GTU Department of Computer Engineering

CSE 222 / 505 – Spring 2022

Homework 3

Ahmet USLUOGLU

# 1801042602

### – System Requirement

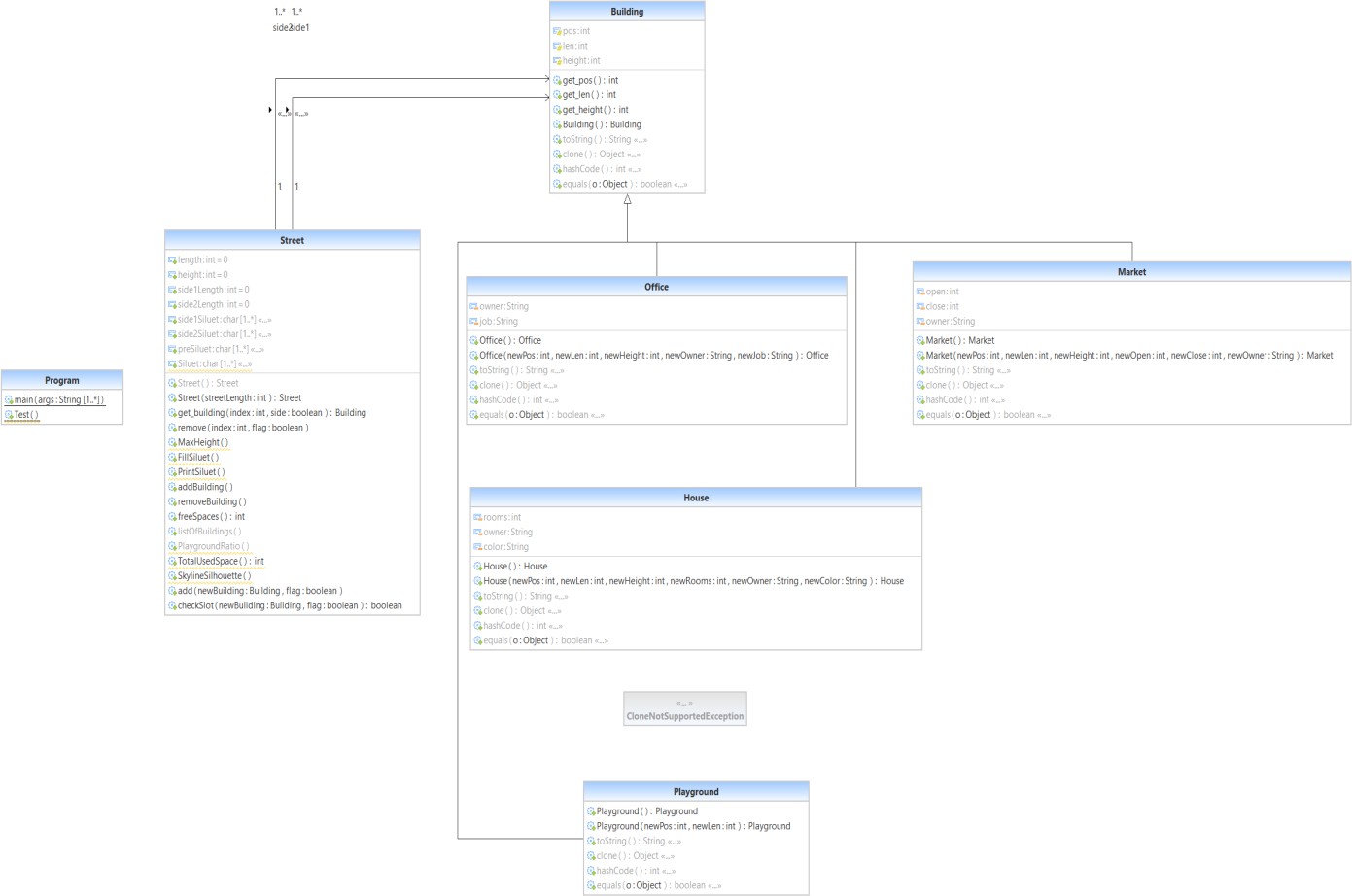
Operating System must have JDK (Java Development Kit) 11 and JRE (Java Runtime Environment) 11 or higher.

