GTU Department of Computer Engineering
CSE 222/505 - SPRING 2022
HOMEWORK 3 REPORT

BURAK ÇİÇEK
1901042260

1.System Requirements

Test of this Street Simulation in accordance with the law, it will not sell any of your personal information to third-party companies.

In this Street Simulation, I have an Interface Class called by **Building** and there is 3 classes for for **Market, Office and Playground** also of course I have two classes called by **Street and Main.**

Minimum System Requirements:
RAM: 128 MB
Disk space: 124 MB for JRE; 2 MB for Java Update
Processor: Minimum Pentium 2 266 MHz processor
Java DK 10+

Building has

- **getLength()**
- **getHeight()**
- **getPosition()**
- **focus()**

Market class has

- **Opening time**
- **Closing time**
- **length**
- **position**
- **height**
- **Owner**
- **focus()**
- **Setters and Getters for all.**

Playground class has

- **length**
- **position**
- **height**
- **focus()**
- **Setters and Getters for all**

House class has

- **Color**
- **Owner**
- **length**
- **position**
- **height**
- **NumberOfRooms**
- **Setters and Getters for all**

Office class has

- **Owner**
- **JobType**
- **length**
- **position**
- **height**
- **Setters and Getters for all**

Street class has

- **setTotalLength**
- **buildingsNumber**
- **buildingsLastIndex**
- **totalLength**
- **currentLength**
- **add**
- **displayRemainingLength()**
- **delete(int Index)**
- **displayBuildings()**
- **totalLengthOfSpecificBuildings()**
- **displayNumberAndRationOfPlaygrounds()**
- **displayRemainingLength()**
- **displayBuildings()**
- **totalLengthOfSpecificBuildings()**
- **displayNumberAndRationOfPlaygrounds()**
- **displaySkylineSilhouette()**

Also in this 3 seperate parts I have 3 data structures.

```
public LinkedList<Building> streetArray = new LinkedList<Building>();
```

```
public ArrayList<Building> streetArray = new ArrayList<Building>();
```

```
public LDLinkedList<Building> streetArray = new LDLinkedList<Building>();
```

In my LDLinkedList Class:

`LDLinkedList<E> deletedContent;` **For store the Deleted contents.**

`public boolean add(E data)` **Add function. Lookin for deleted Content first.**

`public E get(int index)` **Get the ith index.**

`public E set(int index, E element)` **Set the ith index.**

`public E remove(int index)` **Remove function. Add the item to deleted Content. If there is a previous node tied it with the next of the deleted node.**

`public int size()` **Return size of LDLinkedList.**

Also I have Node Class:

```
public class Node <E>

        E data;
        Node next;
```

It has data and next node.

`Node(E data)` **has a constructor, as usual.**

`public void setData(E data)` **setter for data.**

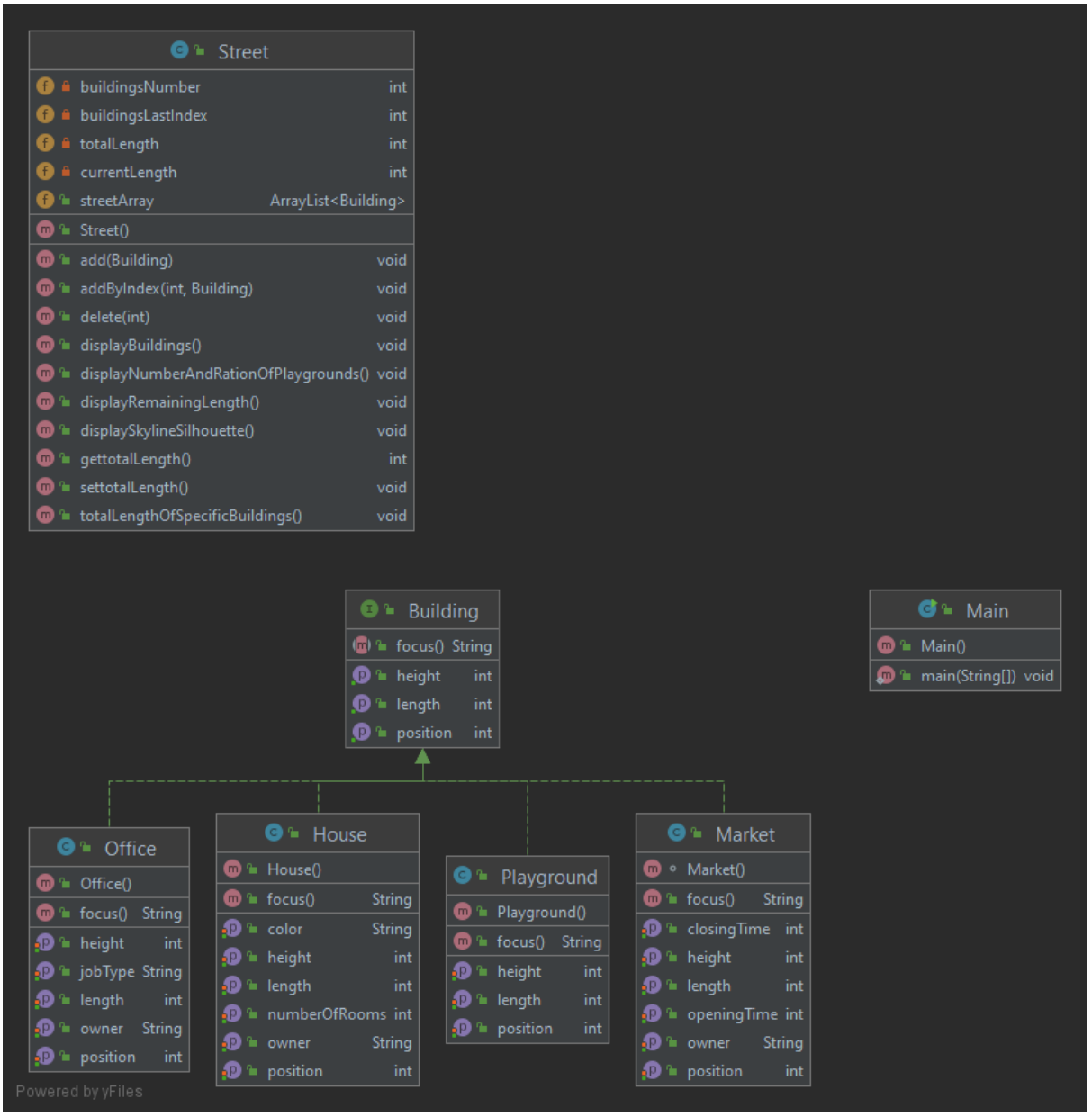
`public void setNext(Node<E> next)` **setter for Next Node.**

