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

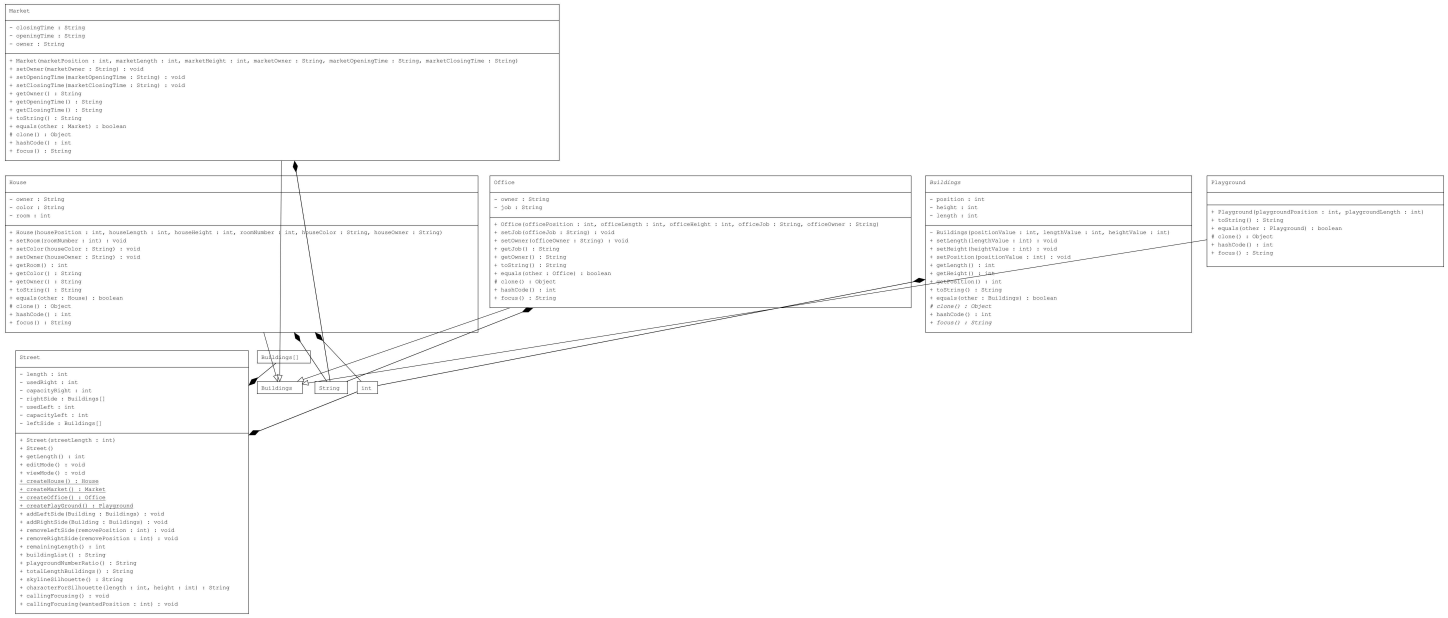
Atacan Başaran
200104004008

1. SYSTEM REQUIREMENTS

This program is a city planning software for designing a small street.We need some designated buildings.Every building has to have something unique to look differen Then we need a street where we will place these created buildings.This street should have a separate structure for the left and right and the capacity to hold the data in these structures.It should be able to add and remove buildings from the street and look at the data of the buildings within it.This information includes the characteristics of the buildings, the appearance of the street, the places on the street, which building occupies how much space, etc.Also a menu is needed where the user can add and remove buildings and see information by guiding him. There are 4 versions of the program that use different data types that provide what I'm talking about above.The purpose of making the program, which will create performance differences between each version, is to observe these differences.