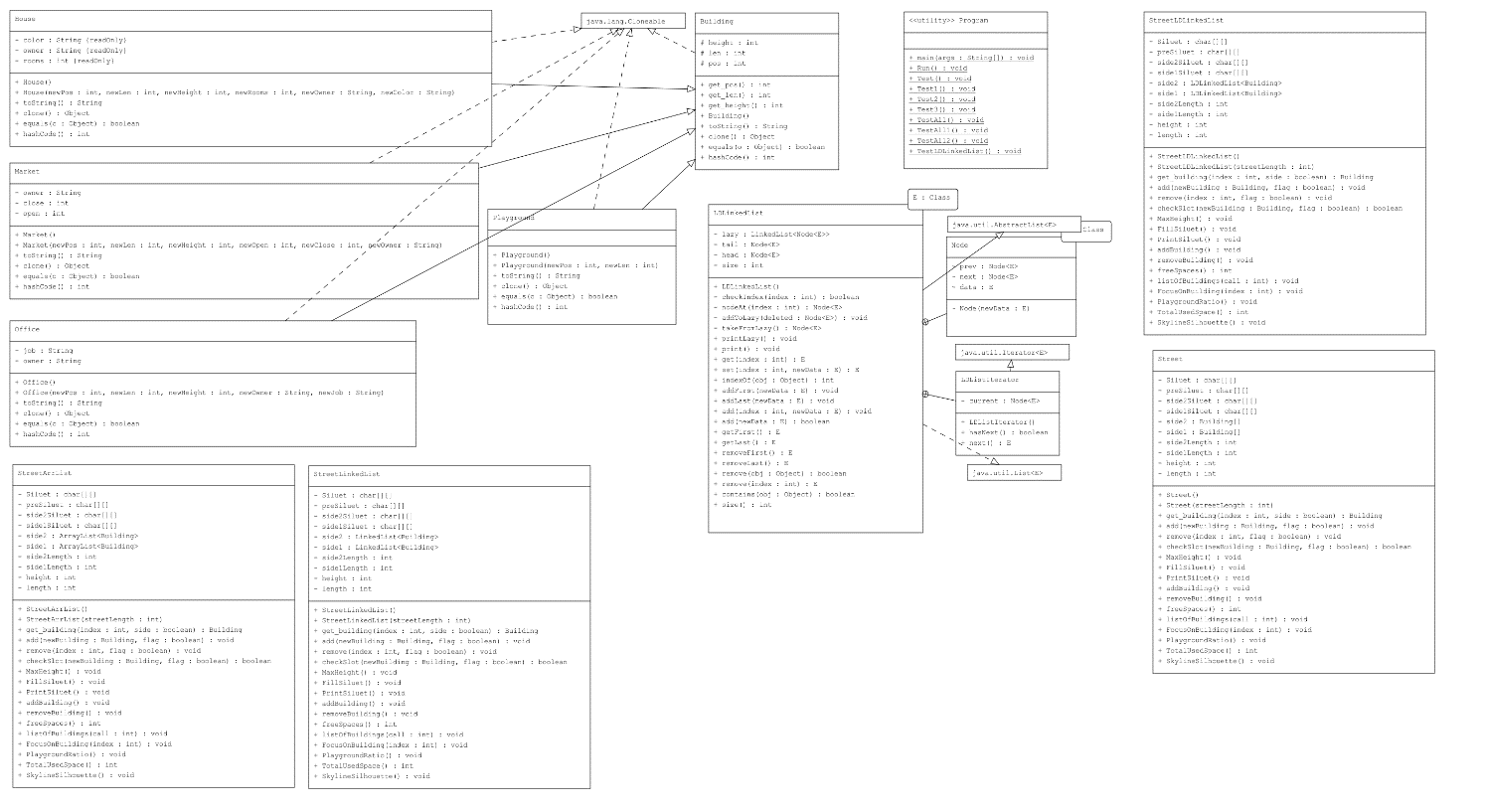
There should be enough space for storing data’s.

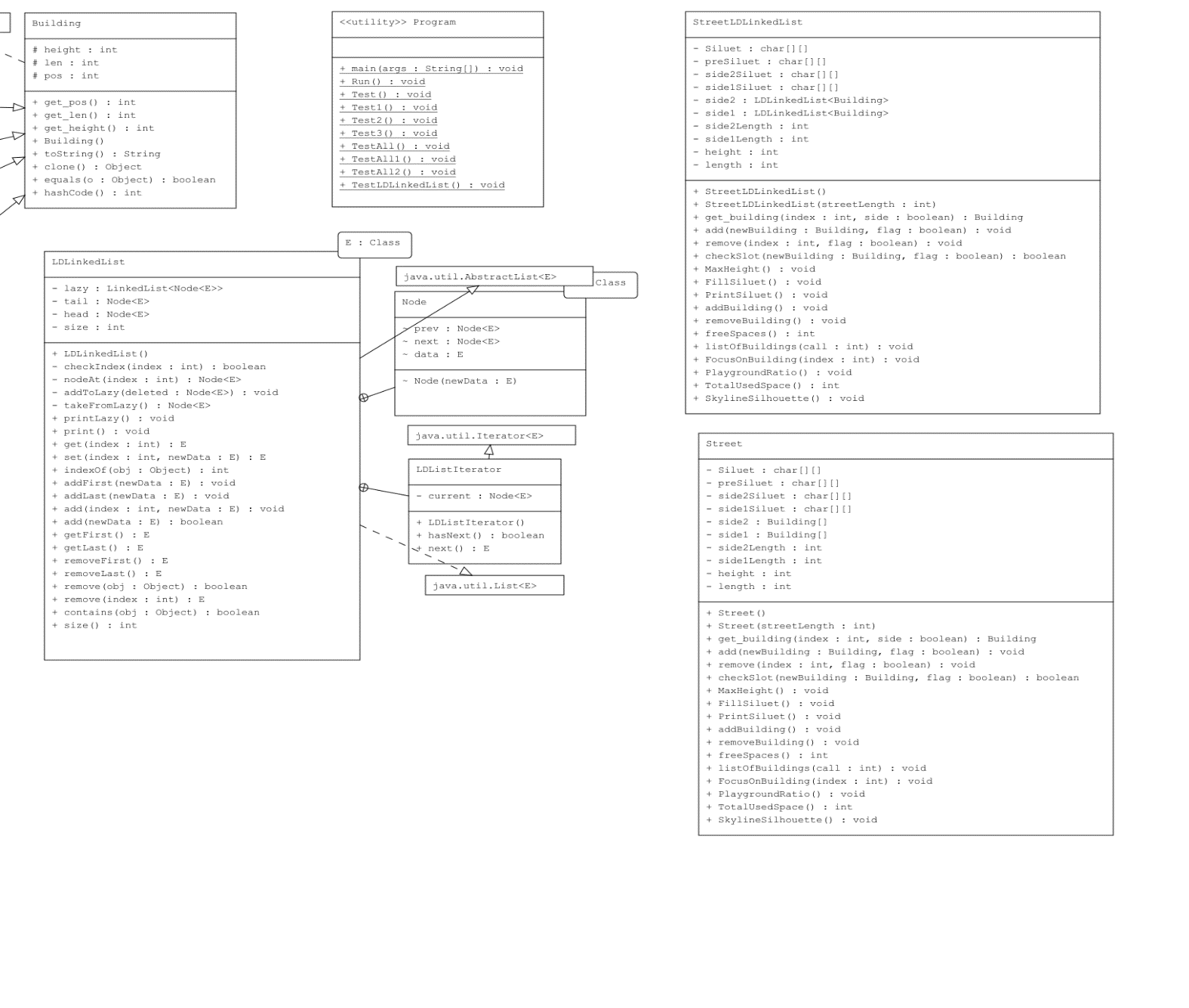
User must create a Street with a fixed length at the start of the program. User must enter correct input values while adding a new building.

