next;
    }
    Node<T> temp = new Node<T>(iter.prev, obj, iter);
    iter.prev.next = temp;
    iter.prev = temp;
    size++;
}
```

Adding New Element

Best Case = O(1)

Worst Case = O(n)

In this homework, index is always equal to size so it works in best case, which is O(1).

```
private Node<T> delete(Object obj){
    Node<T> iter = head;
    if(obj.equals(head.data)){
        Node<T> temp = head;
        head = head.next;
        //head.prev = null;
        size--;
        return temp;
    }
    if(obj.equals(tail.data)){
        Node<T> temp = tail;
        tail = tail.prev;
        tail.next = null;
        size--;
        return temp;
    }
    for(int i = 0; i < size && iter != null; i++){
        if(obj.equals(iter.data)){
            iter.next.prev = iter.prev;
            iter.prev.next = iter.next;
            size--;
            return iter;
        }
        iter = iter.next;
    }
    return null;
}
```

Deleting Element

Best Case = O(1)

Worst Case = O(n)

```
private T get(int ind){
    if(ind < 0 || ind >= size){
        throw new IndexOutOfBoundsException(Integer.toString(ind));
    }
    Node<T> iter = head;
    for(int i = 0; i < ind; i++){
        iter = iter.next;
    }
    return iter.data;
}
```

Accessing Element: O(n)

Experimental Running Time of Driver Functions:

Homework 1 (ContainerArray) version:

```
Driver function ended.

Running time of driver function in Homework 1 (ContainerArray) version is 450376000 nanoseconds.
```

ArrayList version:

```
Driver function ended.

Running time of driver function in ArrayList version is 273341100 nanoseconds.
```

LinkedList version:

```
Driver function ended.

Running time of driver function in LinkedList version is : 359972400 nanoseconds.
```

LDLinkedList version:

```
Driver function ended.

Running time of driver function in LDLinkedList version is 261551600 nanoseconds.
```

LDLinkedList implementation seems to be slightly faster than ArrayList implementations. LinkedList comes next and ContainerArray implementation is by far the slowest.