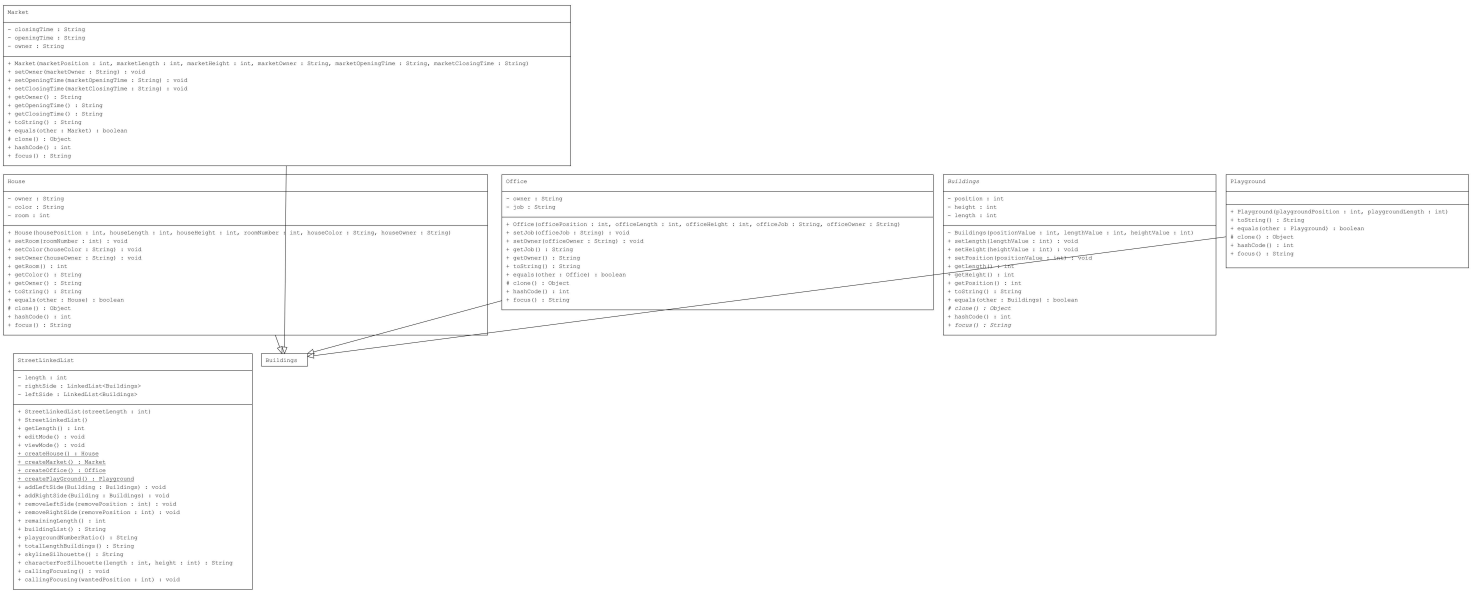
2. CLASS DIAGRAM

Version Array Class Diagram

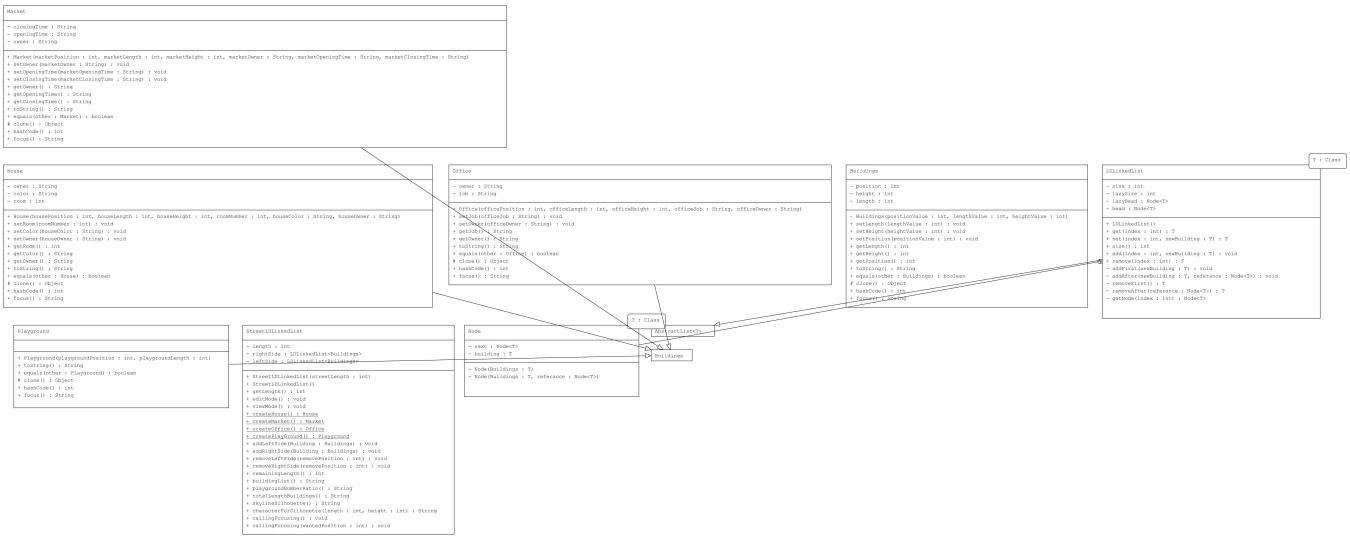


Version ArrayList Class Diagram

Version LinkedList Class Diagram



Version LDLinkedList Class Diagram



3. PROBLEM SOLUTION APPROACH

I kept the street design and driver function in the first assignment. In this assignment, 3 different versions of the first assignment were requested (ArrayList, LinkedList, LDLinkedList). I used it as it is in ArrayList and LinkedList collection, I just needed to call the get() method. The add, remove methods were compatible with my old program already. To see the performance differences of these different data structures, I tested the program with different sized inputs and compared them with the time complexity that should be theoretical. The data matched approximately, but I've had glitches sometimes.

3. TEST CASES

Creating a street

```
Street testStreet = new Street(100);
```

Filling left of the street with 4 different building type(Buildings that cannot be added are shown as error output.)

```
testStreet.addLeftSide(new House(0, 5, 3, 4, "mavi", "Atacan"));
testStreet.addLeftSide(new Market(7, 10, 5, "Mehmet", "08.00", "19.00"));
testStreet.addLeftSide(new Office(20, 4, 20, "Textile", "Nurettin"));
testStreet.addLeftSide(new Playground(40, 30));
testStreet.addLeftSide(new House(15, 5, 5, 3, "pembe", "Ayşe"));
```

Filling right of the street with 4 different building type(Buildings that cannot be added are shown as error output.)

```
testStreet.addRightSide(new House(0, 5, 3, 4, "mor", "Elif"));
testStreet.addRightSide(new Market(7, 10, 5, "Hikmet", "06.00", "21.00"));
testStreet.addRightSide(new Office(20, 4, 20, "Freelancer", "Bedirhan"));
testStreet.addRightSide(new Playground(40, 30));
testStreet.addRightSide(new House(15, 5, 5, 3, "pembe", "Ayşe"));
```

Show data in view mode

```
System.out.printf("\nTotal remainig length of street: %d", testStreet.remainingLength());

