GTU Department of Computer Engineering

CSE 222/505 - Spring 2022

Homework 3 Report

Sena Özbelen
1901042601

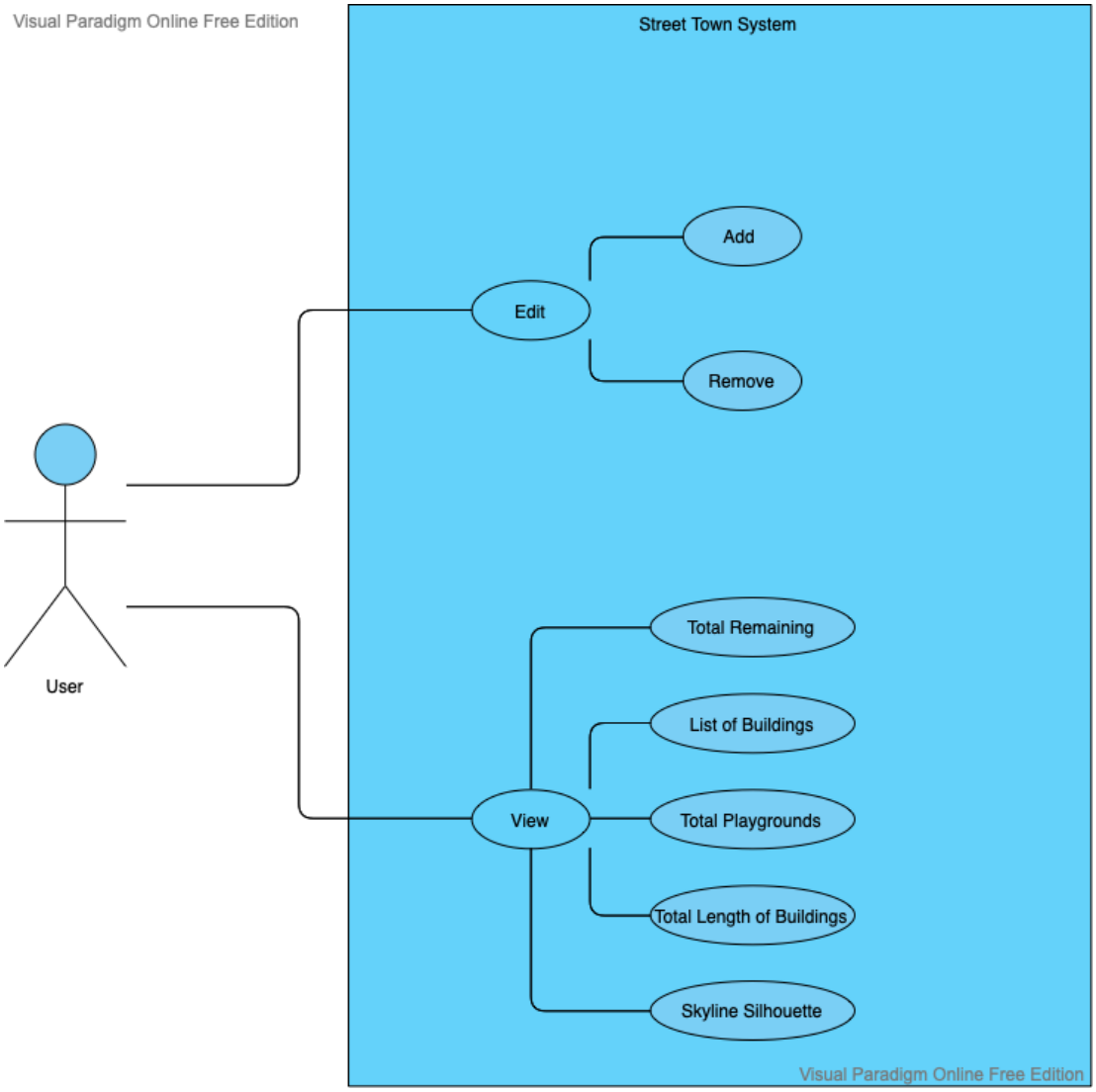
1. System Requirements

The system has 2 modes: edit and view. To access these modes, Town should be created using Town constructor with a parameter which is the length of the street.