1. – Class Diagrams

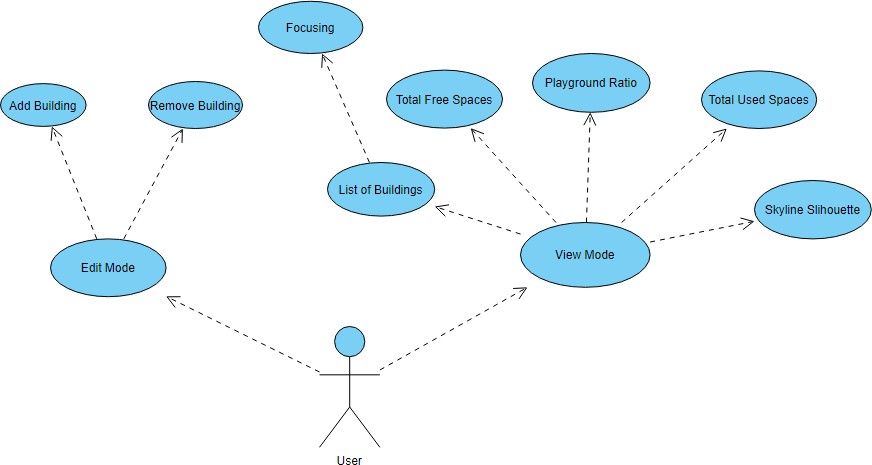
\*Higher Resolution Version of the Class Diagram is in the files.







# – Use Case Diagrams



3. Problem - Solution Approach

Problem:

1 - ) Designing a small Street planning software with the feature of Silhouette of Skyline. Street consists of 2 sides. Each side has free spaces to construct a building. Street’s Length is fixed, and space is limited. Construct buildings that do not conflicts with other buildings at the same side.

2-) Implementing above problem’s solution with 4 types of data structures which are arrays, array lists, linked lists and custom linked lists (LDLinkedList). Compare all data structure’s running time, complexity and theoretical time.

Solution:

1-) The buildings that can be constructed (House, Office, Market, Playground) on the Street inherited from a Building class which has length, position and height properties.

I have then created a Street class to oversee all related functions that will be needed to construct buildings on the street.

I have stored the buildings in 2 separate arrays respective to each side of the Street.

Each side of the Street has 1 Building array as container to store buildings and 1 char array to store silhouette of the street.

After creating 2 different char arrays with silhouettes i have combined them to 1 as preSlihouette array by only adding the parts where both of the arrays are empty.

The Empty points in the preSlihouette array is not finished. I have reversed the array and only printed the walls of the buildings at the outer most side. Thus Skyline Silhouette is completed.