`public E getData()` **getter for data.**

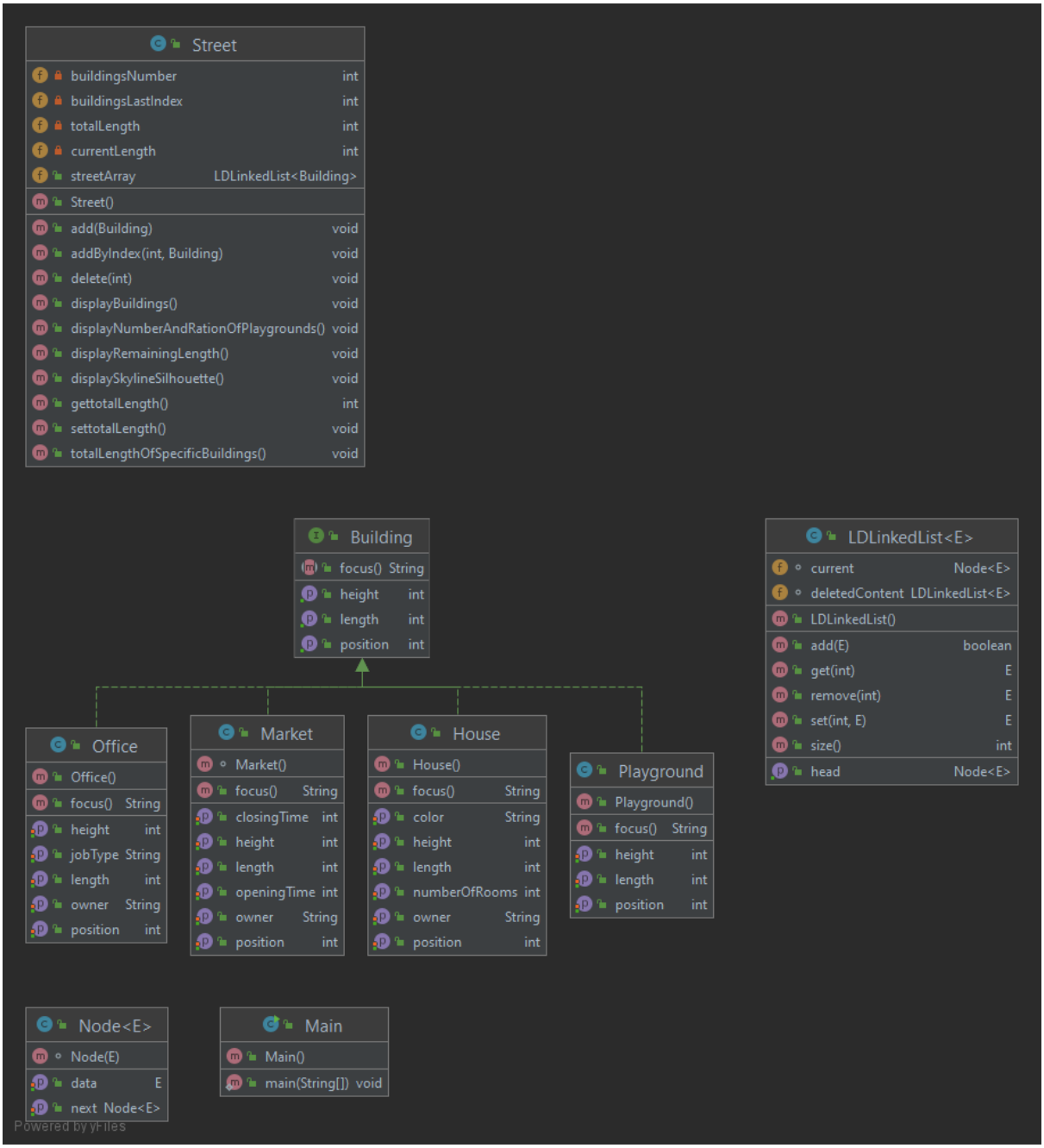
`public Node<E> getNext()` **getter for next Node.**

2.Class Diagrams

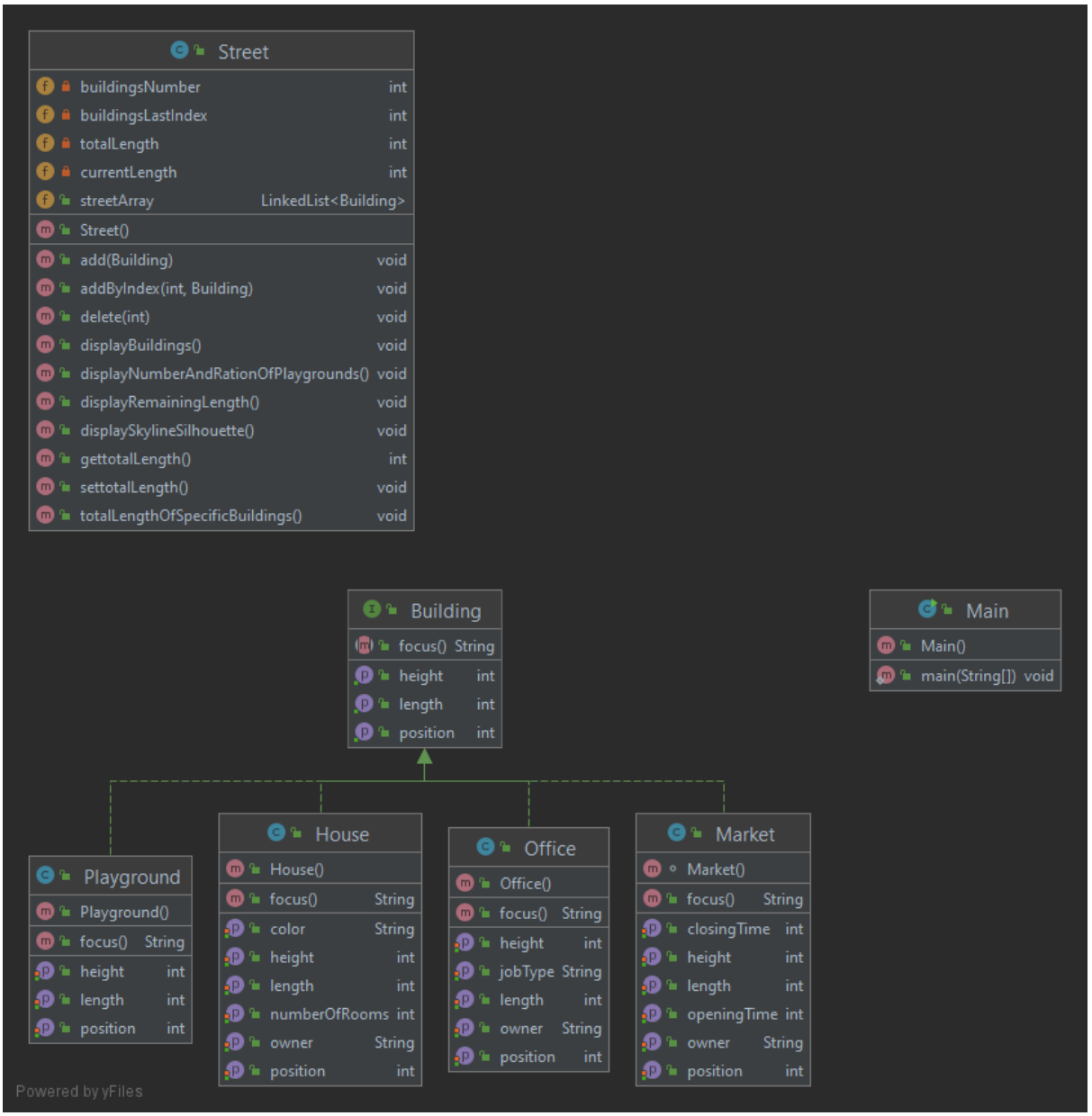
For ArrayList:



For LDLinkedList:


Street
buildingsNumber int
buildingsLastIndex int
totalLength int
currentLength int
streetArray LDLinkedList<Building>
Street()
add(Building) void
addByIndex(int, Building) void
delete(int) void
displayBuildings() void
displayNumberAndRationOfPlaygrounds() void
displayRemainingLength() void
displaySkylineSilhouette() void
gettotalLength() int
settotalLength() void
totalLengthOfSpecificBuildings() void
Building
focus() String
height int
length int
position int
LDLinkedList<E>
current Node<E>
deletedContent LDLinkedList<E>
LDLinkedList()
add(E) boolean
get(int) E
remove(int) E
set(int, E) E
size() int
head Node<E>
Office
Office()
focus() String
height int
jobType String
length int
owner String
position int
Market
Market()
focus() String
closingTime int
height int
length int
openingTime int
owner String
position int
House
House()
focus() String
color String
height int
length int
numberOfRooms int
owner String
position int
Playground
Playground()
focus() String
height int
length int
position int
Node<E>
Node(E)
data E
next Node<E>
Main
Main()
main(String[]) void
Powered by yFiles

For LinkedList:



3.Problem Solution Approach

-I had no problems converting Array to **ArrayList** or **LinkedList**. In this assignment, I encountered all the problems in the **LDLinkedList**. First of all, my first problem was how to keep the deleted nodes in a data structure. I even considered keeping it using Java Collections' LinkedList, and I even did so. But this time, I was keeping only the data inside, not the nodes, which was against the design and homework. Then I thought and put a DeletedContent LD Linked List inside the LDLinkedList to keep all these.

-The second problem I faced was that I didn't know how to navigate in linkedlist. Creating an iterator was an option in this assignment, but I didn't want to use this design. Instead, I created a current node and assign it to the head to enable navigating between nodes.

-Since I am using the current node to move between nodes, I encountered a minor problem with the remove method. When I deleted the node shown by Current, I was also deleting the next node, so I needed an extra auxiliary node to connect the node before and after it. For this, I created the previousCurrent node inside the remove method, which is a node that keeps the information of the previous node of the node to be deleted.