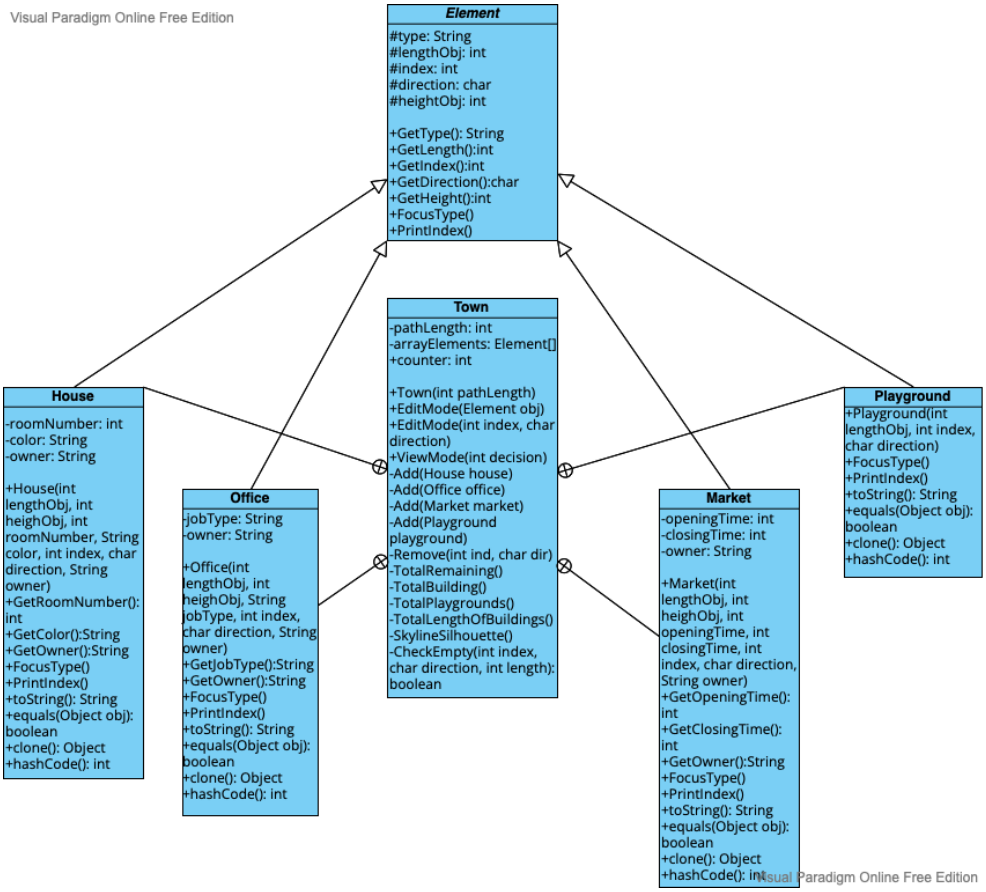
For edit mode,
- Addition or removing should be chosen first.
- To add a new element, the element should be created using constructors and sent as a parameter so that Add and EditMode methods can be overloaded.
- To remove an element, the index and the direction of the element should be given to the EditMode method.
- For both options, if the input is not valid, then it will be caught by the try-catch blocks so the program asks you to give the proper input again. Asking again is only for the user interactive part. Driver method can only catch the exceptions with try-catch blocks.
- Constructors should be called with right parameters. Otherwise, it will fail.

For view mode,
- The option among 5 options should be given to the method.