System.out.println(testStreet.buildingList());

System.out.println(testStreet.playgroundNumberRatio());

System.out.println(testStreet.totalLengthBuildings());

System.out.println(testStreet.skylineSilhouette());

testStreet.callingFocusing(3);
testStreet.callingFocusing(150);
```

Removing buildings on the left of the street by position(non-removable buildings are shown as error output).

```
testStreet.removeLeftSide(15);
testStreet.removeLeftSide(90);
```

Removing buildings on the right of the street by position(non-removable buildings are shown as error output).

```
testStreet.removeRightSide(50);
testStreet.removeRightSide(80);
```

Show data in view mode after removing buildings

```
System.out.printf("\nTotal remainig length of street: %d", testStreet.remainingLength());

System.out.println(testStreet.buildingList());

System.out.println(testStreet.playgroundNumberRatio());

System.out.println(testStreet.totalLengthBuildings());

System.out.println(testStreet.skylineSilhouette());

testStreet.callingFocusing(20);
testStreet.callingFocusing(-5);
```

Testing some overrided object class method

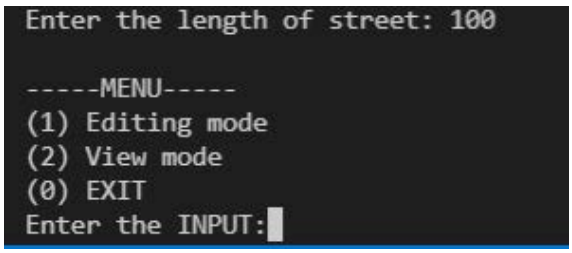
```
House testing1 = new House(0, 5, 5, 3, "gri", "Emre");
House testing2 = new House(0, 5, 5, 3, "gri", "Emre");
House testing3 = new House(5, 10, 10, 3, "kahverengi", "Salih");
System.out.println("\nTesting equals method");
if (testing1.equals(testing2))
    System.out.println("Testing1 and testing2 house are equal each other");
if (!(testing1.equals(testing3)))
    System.out.println("Testing1 and testing3 house are not equal each other");

System.out.println("\nTesting hashcode method");
System.out.print("\nTesting1 hashcode: " + testing1.hashCode());
System.out.print("\nTesting2 hashcode: " + testing2.hashCode());
System.out.print("\nTesting3 hashcode: " + testing3.hashCode());
```

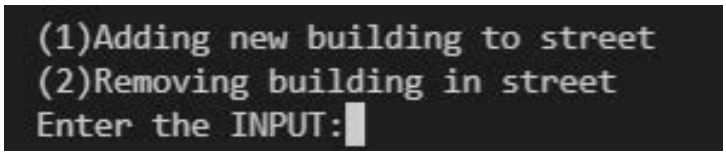
5. RUNNING AND RESULTS

Menu when program run.