4.Test Cases, Running Command and Results

-My test inputs are the same so I'll add a TAG for LinkedList, ArrayList and LDLinkedList to separate them each other.

Case 1 -ArrayList

```
public static void main(String[] args) {
Street myStreet = new Street();
System.out.println("**********STREET SIMULATION**********");
System.out.println("~Test Case 1 -ArrayList~");
System.out.println("Adding Some building to Street and Display");
House myTestHouse1 = new House();
myTestHouse1.setPosition(1);
myTestHouse1.setHeight(4);
myTestHouse1.setLength(4);
myTestHouse1.setOwner("Blue");
myTestHouse1.setNumberOfRooms(3);
myStreet.add(myTestHouse1);
Playground myTestPlayground1 = new Playground();
myTestPlayground1.setPosition(3);
myTestPlayground1.setLength(6);
myStreet.add(myTestPlayground1);
myStreet.displayBuildings();
```

```
**********STREET SIMULATION**********
~Test Case 1 -ArrayList~
Adding Some building to Street and Display
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
```

Case 2 -ArrayList

```
System.out.println("~Test Case 2 -ArrayList~");
System.out.println("Try to add a build but there is no enough space for it.");
Office myTestOffice1 = new Office();
myTestOffice1.setPosition(9);
myTestOffice1.setLength(15);
myTestOffice1.setHeight(4);
myTestOffice1.setJobType("Software Office");
myTestOffice1.setOwner("Bill Gates");
myStreet.add(myTestOffice1);
```

```
~Test Case 2 -ArrayList~
Try to add a build but there is no enough space for it.
There is not enough space for add a new build.
```

Case 3 -ArrayList

```
System.out.println("~Test Case 3 -ArrayList~");
System.out.println("Increase the Total Length and added Office after that display total remaining length.");
myStreet.settotalLength();
myStreet.add(myTestOffice1);
myStreet.displayRemainingLength();
```

```
~Test Case 3 -ArrayList~
Increase the Total Length and added Office after that display total remaining length.
Please enter length of Street:
100
The remainig length of lands on the street is 75 meter.
```

Case 4 -ArrayList

```
System.out.println("~Test Case 4 -ArrayList~");
System.out.println("Create Market added it after that Display Ratio of Length of playgrounds in the street");
Market myTestMarket1 = new Market();
myTestMarket1.setPosition(12);
myTestMarket1.setLength(9);
myTestMarket1.setHeight(8);
myTestMarket1.setOpeningTime(1030);
myTestMarket1.setClosingTime(2230);
myTestMarket1.setOwner("Elon Musk");
myStreet.add(myTestMarket1);
myStreet.displayNumberAndRationOfPlaygrounds();
```

```
~Test Case 4 -ArrayList~
Create Market added it after that Display Ratio of Length of playgrounds in the street
Total Playgrounds Number: 1 Ratio of Playgrounds: 0.06
```

Case 5 -ArrayList

```
System.out.println("~Test Case 5 -ArrayList~");
System.out.println("Again Display Buildings");
myStreet.displayBuildings();
```

```
~Test Case 5 -ArrayList~
Again Display Buildings
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
```

Case 6 -ArrayList

```
System.out.println("~Test Case 6 -ArrayList~");
System.out.println("Display Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
~Test Case 6 -ArrayList~
Display Skyline Silhouette
            #########
            #########
            #########
   ######   #########
  ########################
  ########################
  ########################
  ########################
```

Case 7 -ArrayList

```
System.out.println("~Test Case 7 -ArrayList~");
System.out.println("Add some playgrounds, office and home. Display the list");
Playground myTestPlayground2 = new Playground();
myTestPlayground2.setPosition(18);
myTestPlayground2.setLength(7);
myStreet.addByIndex( i: 4,myTestPlayground2);
Office myTestOffice2 = new Office();
myTestOffice2.setPosition(30);
myTestOffice2.setLength(10);
myTestOffice2.setHeight(10);
myTestOffice2.setJobType("Change Office");
myTestOffice2.setOwner("Volodimir Zelenski");
myStreet.addByIndex( i: 6,myTestOffice2);
House myTestHouse2 = new House();
myTestHouse2.setPosition(32);
myTestHouse2.setHeight(5);
myTestHouse2.setLength(5);
myTestHouse2.setOwner("Red");
myTestHouse2.setNumberOfRooms(6);
myStreet.addByIndex( i: 5,myTestHouse2);
myStreet.displayBuildings();
```

```
~Test Case 7 -ArrayList~
Add some playgrounds, office and home. Display the list
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
Slot: 4 Type: Playground Length: 7
Slot: 5 Type: House Length: 5
Slot: 6 Type: Office Length: 10
```

Case 8 -ArrayList

```
System.out.println("~Test Case 8 -ArrayList~");
System.out.println("Display Length of Street occupied by the Markets");
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 8 -ArrayList~
Display Length of Street occupied by the Markets
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
Market
Total length of Markets in this street is: 9
```

Case 9 -ArrayList

```
System.out.println("~Test Case 9 -ArrayList~");
System.out.println("Display new Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
~Test Case 9 -ArrayList~
Display new Skyline Silhouette
                              ##########
                              ##########
            #########         ##########
            #########         ##########
            #########         ##########
   ######   ##############    ##########
 ########################    ##########
 ########################    ##########
 ########################    ##########
 ########################    ##########
```

Case 10 -ArrayList

```
System.out.println("~Test Case 10 -ArrayList~");
System.out.println("Delete slot 5th House and calculate length of occupied by the House ");
myStreet.delete( Index: 5);
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 10 -ArrayList~
Delete slot 5th House and calculate length of occupied by the House
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
House
Total length of Houses in this street is: 4
```

Case 11 -ArrayList

```
System.out.println("~Test Case 11 -ArrayList~");
System.out.println("Test focus functions");
System.out.println(myStreet.streetArray.get(0).focus());
System.out.println(myStreet.streetArray.get(1).focus());
System.out.println(myStreet.streetArray.get(2).focus());
System.out.println(myStreet.streetArray.get(3).focus());
System.out.println(myStreet.streetArray.get(4).focus());
System.out.println(myStreet.streetArray.get(5).focus());
}
```

```
~Test Case 11 -ArrayList~
Test focus functions
Blue
6
Software Office
2230
7
Change Office
```

Case 1 -LinkedList

```
public static void main(String[] args) {
Street myStreet = new Street();
System.out.println("**********STREET SIMULATION**********");
System.out.println("~Test Case 1 -LinkedList~");
System.out.println("Adding Some building to Street and Display");
House myTestHouse1 = new House();
myTestHouse1.setPosition(1);
myTestHouse1.setHeight(4);
myTestHouse1.setLength(4);
myTestHouse1.setOwner("Blue");
myTestHouse1.setNumberOfRooms(3);
myStreet.add(myTestHouse1);
Playground myTestPlayground1 = new Playground();
myTestPlayground1.setPosition(3);
myTestPlayground1.setLength(6);
myStreet.add(myTestPlayground1);
myStreet.displayBuildings();
```

```
**********STREET SIMULATION**********
~Test Case 1 -LinkedList~
Adding Some building to Street and Display
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
```

Case 2 -LinkedList

```
System.out.println("~Test Case 2 -LinkedList~");
System.out.println("Try to add a build but there is no enough space for it.");
Office myTestOffice1 = new Office();
myTestOffice1.setPosition(9);
myTestOffice1.setLength(15);
myTestOffice1.setHeight(4);
myTestOffice1.setJobType("Software Office");
myTestOffice1.setOwner("Bill Gates");
myStreet.add(myTestOffice1);
```

```
~Test Case 2 -LinkedList~
Try to add a build but there is no enough space for it.
There is not enough space for add a new build.
```

Case 3 -LinkedList

```
System.out.println("~Test Case 3 -LinkedList~");
System.out.println("Increase the Total Length and added Office after that display total remaining length.");
myStreet.settotalLength();
myStreet.add(myTestOffice1);
myStreet.displayRemainingLength();
```

```
~Test Case 3 -LinkedList~
Increase the Total Length and added Office after that display total remaining length.
Please enter length of Street:
100
The remainig length of lands on the street is 75 meter.
```