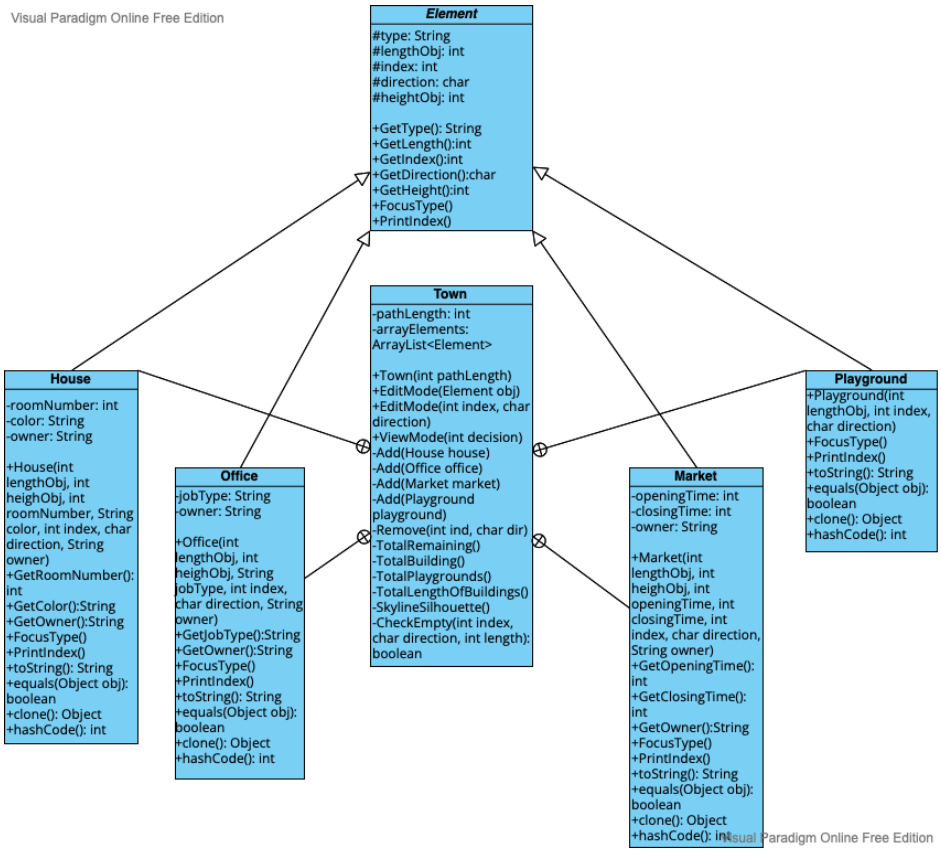
2. Use Case Diagram



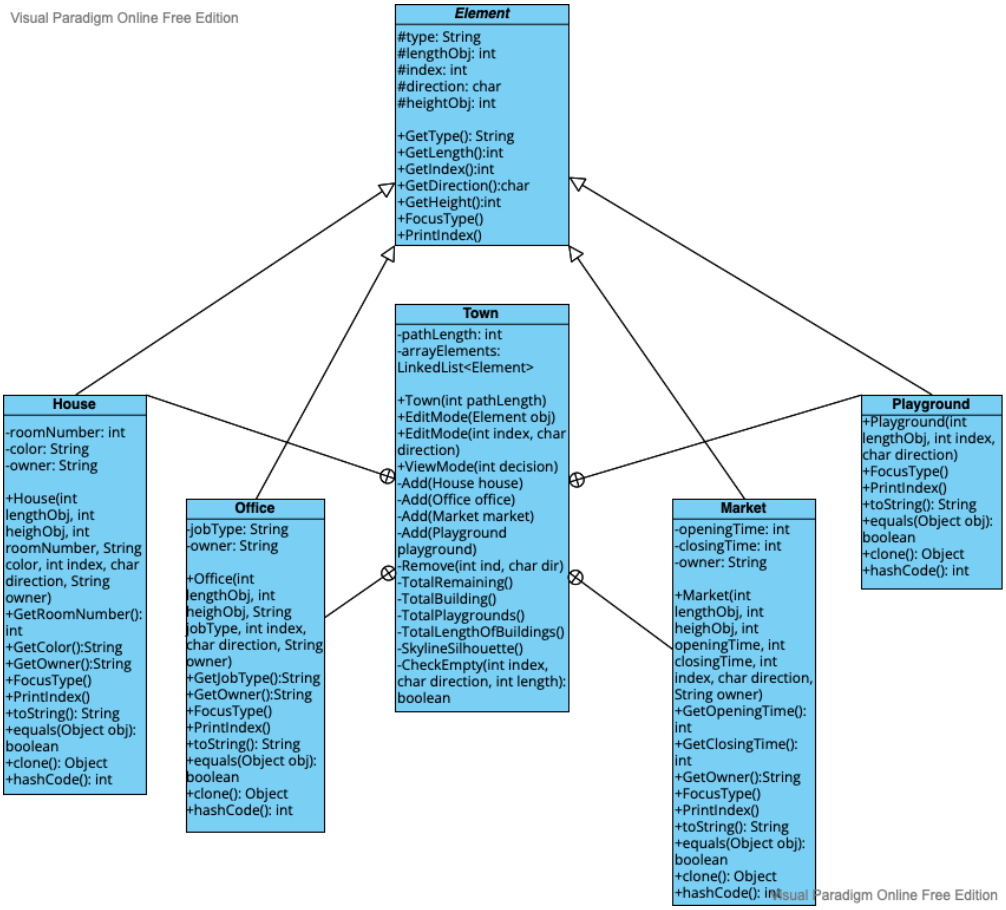
3. Class Diagrams



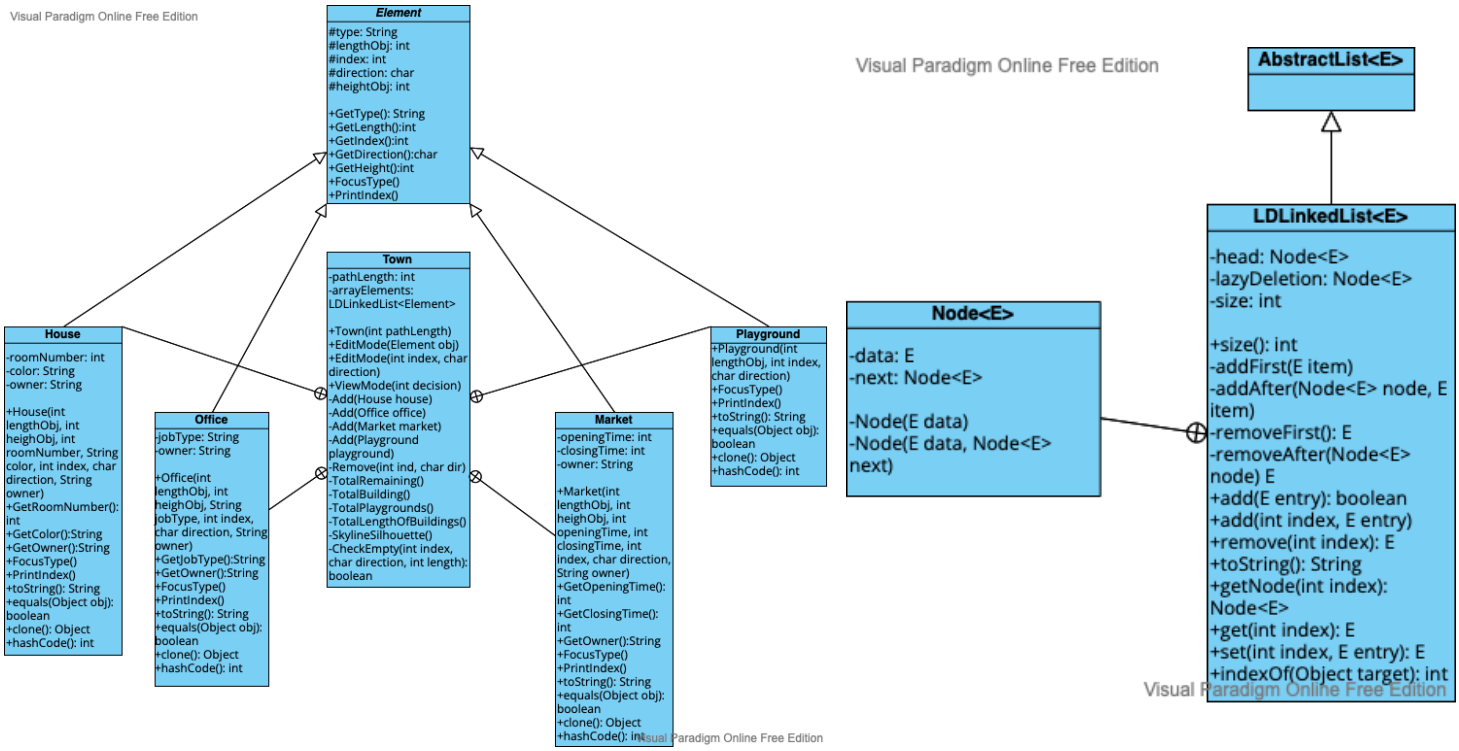
BasicArrayModified UML Class Diagram



ArrayList UML Class Diagram



LinkedList UML Class Diagram



LDLinkedList and Node UML Class Diagram

4.Problem Solution Approach

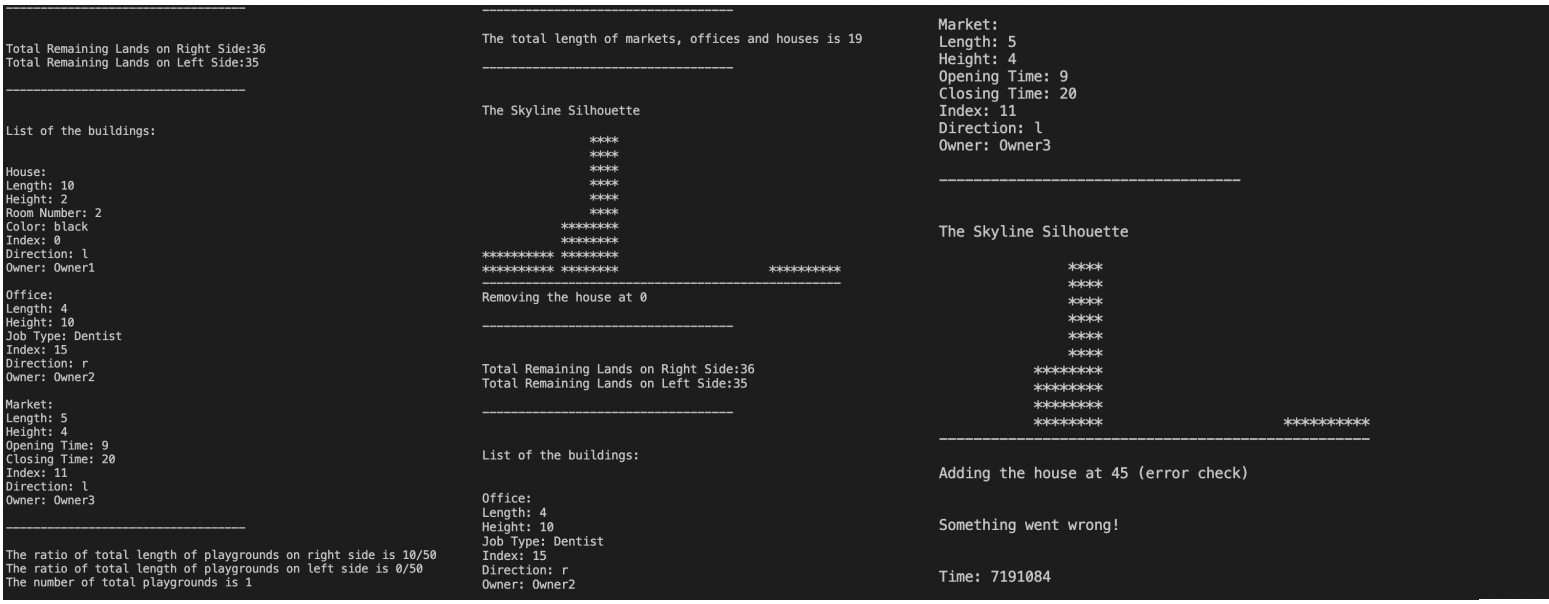
In this homework, we need to implement a class which contains the elements of a street town. Since these elements have a lot in common, I used a superclass to drive all the elements including buildings and playground. This class named “Element” has common features such as length of the element, the starting index of the element, etc and common methods such as getters, focusing mode, etc. The main class named “Town” consists of all the methods and elements. This class has four inner classes for each element and they are inherited from the superclass “Element”. Additionally, I implemented this program using ArrayList, LinkedList and my own LinkedList with ListIterator. In LDLinkedList, AbstractList class is extended so that this class can access the methods of AbstractList class and can override them.