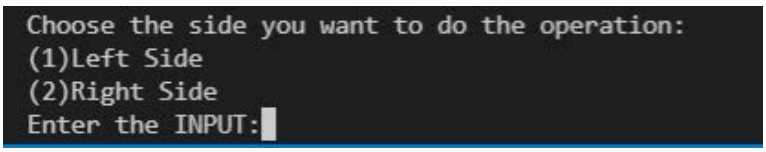
Creating street and menu shown



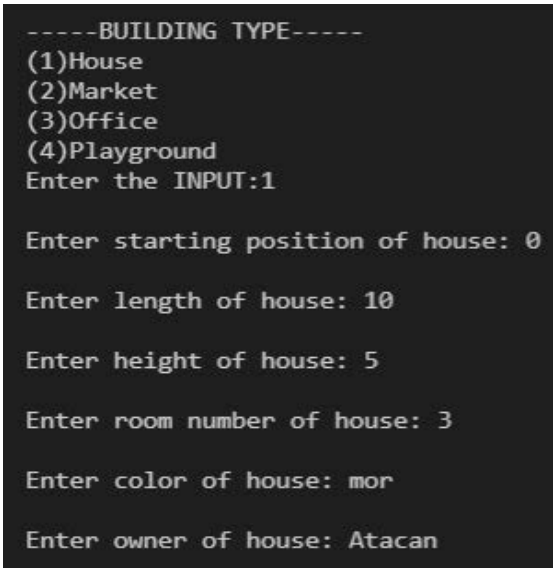
Editing mode



Adding building



After choosing side adding building type



Removing building

```
(1)Adding new building to street
(2)Removing building in street
Enter the INPUT:2

Choose the side you want to do the operation:
(1)Left Side
(2)Right Side
Enter the INPUT:1

Enter the position of the building you want to remove:0
```

View Mode

```
 -----MENU-----
 (1) Editing mode
 (2) View mode
 (0) EXIT
 Enter the INPUT:2

 (1)Total remainig length of street
 (2)List of buildings(number and details)
 (3)Number and Ratio Length of Playgrounds
 (4)Total length of different buildings
 (5)Skyline silhouette
 (6)Focusing a building with starting position
 Enter the INPUT:
```

Total Remaining length of street

```
Enter the INPUT: 1

Total remainig length of street: 175
```

List of buildings

```
There are 1 House,1 Market,0 Office,1 Playground in street
-----Building details-----
Building starting position: 0
Building length: 5
Building height: 5
House room number: 3
House color: mor
House owner: Atacan
------------------------
Building starting position: 40
Building length: 10
Building height: 5
------------------------
Building starting position: 0
Building length: 10
Building height: 5
Market owner: Mehmet
Market working times: 17.00 - 19.00
------------------------
```

Number and Ratio length of playgrounds

```
Enter the INPUT: 3

Occupied length by Playgrounds: 10
Ratio of length of Playgrounds in street: %5.0
```

Total length of different buildings

```
Enter the INPUT: 4

Occupied length by House: 5
Occupied length by Market: 10
Occupied length by Office: 0
```

Focusing a building

```
Enter the INPUT: 6

Enter the position of the building you want to focus on: 10

Building details on right side:
Building starting position: 0
Building length: 10
Building height: 5
Market owner: Mehmet
Market working times: 17.00 - 19.00

Focusing results on right side: 19.00
```

6. TIME COMPLEXITIES

	Size	Array	Time Complexity	ArrayList	Time Complexity	LinkedList	Time Complexity	LDLinkedList	Time Complexity
get() and []	10	2	θ(1)	3	θ(1)	3	O(n)	3	O(n)
	100	2		3		32		27	
	1000	2		3		250		200	
add()	10	25	O(n)	50	O(n)	14	O(n)	33	O(n)
	100	270		570		120		175	
	1000	2000		5000		1000		2000	
remove()	10	6	O(n)	10	O(n)	10	O(n)	17	O(n)
	100	12		70		42		51	
	1000	92		500		570		450	
skylineSilhouette()	10	285	θ(n)	500	θ(n)	750	θ(n)	850	θ(n)
	100	4800		12500		21000		26000	
	1000	75000		115 000		200 000		280 000	