Case 4 -LinkedList

```
System.out.println("~Test Case 4 -LinkedList~");
System.out.println("Create Market added it after that Display Ratio of Length of playgrounds in the street");
Market myTestMarket1 = new Market();
myTestMarket1.setPosition(12);
myTestMarket1.setLength(9);
myTestMarket1.setHeight(8);
myTestMarket1.setOpeningTime(1030);
myTestMarket1.setClosingTime(2230);
myTestMarket1.setOwner("Elon Musk");
myStreet.add(myTestMarket1);
myStreet.displayNumberAndRationOfPlaygrounds();
```

```
~Test Case 4 -LinkedList~
Create Market added it after that Display Ratio of Length of playgrounds in the street
Total Playgrounds Number: 1 Ratio of Playgrounds: 0.06
```

Case 5 -LinkedList

```
System.out.println("~Test Case 5 -LinkedList~");
System.out.println("Again Display Buildings");
myStreet.displayBuildings();
```

```
~Test Case 5 -LinkedList~
Again Display Buildings
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
```

Case 6 -LinkedList

```
System.out.println("~Test Case 6 -LinkedList~");
System.out.println("Display Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
~Test Case 6 -LinkedList~
Display Skyline Silhouette
            #########
            #########
            #########
   ######   #########
 ########################
 ########################
 ########################
 ########################
```

Case 7 -LinkedList

```
System.out.println("~Test Case 7 -LinkedList~");
System.out.println("Add some playgrounds, office and home. Display the list");
Playground myTestPlayground2 = new Playground();
myTestPlayground2.setPosition(18);
myTestPlayground2.setLength(7);
myStreet.add(myTestPlayground2);
Office myTestOffice2 = new Office();
myTestOffice2.setPosition(30);
myTestOffice2.setLength(10);
myTestOffice2.setHeight(10);
myTestOffice2.setJobType("Change Office");
myTestOffice2.setOwner("Volodimir Zelenski");
House myTestHouse2 = new House();
myTestHouse2.setPosition(32);
myTestHouse2.setHeight(5);
myTestHouse2.setLength(5);
myTestHouse2.setOwner("Red");
myTestHouse2.setNumberOfRooms(6);
myStreet.add(myTestHouse2);
myStreet.add(myTestOffice2);
myStreet.displayBuildings();
```

```
~Test Case 7 -LinkedList~
Add some playgrounds, office and home. Display the list
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
Slot: 4 Type: Playground Length: 7
Slot: 5 Type: House Length: 5
Slot: 6 Type: Office Length: 10
```

Case 8 -LinkedList

```
System.out.println("~Test Case 8 -LinkedList~");
System.out.println("Display Length of Street occupied by the Markets");
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 8 -LinkedList~
Display Length of Street occupied by the Markets
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
Market
Total length of Markets in this street is: 9
```

Case 9 -LinkedList

```
System.out.println("~Test Case 9 -LinkedList~");
System.out.println("Display new Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
 ~Test Case 9 -LinkedList~
 Display new Skyline Silhouette
                              ##########
                              ##########
            #########         ##########
            #########         ##########
            #########         ##########
    ######  ##########        ##########
  ########################    ##########
  ########################    ##########
  ########################    ##########
  ########################    ##########
```

Case 10 -LinkedList

```
System.out.println("~Test Case 10 -LinkedList~");
System.out.println("Delete slot 5th House and calculate length of occupied by the House ");
myStreet.delete( Index: 5);
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 10 -LinkedList~
Delete slot 5th House and calculate length of occupied by the House
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
House
Total length of Houses in this street is: 4
```

Case 11 -LinkedList

```
System.out.println("~Test Case 11 -LinkedList~");
System.out.println("Test focus functions");
System.out.println(myStreet.streetArray.get(0).focus());
System.out.println(myStreet.streetArray.get(1).focus());
System.out.println(myStreet.streetArray.get(2).focus());
System.out.println(myStreet.streetArray.get(3).focus());
System.out.println(myStreet.streetArray.get(4).focus());
System.out.println(myStreet.streetArray.get(5).focus());
```

```
~Test Case 11 -LinkedList~
Test focus functions
Blue
6
Software Office
2230
7
Change Office
```

Case 1 -LDLinkedList

```
public static void main(String[] args) {
Street myStreet = new Street();
System.out.println("**********STREET SIMULATION**********");
System.out.println("~Test Case 1 -LDLinkedList~");
System.out.println("Adding Some building to Street and Display");
House myTestHouse1 = new House();
myTestHouse1.setPosition(1);
myTestHouse1.setHeight(4);
myTestHouse1.setLength(4);
myTestHouse1.setOwner("Blue");
myTestHouse1.setNumberOfRooms(3);
myStreet.add(myTestHouse1);
Playground myTestPlayground1 = new Playground();
myTestPlayground1.setPosition(3);
myTestPlayground1.setLength(6);
myStreet.add(myTestPlayground1);
myStreet.displayBuildings();
```

```
Please enter length of Street:
10
**********STREET SIMULATION**********
~Test Case 1 -LDLinkedList~
Adding Some building to Street and Display
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
```

Case 2 -LDLinkedList

```
System.out.println("~Test Case 2 -LDLinkedList~");
System.out.println("Try to add a build but there is no enough space for it.");
Office myTestOffice1 = new Office();
myTestOffice1.setPosition(9);
myTestOffice1.setLength(15);
myTestOffice1.setHeight(4);
myTestOffice1.setJobType("Software Office");
myTestOffice1.setOwner("Bill Gates");
myStreet.add(myTestOffice1);
```

```
~Test Case 2 -LDLinkedList~
Try to add a build but there is no enough space for it.
There is not enough space for add a new build.
```

Case 3 -LDLinkedList

```
System.out.println("~Test Case 3 -LDLinkedList~");
System.out.println("Increase the Total Length and added Office after that display total remaining length.");
myStreet.settotalLength();
myStreet.add(myTestOffice1);
myStreet.displayRemainingLength();
```

```
~Test Case 3 -LDLinkedList~
Increase the Total Length and added Office after that display total remaining length.
Please enter length of Street:
100
The remainig length of lands on the street is 75 meter.
```

Case 4 -LDLinkedList

```
System.out.println("~Test Case 4 -LDLinkedList~");
System.out.println("Create Market added it after that Display Ratio of Length of playgrounds in the street");
Market myTestMarket1 = new Market();
myTestMarket1.setPosition(12);
myTestMarket1.setLength(9);
myTestMarket1.setHeight(8);
myTestMarket1.setOpeningTime(1030);
myTestMarket1.setClosingTime(2230);
myTestMarket1.setOwner("Elon Musk");
myStreet.add(myTestMarket1);
myStreet.displayNumberAndRationOfPlaygrounds();
```

```
~Test Case 4 -LDLinkedList~
Create Market added it after that Display Ratio of Length of playgrounds in the street
Total Playgrounds Number: 1 Ratio of Playgrounds: 0.06
```

Case 5 -LDLinkedList

```
System.out.println("~Test Case 5 -LDLinkedList~");
System.out.println("Again Display Buildings");
myStreet.displayBuildings();
```

```
~Test Case 5 -LDLinkedList~
Again Display Buildings
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
```

Case 6 -LDLinkedList

```
System.out.println("~Test Case 6 -LDLinkedList~");
System.out.println("Display Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
~Test Case 6 -LDLinkedList~
Display Skyline Silhouette
            #########
            #########
            #########
   ######   #########
  ########################
  ########################
  ########################
  ########################
```

Case 7 -LDLinkedList

```
System.out.println("~Test Case 7 -LDLinkedList~");
System.out.println("Add some playgrounds, office and home. Display the list");
Playground myTestPlayground2 = new Playground();
myTestPlayground2.setPosition(18);
myTestPlayground2.setLength(7);
myStreet.add(myTestPlayground2);
Office myTestOffice2 = new Office();
myTestOffice2.setPosition(30);
myTestOffice2.setLength(10);
myTestOffice2.setHeight(10);
myTestOffice2.setJobType("Change Office");
myTestOffice2.setOwner("Volodimir Zelenski");
House myTestHouse2 = new House();
myTestHouse2.setPosition(32);
myTestHouse2.setHeight(5);
myTestHouse2.setLength(5);
myTestHouse2.setOwner("Red");
myTestHouse2.setNumberOfRooms(6);
myStreet.add(myTestHouse2);
myStreet.add(myTestOffice2);
myStreet.displayBuildings();
```

```
~Test Case 7 -LDLinkedList~
Add some playgrounds, office and home. Display the list
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
Slot: 4 Type: Playground Length: 7
Slot: 5 Type: House Length: 5
Slot: 6 Type: Office Length: 10
```