This program have 2 modes: edit and view. Edit mode allows the user to add and remove the elements. I implemented add method for each element by overloading it. Remove method is based on the index and it removes the element at the given index and direction. All errors are handled in try-catch blocks by using exceptions. (InputMismatchException and ArrayIndexOutOfBoundsException)

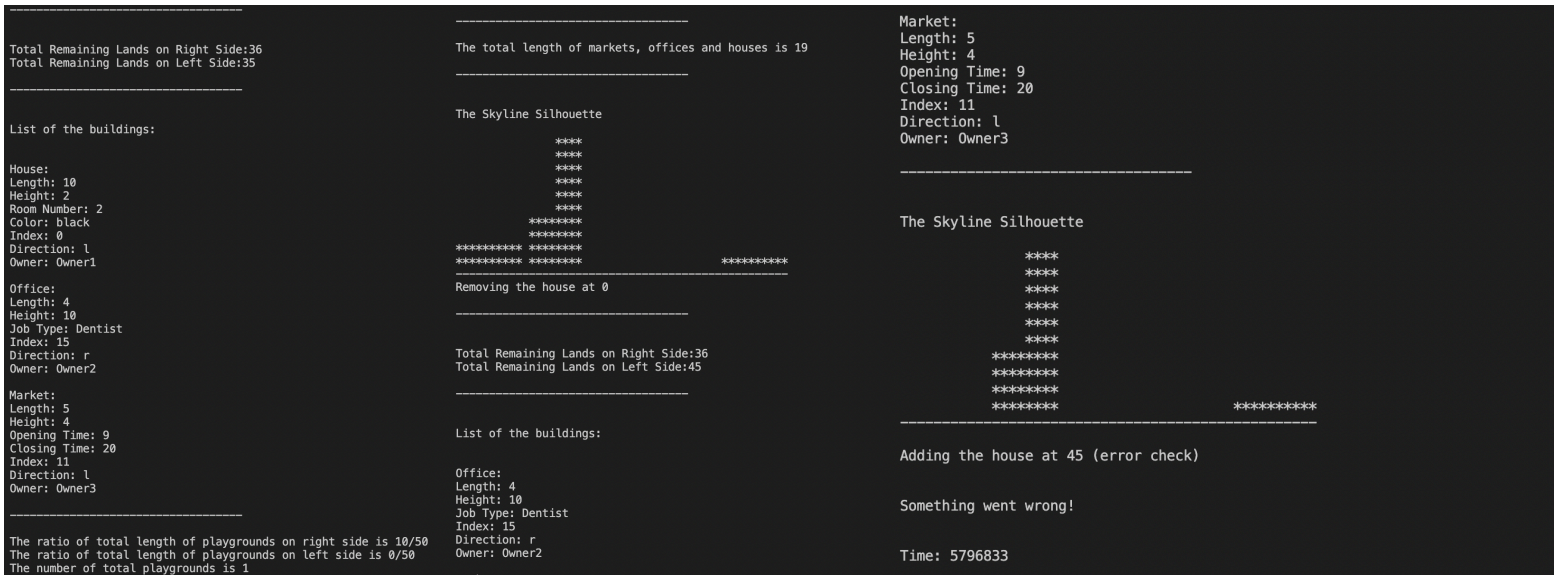
5. Test Cases

- Create a street with a length of 50.
- Create a house, an office, a market and a playground using constructors.
- Add them by overloading the EditMode method.
- View all 5 options.
- Removing the house at index 0
- View related information about buildings after removing.
- For error check, try to add a house at index 45 with a length of 10 which exceeds the length of the street.