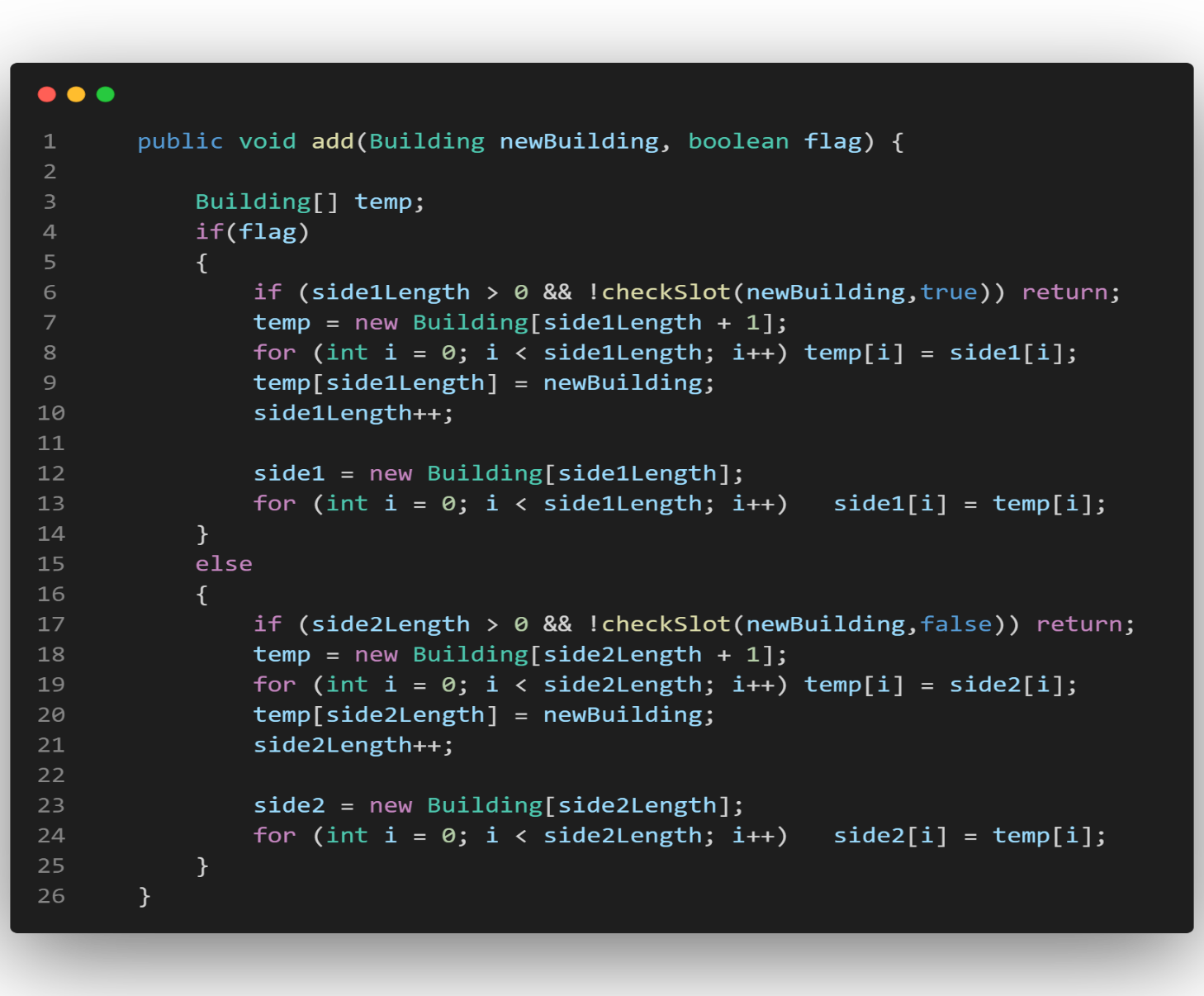
Other functions are simple calculation functions that sums up total lengths, empty spaces, used spaces etc.

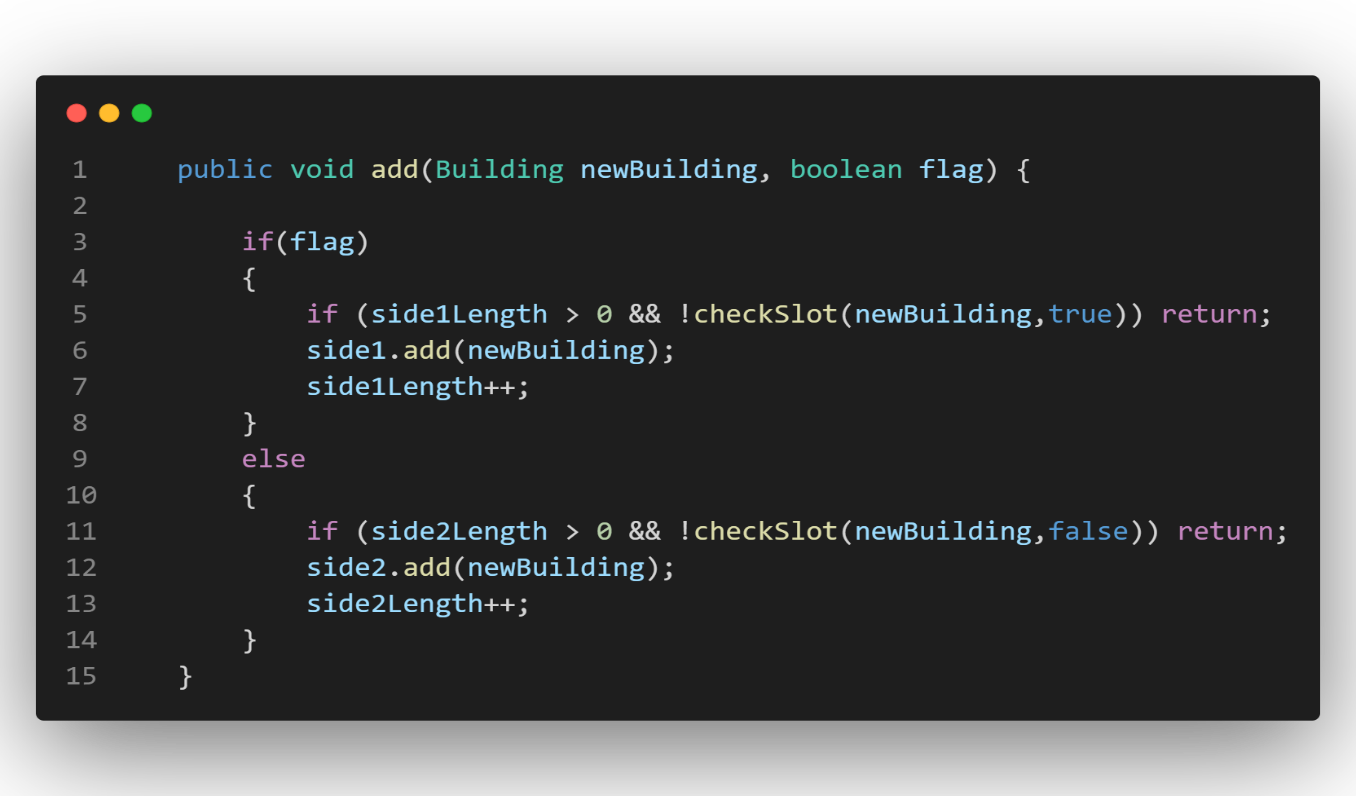
Exceptions are handled in case of invalid values for inputs are given.