Case 8 -LDLinkedList

```
System.out.println("~Test Case 8 -LDLinkedList~");
System.out.println("Display Length of Street occupied by the Markets");
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 8 -LDLinkedList~
Display Length of Street occupied by the Markets
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
Market
Total length of Markets in this street is: 9
```

Case 9 -LDLinkedList

```
System.out.println("~Test Case 9 -LDLinkedList~");
System.out.println("Display new Skyline Silhouette");
myStreet.displaySkylineSilhouette();
```

```
 ~Test Case 9 -LDLinkedList~
 Display new Skyline Silhouette
                              ##########
                              ##########
            #########         ##########
            #########         ##########
            #########         ##########
   ######   ##############    ##########
  ########################    ##########
  ########################    ##########
  ########################    ##########
  ########################    ##########
```

Case 10 -LDLinkedList

```
System.out.println("~Test Case 10 -LDLinkedList~");
System.out.println("Delete slot 5th House and calculate length of occupied by the House ");
myStreet.delete( Index: 5);
myStreet.totalLengthOfSpecificBuildings();
```

```
~Test Case 10 -LDLinkedList~
Delete slot 5th House and calculate length of occupied by the House
Please enter name of building to calculate total length with case sensitive (Office, House, Market):
House
Total length of Houses in this street is: 4
```

Case 11 -LDLinkedList

```
System.out.println("~Test Case 11 -LDLinkedList~");
System.out.println("Test focus functions");
myStreet.displayBuildings();
System.out.println(myStreet.streetArray.get(0).focus());
System.out.println(myStreet.streetArray.get(1).focus());
System.out.println(myStreet.streetArray.get(2).focus());
System.out.println(myStreet.streetArray.get(3).focus());
System.out.println(myStreet.streetArray.get(4).focus());
System.out.println(myStreet.streetArray.get(5).focus());
```

```
~Test Case 11 -LDLinkedList~
Test focus functions
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
Slot: 4 Type: Playground Length: 7
Slot: 5 Type: Office Length: 10
```

I also have to check that is the removed build/object added to **deletedContent LDLinkedList**? So I add I System.out.Println to there for the test this scenario.

```
}
if(deletedContent!= null){
    current=deletedContent.head;
    while(current != null){
        if(data == current.data){ // if you found the element of deletedContent than last points current, current points null.
            last.next=current;
            last.next.next = null;
            System.out.println("The object/build found in the Deleted Content!");
            return true;
        }
```

I will add the house that I deleted at Case 10, So

Case 12 -LDLinkedList

```
System.out.println("~Test Case 12 -LDLinkedList~");
System.out.println("Adding a deleted build, take it from Deleted Content and display the buildings");
myStreet.add(myTestHouse2);
myStreet.displayBuildings();
```

```
~Test Case 12 -LDLinkedList~
Adding a deleted build, take it from Deleted Content and display the buildings
The object/build found in the Deleted Content!
Slot: 0 Type: House Length: 4
Slot: 1 Type: Playground Length: 6
Slot: 2 Type: Office Length: 15
Slot: 3 Type: Market Length: 9
Slot: 4 Type: Playground Length: 7
Slot: 5 Type: Office Length: 10
Slot: 6 Type: House Length: 5
```

Test cases seems well, it’s time to Time Complexities:

5.Time Complexities:

-For Array

add() Method

```
public void add(Building build){//O(1)
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){ //O(1)
        System.out.println("There is not enough space for add a new build."); //O(1)
    }
    else{
    streetArray[buildingsLastIndex] = build; //O(1)
    currentLength += build.getLength(); //O(1)
    buildingsLastIndex++; //O(1)
    buildingsNumber++; // O(1)
    }}
```

O(1)

addByIndex() Method

```
public void addByIndex(int i, Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){ //O(1)
        System.out.println("There is not enough space for add a new build."); //O(1)
    }
    else{
        streetArray[i] = build; //O(1)
        currentLength += build.getLength(); //O(1)
        buildingsLastIndex++; //O(1)
        buildingsNumber++; //O(1)
    }
}
```

O(1)

displayRemainingLength() Method

```
  public void displayRemainingLength(){
      if(currentLength >= totalLength){//O(1)
          System.out.println("There is no space. This street is full of capacity!");//O(1)
      }
      else{
      System.out.println("The remainig length of lands on the street is " + (totalLength - currentLength) + " meter." );//O(1)
  }}
```

O(1)

delete() Method

```
  public void delete(int Index){
      if(streetArray[Index] == null){
          System.out.println("The building that you try to delete is not exist! Please enter a valid Index/Position!");//O(1)
      }
      else{
          currentLength -= streetArray[Index].getLength();//O(1)
          streetArray[Index] = null;//O(1)
          buildingsNumber--;//O(1)
      }
  }
```

O(1)

displayBuildings() Method

```
public void displayBuildings(){
    String classname;
    if(buildingsNumber == 0 ){
        System.out.println("There is no build in this street! "); //O(1)
    }
    else{
        for(int i=0; i<=buildingsLastIndex; i++){ //O(N)
            if(streetArray[i] != null ){ //O(1)
                classname = streetArray[i].getClass().getSimpleName(); //O(1)
                System.out.println("Slot: " + i + " Type: " + classname + " Length: " + streetArray[i].getLength()); //O(1)
            }
        }
    }

}
```

O(N)

totalLengthOfSpecificBuildings() Method

```
public void totalLengthOfSpecificBuildings(){
    int lengthCounter = 0; //O(1)
    String buildType;//O(1)
    if(buildingsNumber == 0){ //O(1)
        System.out.println("There is no building in this street"); //O(1)
    }
    else{
        System.out.println("Please enter name of building to calculate total length with case sensitive (Office, House, Market): "); //O(1)
        Scanner sc = new Scanner(System.in); //O(1)
        String classname; //O(1)
        buildType = sc.nextLine();//O(1)
        switch(buildType){//O(1)
            case "Office": //O(1)
                for(int i=0; i<=buildingsLastIndex; i++){ //O(N)
                    if(streetArray[i] != null){ //O(1)
                    classname = streetArray[i].getClass().getSimpleName();//O(1)
                        if(classname.equals("Office")){//O(1)
                            lengthCounter += streetArray[i].getLength();//O(1)
                        }
                    }

                }
        System.out.println("Total length of Offices in this street is: " + lengthCounter);//O(1)
            break; //O(1)

            case "House": //O(1)

                for(int i=0; i<=buildingsLastIndex; i++){ //O(N)
                    if(streetArray[i] != null){ //O(1)
                        classname = streetArray[i].getClass().getSimpleName();//O(1)
                        if(classname.equals("House")){//O(1)
                            lengthCounter += streetArray[i].getLength();//O(1)
                        }
                    }
                    else
                        continue;//O(1)

                }
                System.out.println("Total length of Houses in this street is: " + lengthCounter);//O(1)
                break;
```

```
        case "Market":

            for(int i=0; i<=buildingsLastIndex; i++){ //O(N)
                if(streetArray[i] != null){//O(1)
                    classname = streetArray[i].getClass().getSimpleName();//O(1)
                    if(classname.equals("Market")){//O(1)
                        lengthCounter += streetArray[i].getLength();//O(1)
                    }
                }
                else
                    continue;//O(1)

            }
            System.out.println("Total length of Markets in this street is: " + lengthCounter);//O(1)
            break;
    }

  }
}
```

O(N)

displayNumberAndRatioPlaygrounds() Method

```
public void displayNumberAndRationOfPlaygrounds(){
    float lengthCounter = 0; int numberOfPlayground =0; //O(1)
    String classname; float ratio; //O(1)
    for(int i=0; i<=buildingsLastIndex; i++){ //O(N)
        if(streetArray[i] != null){ //O(1)
            classname = streetArray[i].getClass().getSimpleName(); //O(1)
            if(classname.equals("Playground")){ //O(1)
                lengthCounter += streetArray[i].getLength(); //O(1)
                numberOfPlayground++; //O(1)
            }
        }
    }
    ratio = lengthCounter/this.totalLength; //O(1)
    System.out.println("Total Playgrounds Number: " + numberOfPlayground + " Ratio of Playgrounds: " + ratio); //O(1)
}
```