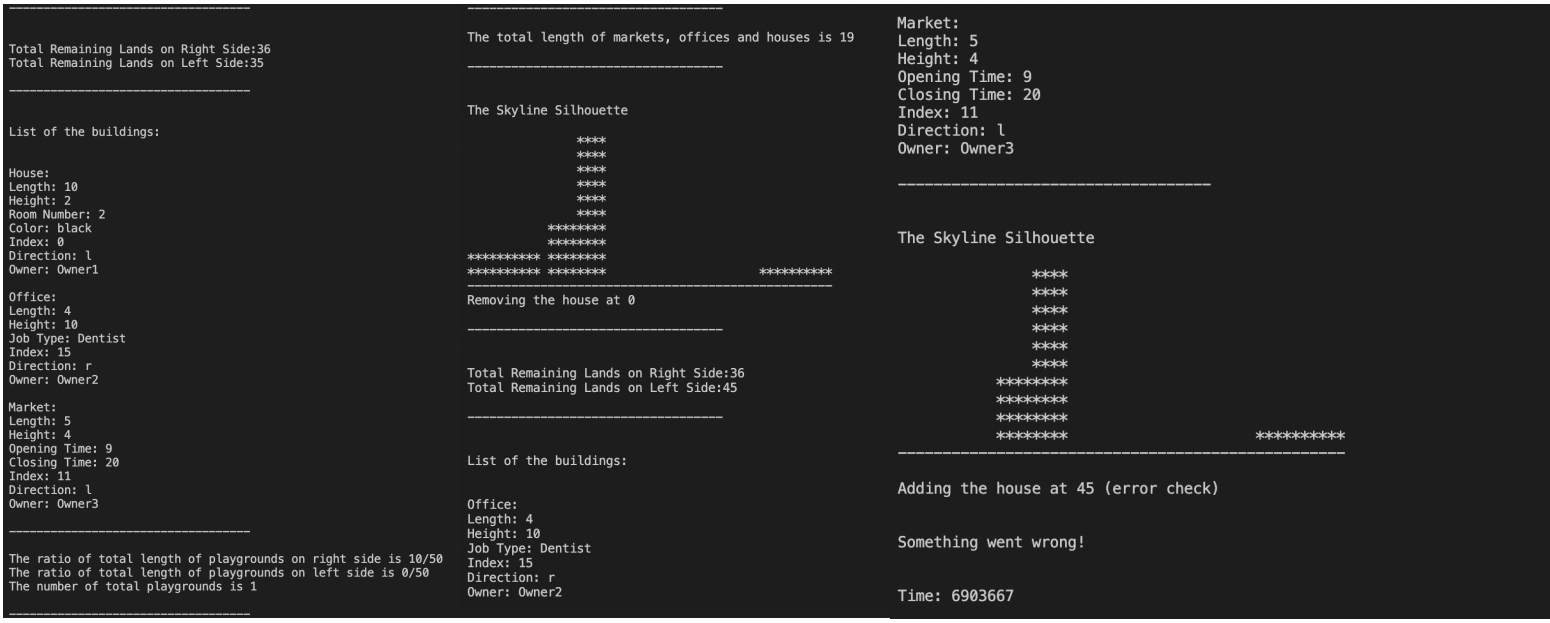
6. Running Command and Results



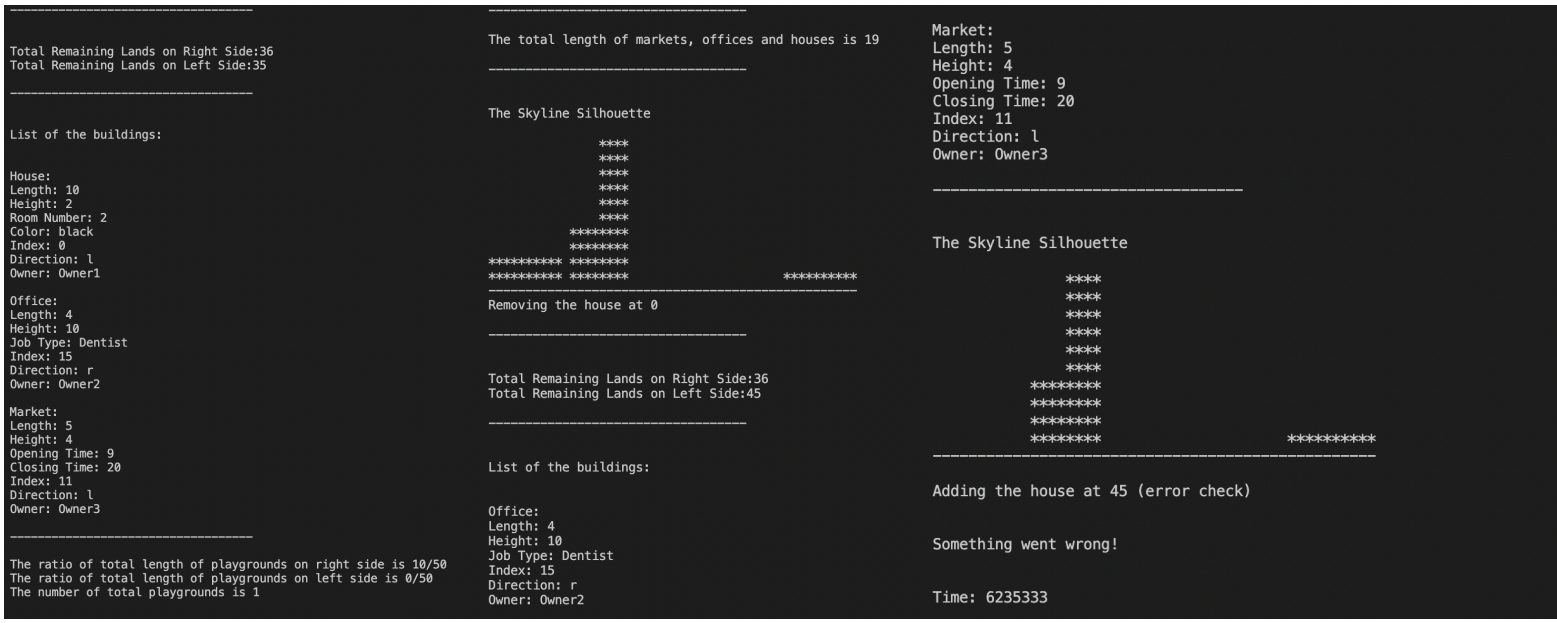
BasicArrayModified



ArrayList



LinkedList



LDLinkedList

Time Complexity

1.BasicArrayModified Running Time: 7191084 ns
2.ArrayList Running Time: 5796833 ns
3.LinkedList Running Time: 6903667 ns
4.LDLinkedList Running Time: 6235333 ns

1

- Create a street with a length of 50. ==> O(n)
- Create a house, an office, a market and a playground using constructors. ==> O(1)
- Add them by overloading the EditMode method. ==> O(n^2)
- View all 5 options. ==> O(n^3)
- Removing the house at index 0 ==> O(n)
- View related information about buildings ==> O(n^3)

2

- Create a street with a length of 50. ==> O(1)
- Create a house, an office, a market and a playground using constructors. ==> O(1)
- Add them by overloading the EditMode method. ==> O(n^2)
- View all 5 options. ==> O(n^3)
- Removing the house at index 0 ==> O(n^2)
- View related information about buildings ==> O(n^3)

3

- Create a street with a length of 50. ==> O(1)
- Create a house, an office, a market and a playground using constructors. ==> O(1)
- Add them by overloading the EditMode method. ==> O(n^2)
- View all 5 options. ==> O(n^3)
- Removing the house at index 0 ==> O(n^2)
- View related information about buildings ==> O(n^3)

4

- Create a street with a length of 50. ==> O(1)
- Create a house, an office, a market and a playground using constructors. ==> O(1)
- Add them by overloading the EditMode method. ==> O(n^2)
- View all 5 options. ==> O(n^3)
- Removing the house at index 0 ==> O(n^2)
- View related information about buildings ==> O(n^3)

We can see that all versions have close time complexity but there are small differences between each other. Array list is the most efficient data structure as seen from the running time. For the first initialization, array list and linked list takes constant time but basic array takes linear time.
This program requires to add new elements at the end of the list so array list is the most suitable one for addition. Removing can be done everywhere so array list and linked list requires n^2 time while basic array requires n time because array list has to shift elements and linked list has to get the previous node. However, in basic array version, I don't remove the elements of array and use them for new elements. Overall, array list stands out for this program.