2-) Using above solution I have created 3 other street classes that uses array list, linked list and LDlinked list as container types for buildings. After creating total of 4 different street class for each of the data structure I have tested their running times with different size of buildings, calculated their complexity and compared their experimental times.

**Example of Add Building Function with Different Data Structures**

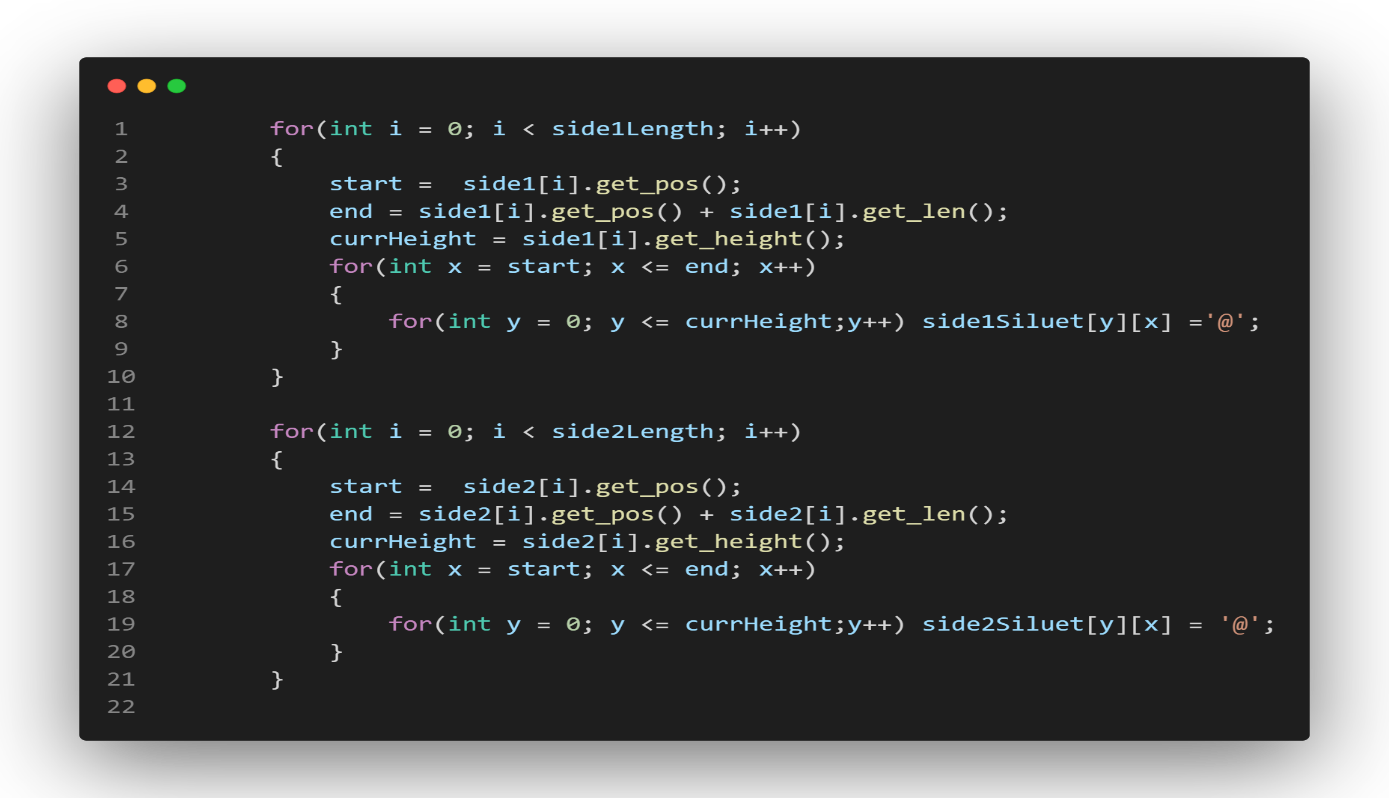
**- Using Arrays**

****

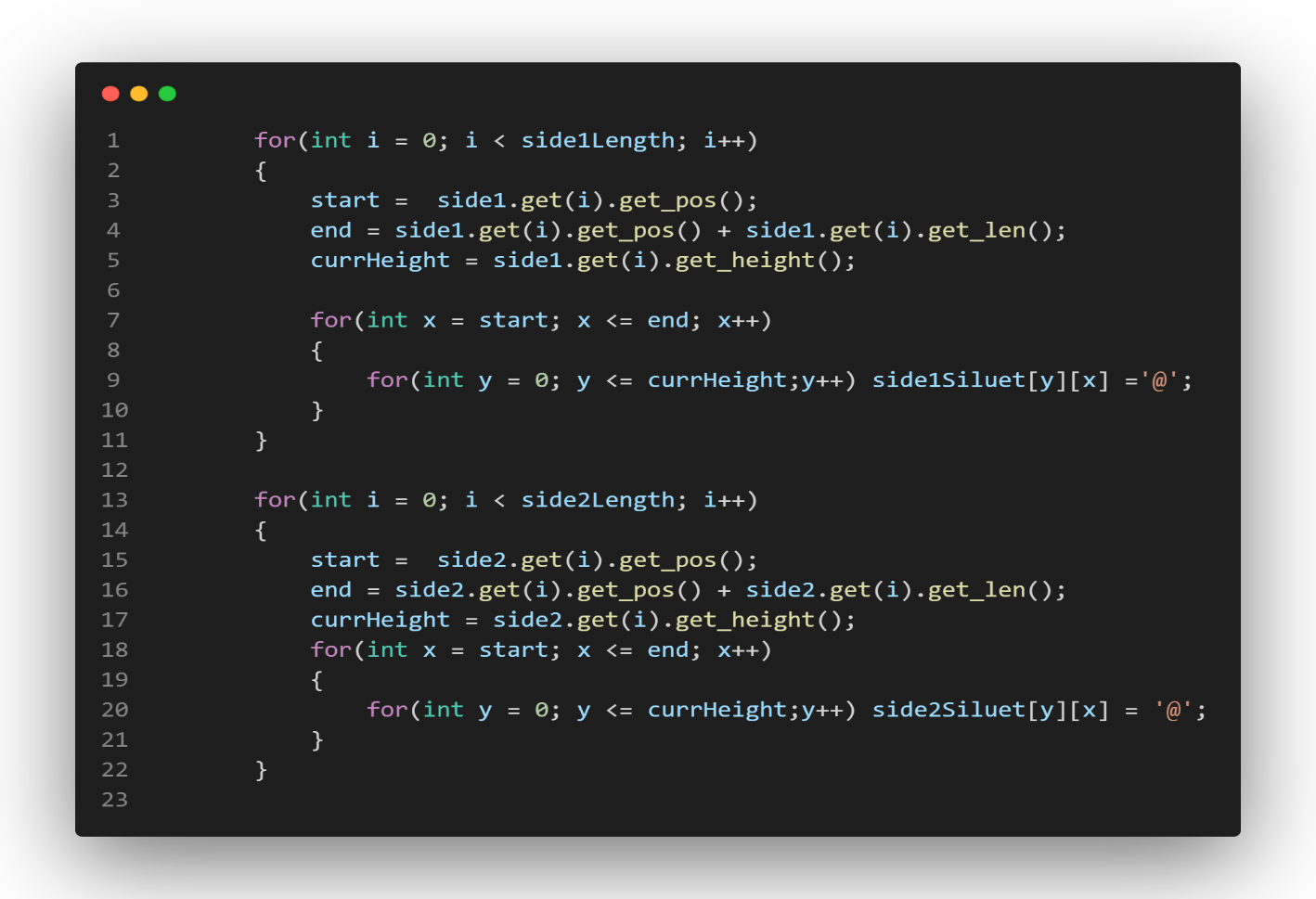
**- Using Array list, Linked list, LDLinked list**

**Example of Print Building Function with Different Data Structures**

**- Using Arrays**

****

**- Using Array list, Linked list, LDLinked list**

****

Side1length and side2length represents the number of buildings on each side of the street.