O(N)

displaySkylineSilhouette()

```java
public void displaySkylineSilhouette(){ //O(1)
    int maxHeight = 0; //O(1)
    int maxWidth = 0; //O(1)

    for(int i = 0;i<buildingsLastIndex;i++){//O(N)
        maxHeight = maxHeight > streetArray[i].getHeight() ? maxHeight : streetArray[i].getHeight();//O(1)
        maxWidth = maxWidth > streetArray[i].getLength() + streetArray[i].getPosition() ? maxWidth : streetArray[i].getLength() + streetArray[i].getPosition(); //O(1)
    }

    int [] totalHeights = new int[maxWidth]; //O(1)

    for(int i = 0 ;i<buildingsLastIndex;i++){ //O(N)
        for(int j = streetArray[i].getPosition();j<streetArray[i].getPosition()+streetArray[i].getLength();j++){//O(N)
            //System.out.println(totalHeights[j]);
            totalHeights[j] = totalHeights[j] > streetArray[i].getHeight() ? totalHeights[j] : streetArray[i].getHeight(); //O(1)
        }
    }
    //System.out.println(maxHeight);
    for(int i = 0 ; i< maxWidth;i++){ //O(N)
        System.out.print(totalHeights[i] + " "); //O(1)
    }
    System.out.println();

    for(int i = 0;i<maxHeight;i++){ //O(N)
        for(int j = 0;j<maxWidth;j++){ //O(N)
            if(maxHeight - i > totalHeights[j]){ //O(1)
                System.out.print(' '); //O(1)
            }
            else
                System.out.print('#'); //O(1)
        }
        System.out.println();//O(1)
    }
```

O(N^2)

-For ArrayList

add() Method

```java
public void add(Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
    streetArray.add(build);//O(1)
    //streetArray[buildingsLastIndex] = build;
    currentLength += build.getLength();//O(1)
    //buildingsLastIndex++;
    buildingsNumber++;//O(1)
    }}
```

O(1)

addByIndex() Method

```java
public void addByIndex(int i, Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
        if(i< streetArray.size()){ // //O(N)
            streetArray.add(i,build); //O(N)
            //streetArray[i] = build;
            currentLength += build.getLength();// //O(1)
            //buildingsLastIndex++;
            buildingsNumber++; //O(1)
        }
        else{
            streetArray.add(build);//O(1)
            //streetArray[i] = build;
            currentLength += build.getLength();//O(1)
            //buildingsLastIndex++;
            buildingsNumber++;//O(1)
        }

    }
}
```

O(N)

displayRemainingLength() Method

```
public void displayRemainingLength(){
    if(currentLength >= totalLength){//O(1)
        System.out.println("There is no space. This street is full of capacity!");//O(1)
    }
    else{
    System.out.println("The remainig length of lands on the street is " + (totalLength - currentLength) + " meter." );//O(1)
}}
```

O(1)

delete() Method

```
public void delete(int Index){
    if(streetArray.get(Index) == null){//O(1)
        System.out.println("The building that you try to delete is not exist! Please enter a valid Index/Position!");//O(1)
    }
    else{
        currentLength -= streetArray.get(Index).getLength();//O(1)
        streetArray.remove(Index);//O(N)
        //streetArray[Index] = null;
        buildingsNumber--;//O(1)
    }
}
```

O(N)

displayBuildings() Method

```
public void displayBuildings(){
    String classname;
    if(buildingsNumber == 0 ){//O(1)
        System.out.println("There is no build in this street! ");//O(1)
    }
    else{
        for(int i=0; i<streetArray.size(); i++){//O(N)
            if(streetArray.get(i) != null ){//O(1)
                classname = streetArray.get(i).getClass().getSimpleName();//O(1)
                System.out.println("Slot: " + i + " Type: " + classname + " Length: " + streetArray.get(i).getLength());//O(1)
            }
        }
    }

}
```

O(N)

totalLengthOfSpecificBuildings() Method

```java
public void totalLengthOfSpecificBuildings(){
    int lengthCounter = 0;//O(1)
    String buildType;//O(1)
    if(buildingsNumber == 0){//O(1)
        System.out.println("There is no building in this street");//O(1)
    }
    else{
        System.out.println("Please enter name of building to calculate total length with case sensitive (Office, House, Market): ");//O(1)
        Scanner sc = new Scanner(System.in);//O(1)
        String classname;//O(1)
        buildType = sc.nextLine();//O(1)
        switch(buildType){//O(1)
            case "Office"://O(1)
                for(int i=0; i<streetArray.size(); i++){//O(N)
                    if(streetArray.get(i) != null){//O(1)
                    classname = streetArray.get(i).getClass().getSimpleName();//O(1)
                        if(classname.equals("Office")){//O(1)
                            lengthCounter += streetArray.get(i).getLength();//O(1)
                        }
                    }

                }
        System.out.println("Total length of Offices in this street is: " + lengthCounter);//O(1)
            break;//O(1)
```

```java
  case "House"://O(1)

      for(int i=0; i<streetArray.size(); i++){//O(N)
          if(streetArray.get(i) != null){//O(1)
              classname = streetArray.get(i).getClass().getSimpleName();//O(1)
              if(classname.equals("House")){//O(1)
                  lengthCounter += streetArray.get(i).getLength();//O(1)
              }
          }
          else
              continue;//O(1)

      }
      System.out.println("Total length of Houses in this street is: " + lengthCounter);//O(1)
      break;
```

```java
case "Market":

    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(1)
            classname = streetArray.get(i).getClass().getSimpleName();//O(1)
            if(classname.equals("Market")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(1)
            }
        }
        else
            continue;

    }
    System.out.println("Total length of Markets in this street is: " + lengthCounter);//O(1)
    break;
```

O(N) (for every case)

displayNumberAndRationOfPlaygrounds() Method

```java
public void displayNumberAndRationOfPlaygrounds(){
    float lengthCounter = 0; int numberOfPlayground =0;//O(1)
    String classname; float ratio;//O(1)
    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(1)
            classname = streetArray.get(i).getClass().getSimpleName();//O(1)
            if(classname.equals("Playground")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(1)
                numberOfPlayground++;//O(1)
            }
        }
    }
    ratio = lengthCounter/this.totalLength;//O(1)
    System.out.println("Total Playgrounds Number: " + numberOfPlayground + " Ratio of Playgrounds: " + ratio);//O(1)
}
```

O(N)

displaySkylineSilhouette() Method

```java
public void displaySkylineSilhouette(){
    int maxHeight = 0;
    int maxWidth = 0;

    for(int i = 0;i<streetArray.size();i++){//O(N)
        maxHeight = maxHeight > streetArray.get(i).getHeight() ? maxHeight : streetArray.get(i).getHeight();//O(1)
        maxWidth = maxWidth > streetArray.get(i).getLength() + streetArray.get(i).getPosition() ? maxWidth : streetArray.get(i).getLength() + streetArray.get(i).getPosition();//O(1)
    }

    //int [] totalHeights = new int[maxWidth];
    ArrayList <Integer> totalHeights = new ArrayList<~>();//O(1)
    for(int i=0;i<maxWidth;i++){//O(N)
        totalHeights.add(i, element: 0);//O(1)
    }

    for(int i = 0 ;i<streetArray.size();i++){//O(N)
        for(int j = streetArray.get(i).getPosition();j<streetArray.get(i).getPosition()+streetArray.get(i).getLength();j++){//O(N)
                totalHeights.set(j, totalHeights.get(j) > streetArray.get(i).getHeight() ? totalHeights.get(j) : streetArray.get(i).getHeight());//O(1)
        }
    }
    /*System.out.println(totalHeights.size());
    for(int i = 0 ; i< totalHeights.size();i++){
        System.out.println(i);
        System.out.print(totalHeights.get(i) + " ");
    }
    System.out.println();*/

    for(int i = 0;i<maxHeight;i++){//O(N)
        for(int j = 0;j<maxWidth;j++){//O(N)
            if(maxHeight - i > totalHeights.get(j)){//O(1)
                System.out.print(' ');//O(1)
            }
            else
                System.out.print('#');//O(1)
        }
        System.out.println();//O(1)
    }
```

O(N^2)

-For LinkedList

add() Method

```
public void add(Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
    streetArray.addLast(build);//O(1)
    //streetArray[buildingsLastIndex] = build;
    currentLength += build.getLength();//O(1)
    //buildingsLastIndex++;
    buildingsNumber++;//O(1)
    }}
```