Silhouette functions time complexities are related to the total number of buildings.

1. – Test Cases

a-) Testing the complexity and running of the street classes

1 – Create Street objects with each of the Data Structures

2 – Add Different Set of number of buildings to each street

3 – Compare Running Times

4 – Calculate Time complexity

5 – Testing the LDLinkedList Class

b-) Testing the functionality of the street classes

* 1. Create a Street
  2. Create and Add Buildings 3- Display Silhouette

1. Display Details
2. Remove Buildings
3. Display Total Free Spaces 7- Display Total Used Spaces

8- Display Playground to Buildings Ratio 9- Print List of the Buildings

## – Running Program and Results

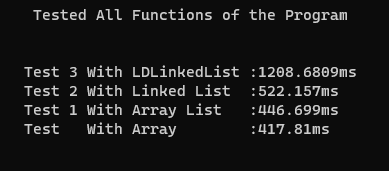
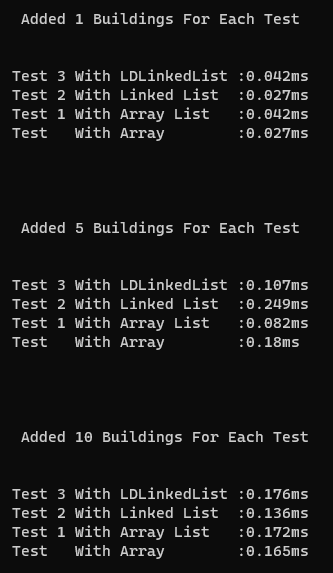
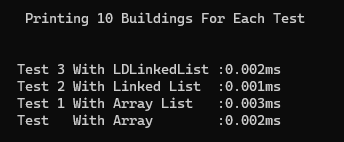
a-) Testing the complexity and running of the street classes

### 1 – Create Street objects with each of the Data Structures

### 

2 – Add Different Set of number of buildings to each street

### 

3 – Compare Running Times

### 4 – Calculate Time complexity

### -In all Classes Time Complexity is related to total number of Buildings (n),

### Length of the street (x) and the maximum height of the buildings(y).

### - The time Complexity for The Street Class that uses Array

### Tetha (n \* x \* y) = Tetha (N)

### - The time Complexity for The Street Class that uses Array List

### Tetha (n \* x \* y) = Tetha(N)

### - The time Complexity for The Street Class that uses Linked List

### Tetha (n\*n \* x \* y) = Tetha(N\*N)

### - The time Complexity for The Street Class that uses LDLinked List

### Tetha (n\*n \* x \* y) = Tetha(N\*N)

### For 1 Building Added elements Instructions

### Array = 0.027ms n = 1 n = 1

### Array List= 0.042ms n = 1 n = 1

### Linked List = 0.027ms n = 1 n\*n = 1

### LD Linked List = 0.042ms n = 1 n \*n = 1

### For 5 Building Added elements Instructions Experimental Result

### Array = 0.18ms n = 5 n = 5 0.135

### Array List= 0.082ms n = 5 n = 5 0.21

### Linked List = 0.249ms n = 5 n\*n = 25 0.135

### LD Linked List = 0.107ms n = 5 n \*n = 25 0.21

### For 10 Building Added elements Instructions Experimental Result

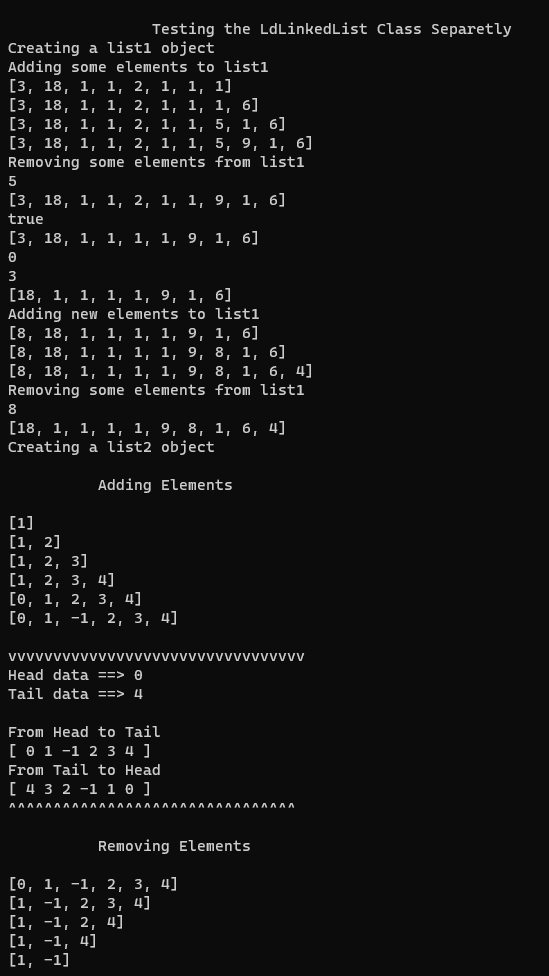
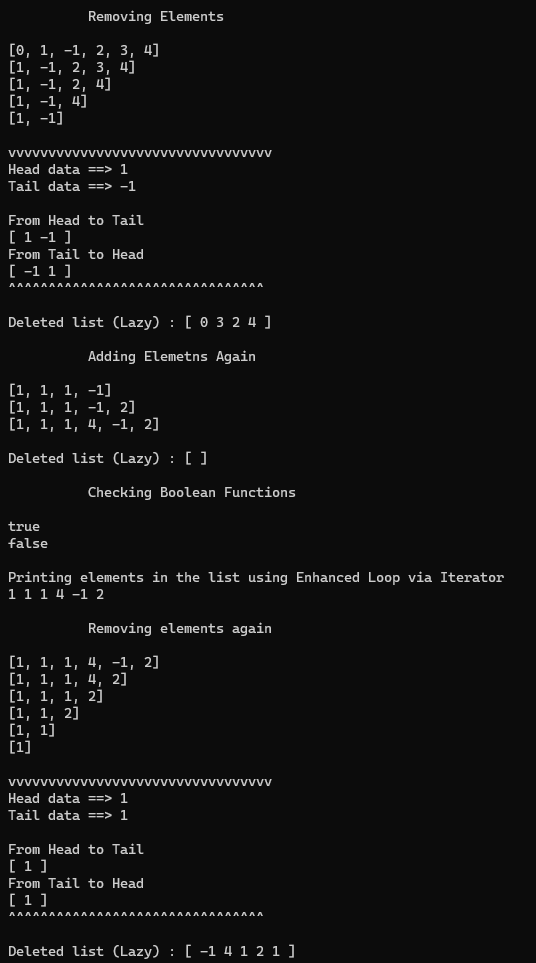
### Array = 0.176ms n = 10 n = 10 0.27