O(1)

addByIndex() Method

```
public void addByIndex(int i, Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
        if(i< streetArray.size()){//O(1)
            streetArray.add(i,build);//O(N)
            //streetArray[i] = build;
            currentLength += build.getLength();//O(1)
            //buildingsLastIndex++;
            buildingsNumber++;//O(1)
        }
        else{
            streetArray.push(build);//O(1)
            //streetArray[i] = build;
            currentLength += build.getLength();//O(1)
            //buildingsLastIndex++;
            buildingsNumber++;//O(1)
        }

    }
}
```

O(N)

displayRemainingLength() Method

```
public void displayRemainingLength(){
    if(currentLength >= totalLength){//O(1)
        System.out.println("There is no space. This street is full of capacity!");//O(1)
    }
    else{
    System.out.println("The remainig length of lands on the street is " + (totalLength - currentLength) + " meter." );//O(1)
}}
```

O(1)

delete() Method

```
public void delete(int Index){
    if(streetArray.get(Index) == null){//O(1)
        System.out.println("The building that you try to delete is not exist! Please enter a valid Index/Position!");//O(1)
    }
    else{
        currentLength -= streetArray.get(Index).getLength();//O(N)
        streetArray.remove(Index);//O(N)
        //streetArray[Index] = null;
        buildingsNumber--;//O(1)
    }
}
```

O(N)

displayBuildings() Method

```
public void displayBuildings(){
    String classname;//O(1)
    if(buildingsNumber == 0 ){//O(1)
        System.out.println("There is no build in this street! ");//O(1)
    }
    else{
        for(int i=0; i<streetArray.size(); i++){//O(N)
            if(streetArray.get(i) != null ){//O(N)
                classname = streetArray.get(i).getClass().getSimpleName();//O(1)
                System.out.println("Slot: " + i + " Type: " + classname + " Length: " + streetArray.get(i).getLength());//O(1)
            }
        }
    }

}
```

O(N^2)

totalLengthOfSpecificBuildings() Method

```
public void totalLengthOfSpecificBuildings(){
    int lengthCounter = 0;//O(1)
    String buildType;//O(1)
    if(buildingsNumber == 0){//O(1)
        System.out.println("There is no building in this street");//O(1)
    }
    else{
        System.out.println("Please enter name of building to calculate total length with case sensitive (Office, House, Market): ");//O(1)
        Scanner sc = new Scanner(System.in);//O(1)
        String classname;//O(1)
        buildType = sc.nextLine();//O(1)
        switch(buildType){//O(1)
            case "Office"://O(1)
                for(int i=0; i<streetArray.size(); i++){//O(N)
                    if(streetArray.get(i) != null){//O(N)
                    classname = streetArray.get(i).getClass().getSimpleName();//O(1)
                        if(classname.equals("Office")){//O(1)
                            lengthCounter += streetArray.get(i).getLength();//O(1)
                        }
                    }

                }
        System.out.println("Total length of Offices in this street is: " + lengthCounter);//O(1)
            break;//O(1)
```

```
case "House":

    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(1)
            if(classname.equals("House")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(1)
            }
        }
        else
            continue;

    }
    System.out.println("Total length of Houses in this street is: " + lengthCounter);//O(1)
    break;
```

```
case "Market":

    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(1)
            if(classname.equals("Market")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(1)
            }
        }
        else
            continue;//O(1)

    }
    System.out.println("Total length of Markets in this street is: " + lengthCounter);//O(1)
    break;
```

O(N^2)

displayNumberAndRationOfPlaygrounds() Method

```
public void displayNumberAndRationOfPlaygrounds(){
    float lengthCounter = 0; int numberOfPlayground =0;//O(1)
    String classname; float ratio;//O(1)
    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(1)
            if(classname.equals("Playground")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(1)
                numberOfPlayground++;//O(1)
            }
        }
    }
    ratio = lengthCounter/this.totalLength;//O(1)
    System.out.println("Total Playgrounds Number: " + numberOfPlayground + " Ratio of Playgrounds: " + ratio);//O(1)
}
```

O(N^2)

displaySkylineSilhouette() Method

```
public void displaySkylineSilhouette(){
    int maxHeight = 0;
    int maxWidth = 0;

    for(int i = 0;i<streetArray.size();i++){//O(N)
        maxHeight = maxHeight > streetArray.get(i).getHeight() ? maxHeight : streetArray.get(i).getHeight();//O(N)
        maxWidth = maxWidth > streetArray.get(i).getLength() + streetArray.get(i).getPosition() ? maxWidth : streetArray.get(i).getLength() + streetArray.get(i).getPosition();//O(N)
    }

    //int [] totalHeights = new int[maxWidth];
    LinkedList <Integer> totalHeights = new LinkedList<~>();
    for(int i=0;i<maxWidth;i++){//O(N)
        totalHeights.add(i, element: 0);//O(1)
    }

    for(int i = 0 ;i<streetArray.size();i++){//O(N)
        for(int j = streetArray.get(i).getPosition();j<streetArray.get(i).getPosition()+streetArray.get(i).getLength();j++){//O(N)
                totalHeights.set(j, totalHeights.get(j) > streetArray.get(i).getHeight() ? totalHeights.get(j) : streetArray.get(i).getHeight());//O(N)
        }
    }

    for(int i = 0;i<maxHeight;i++){//O(N)
        for(int j = 0;j<maxWidth;j++){//O(N)
            if(maxHeight - i > totalHeights.get(j)){//O(N)
                System.out.print(' ');//O(1)
            }
            else
                System.out.print('#');//O(1)
        }
        System.out.println();//O(1)
    }
```

O(N^3)

-For LDLinkedList

```java
public boolean add(E data)
{

    // Create a new node with given data
    Node new_node = new Node(data);//O(1)
    //last points to last node of LDLinkedList.
    Node last = this.head;//O(1)
    if (this.head == null) {//O(1)
        this.head = new_node;//O(1)
        return true;
    }

    while(last.next != null){//O(N)
        last=last.next;//O(1)
    }

    if(deletedContent == null){//O(1)
        last.next = new_node;//O(1)
        return true;

    }
    if(deletedContent!= null){//O(1)
        current=deletedContent.head;//O(1)
        while(current != null){//O(N)
            if(data == current.data){//O(1)
                last.next=current;//O(1)
                last.next.next = null;//O(1)
                System.out.println("The object/build found in the Deleted Content!");//O(1)
                return true;
            }
            current=current.next;//O(1)
        }

    }

    last.next = new_node;//if couldn't find the create new node and go for it !
    return true;
}
```

O(N)

```java
@Override
public E get(int index) {
    int traverseIndex = 0;//O(1)
    current = this.head;//O(1)

    while(current != null){//O(N)
        if(traverseIndex == index){//O(1)
            return current.data;//O(1)
        }
        traverseIndex++;//O(1)
        current = current.next;//O(1)

    }
    return null;//O(1)
}
```

O(N)

```java
    @Override
    public E set(int index, E element) {
        int traverseIndex = 0;//O(1)
        current = this.head;//O(1)

        while(current != null){//O(N)
            if(traverseIndex == index){//O(1)
                current.data = element;//O(1)
                return element;//O(1)
            }
            traverseIndex++;//O(1)
            current = current.next;//O(1)

        }
        return null;//O(1)
    }
```

O(N)

```java
public E remove(int index)
{
    if(deletedContent==null){ deletedContent = new LDLinkedList<E>();}//for the first call.O(1)
    current = this.head;//O(1)
    int currentIndex = 0;//O(1)
    E willDelete = null;//O(1)

    Node previousCurrent = null;

    while(current != null){//O(N)
        if(currentIndex == index - 1){
            previousCurrent=current;//O(1)
            deletedContent.add((E) current.next.data);//O(N)
            previousCurrent.next=current.next.next;//O(1)
        }
        if(currentIndex == index){

                willDelete = current.data;//O(1)
                current.next=null;//O(1)
                return willDelete;//O(1)
        }
        currentIndex++;//O(1)
        current=current.next;//O(1)
    }

    return willDelete;//O(1)
}
```

O(N^2)

```java
    @Override
    public int size() {
        int size = 0;//O(1)
        current = this.head;//O(1)
        while(current != null){//O(N)
            size++;//O(1)
            current=current.next;//O(1)
        }
        return size;//O(1)
    }
}
```