### Array List= 0.136ms n = 10 n = 10 0.42

### Linked List = 0.172ms n = 10 n\*n = 100 0.27

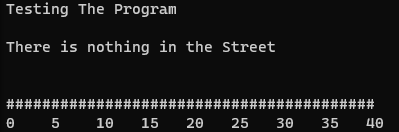
### LD Linked List = 0.165ms n = 10 n \*n = 100 0.42

### 5 – Testing the LDLinkedList Class

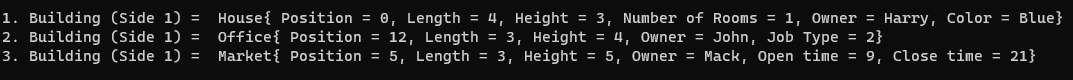


b-) Testing the functionality of the street classes

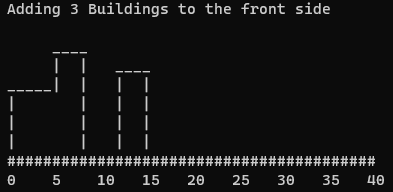
1. – Create a Street



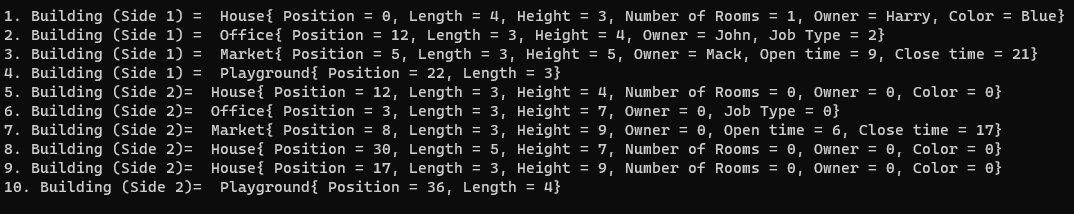
### – Create And Add Buildings



1. – Display Slihouette



## 4- Display Details



1. - Remove Buildings



## – Display Total Free Spaces

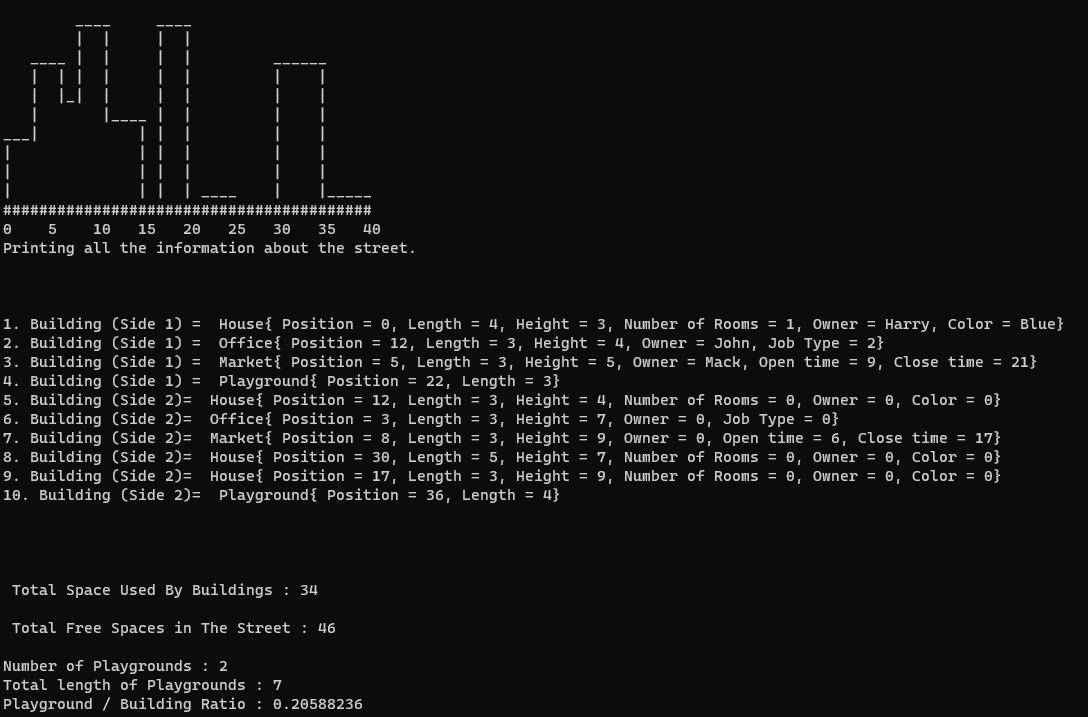


1. – Display Total Used Spaces



## – Display Playground Ratio

1. – Display List of Buildings



### Lastly User Interface Menus