O(N)

```java
public void add(Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
    streetArray.add(build);//O(N)
    //streetArray[buildingsLastIndex] = build;
    currentLength += build.getLength();//O(1)
    //buildingsLastIndex++;
    buildingsNumber++;//O(1)
    }}
```

O(N)

```java
public void addByIndex(int i, Building build){
    if(this.currentLength + build.getLength() > totalLength || this.currentLength == totalLength || build.getLength() > this.totalLength){//O(1)
        System.out.println("There is not enough space for add a new build.");//O(1)
    }
    else{
        if(i< streetArray.size()){//O(N)
            streetArray.set(i,build);//O(N)
            //streetArray[i] = build;
            currentLength += build.getLength();//O(1)
            //buildingsLastIndex++;
            buildingsNumber++;//O(1)
        }
        else{
            streetArray.add(build);//O(N)
            //streetArray[i] = build;
            currentLength += build.getLength();//O(1)
            //buildingsLastIndex++;
            buildingsNumber++;//O(1)
        }

    }
}
```

O(N)

```java
public void delete(int Index){
    if(streetArray.get(Index) == null){//O(1)
        System.out.println("The building that you try to delete is not exist! Please enter a valid Index/Position!");//O(1)
    }
    else{
        currentLength -= streetArray.get(Index).getLength();//O(1)
        streetArray.remove(Index);//O(N^2)
        //streetArray[Index] = null;
        buildingsNumber--;//O(1)
    }
}
```

O(N^2)

```java
public void displayBuildings(){
    String classname;//O(1)
    if(buildingsNumber == 0 ){//O(1)
        System.out.println("There is no build in this street! ");//O(1)
    }
    else{
        for(int i=0; i<=streetArray.size(); i++){//O(N)
            if(streetArray.get(i) != null ){//O(N)
                classname = streetArray.get(i).getClass().getSimpleName();//O(1)
                System.out.println("Slot: " + i + " Type: " + classname + " Length: " + streetArray.get(i).getLength());//O(N)
            }
        }
    }

}
```

O(N^2)

```java
public void totalLengthOfSpecificBuildings(){
    int lengthCounter = 0;//O(1)
    String buildType;//O(1)
    if(buildingsNumber == 0){//O(1)
        System.out.println("There is no building in this street");//O(1)
    }
    else{
        System.out.println("Please enter name of building to calculate total length with case sensitive (Office, House, Market): ");//O(1)
        Scanner sc = new Scanner(System.in);//O(1)
        String classname;//O(1)
        buildType = sc.nextLine();//O(1)
        switch(buildType){//O(1)
            case "Office"://O(1)
                for(int i=0; i<streetArray.size(); i++){//O(N)
                    if(streetArray.get(i) != null){//O(N)
                    classname = streetArray.get(i).getClass().getSimpleName();//O(N)
                        if(classname.equals("Office")){//O(1)
                            lengthCounter += streetArray.get(i).getLength();//O(N)
                        }
                    }

                }
        System.out.println("Total length of Offices in this street is: " + lengthCounter);//O(1)
            break;
```

```java
case "House":

    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(N)
            if(classname.equals("House")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(N)
            }
        }
        else
            continue;//O(1)

    }
    System.out.println("Total length of Houses in this street is: " + lengthCounter);//O(1)
    break;
```

```java
case "Market":

    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(N)
            if(classname.equals("Market")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(N)
            }
        }
        else
            continue;//O(1)

    }
    System.out.println("Total length of Markets in this street is: " + lengthCounter);//O(1)
    break;//O(1)
```

O(N^2)

```
public void displayNumberAndRationOfPlaygrounds(){
    float lengthCounter = 0; int numberOfPlayground =0;//O(1)
    String classname; float ratio;//O(1)
    for(int i=0; i<streetArray.size(); i++){//O(N)
        if(streetArray.get(i) != null){//O(N)
            classname = streetArray.get(i).getClass().getSimpleName();//O(N)
            if(classname.equals("Playground")){//O(1)
                lengthCounter += streetArray.get(i).getLength();//O(N)
                numberOfPlayground++;//O(1)
            }
        }
    }
    ratio = lengthCounter/this.totalLength;//O(1)
    System.out.println("Total Playgrounds Number: " + numberOfPlayground + " Ratio of Playgrounds: " + ratio);//O(1)
}
```

O(N^2)

```
public void displaySkylineSilhouette(){
    int maxHeight = 0;//O(1)
    int maxWidth = 0;//O(1)

    for(int i = 0;i<streetArray.size();i++){//O(N)
        maxHeight = maxHeight > streetArray.get(i).getHeight() ? maxHeight : streetArray.get(i).getHeight();//O(N)
        maxWidth = maxWidth > streetArray.get(i).getLength() + streetArray.get(i).getPosition() ? maxWidth : streetArray.get(i).getLength() + streetArray.get(i).getPosition();//O(N)
    }

    //int [] totalHeights = new int[maxWidth];
    LDLinkedList <Integer> totalHeights = new LDLinkedList<~>();
    for(int i=0;i<maxWidth;i++){//O(N)
        totalHeights.add(0);//O(1)
    }

    for(int i = 0 ;i<streetArray.size();i++){//O(N)
        for(int j = streetArray.get(i).getPosition();j<streetArray.get(i).getPosition()+streetArray.get(i).getLength();j++){//O(N)
                totalHeights.set(j, totalHeights.get(j) > streetArray.get(i).getHeight() ? totalHeights.get(j) : streetArray.get(i).getHeight());//O(N)
        }
    }
    /*System.out.println(totalHeights.size());
    for(int i = 0 ; i< totalHeights.size();i++){
        System.out.println(i);
        System.out.print(totalHeights.get(i) + " ");
    }
    System.out.println();*/

    for(int i = 0;i<maxHeight;i++){//O(N)
        for(int j = 0;j<maxWidth;j++){//O(N)
            if(maxHeight - i > totalHeights.get(j)){//O(N)
                System.out.print(' ');//O(1)
            }
            else
                System.out.print('#');//O(1)
        }
        System.out.println();//O(1)
    }
```

O(N^3)

5. Calculations:

```java
System.out.println("~Test Case 13 speeds:~");
long startTime = System.nanoTime();
for(int i=0; i<10; i++){
    myTestArray[i] = i;
}
long endTime = System.nanoTime();
long duration = (endTime - startTime);

System.out.println("Array 10 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<10; i++){
    myArrayListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("ArrayList 10 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<10; i++){
    myLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LinkedList 10 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<10; i++){
    myLDLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LDLinkedList 10 element addition time: "+duration+" ns");
```

```
startTime = System.nanoTime();
for(int i=0; i<100; i++){
    myTestArray[i] = i;
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println();
System.out.println("Array 100 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<100; i++){
    myArrayListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("ArrayList 100 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<100; i++){
    myLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LinkedList 100 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<100; i++){
    myLDLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LDLinkedList 100 element addition time: "+duration+" ns");
```

```
startTime = System.nanoTime();
for(int i=0; i<1000; i++){
    myTestArray[i] = i;
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println();
System.out.println("Array 1000 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<1000; i++){
    myArrayListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("ArrayList 1000 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<1000; i++){
    myLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LinkedList 1000 element addition time: "+duration+" ns");

startTime = System.nanoTime();
for(int i=0; i<1000; i++){
    myLDLinkedListTest.add(i);
}
endTime = System.nanoTime();
duration = (endTime - startTime);

System.out.println("LDLinkedList 1000 element addition time: "+duration+" ns");
```

```
~Test Case 13 speeds:~
Array 10 element addition time: 700 ns
ArrayList 10 element addition time: 11500 ns
LinkedList 10 element addition time: 7500 ns
LDLinkedList 10 element addition time: 6899 ns

Array 100 element addition time: 1700 ns
ArrayList 100 element addition time: 51500 ns
LinkedList 100 element addition time: 38000 ns
LDLinkedList 100 element addition time: 89199 ns

Array 1000 element addition time: 10200 ns
ArrayList 1000 element addition time: 158900 ns
LinkedList 1000 element addition time: 274499 ns
LDLinkedList 1000 element addition time: 1431200 ns
```

When we look at the increase rates, you can see that we get a very parallel result to the time complexity we calculated above. Best to worst, when we rank the time complexity:

We can say Array>ArrayList>LinkedList>LDLinkedList.