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

BURAK KOCAUSTA
1901042605

1. SYSTEM REQUIREMENTS

Street is named “<TYPE>Street” in this City Planning Software. TYPE can be replaced with ArrayList, LinkedList, LDLinkedList or BasicArray It needs two sides to hold buildings. There should be length information to construct to street initially, then buildings can be added to wanted sides.

```
BasicArrayStreet street1 = new BasicArrayStreet( 55 );
```

```
LDLinkedListStreet street1 = new LDLinkedListStreet( 55 );
```

```
ArrayListStreet street1 = new ArrayListStreet( 55 );
```

```
LinkedListStreet street1 = new LinkedListStreet( 55 );
```

In this class buildings are named as “CityBuilding”. There must be position, length information about buildings and street to prevent any conflict between buildings. Sometimes building’s length can be greater than street’s length or some of the building’s position information are not suitable for street. So, position and length information of buildings are required. To print Skyline Silhouette, there must be height information of buildings.

This can be called from derived classes constructors like super(…)

```
/** ...
public CityBuilding ( int position, int length, int height )
```

There is no limit for height in this software, but length of street, and buildings are immutable. If you want to change length, you must create new street or building. Building’s position is also immutable, because during insertion of the buildings to street some contracts(conditions) are checked. After insertions, these changes shouldn’t be done from outside. There is delete option in edit mode of street. In this mode that kind of changings can be done.

Building is thought as an abstract notion. Many things can be derived from building. In this software, there is house, market, office, and playgrounds. CityBuilding is an abstract reusable class. Other concrete classes are derived from this.

A house requires an owner, color, and number of the room information.

```
CityBuilding house1 = new House( 6, 7, 5, "burak", "green", 3 );
```

A market requires an owner, opening and closing time information.

```
CityBuilding market1 = new Market( 0, 8, 20, "james" , "08:00" , "21:00" );
```

An office also requires an owner, and job type information.

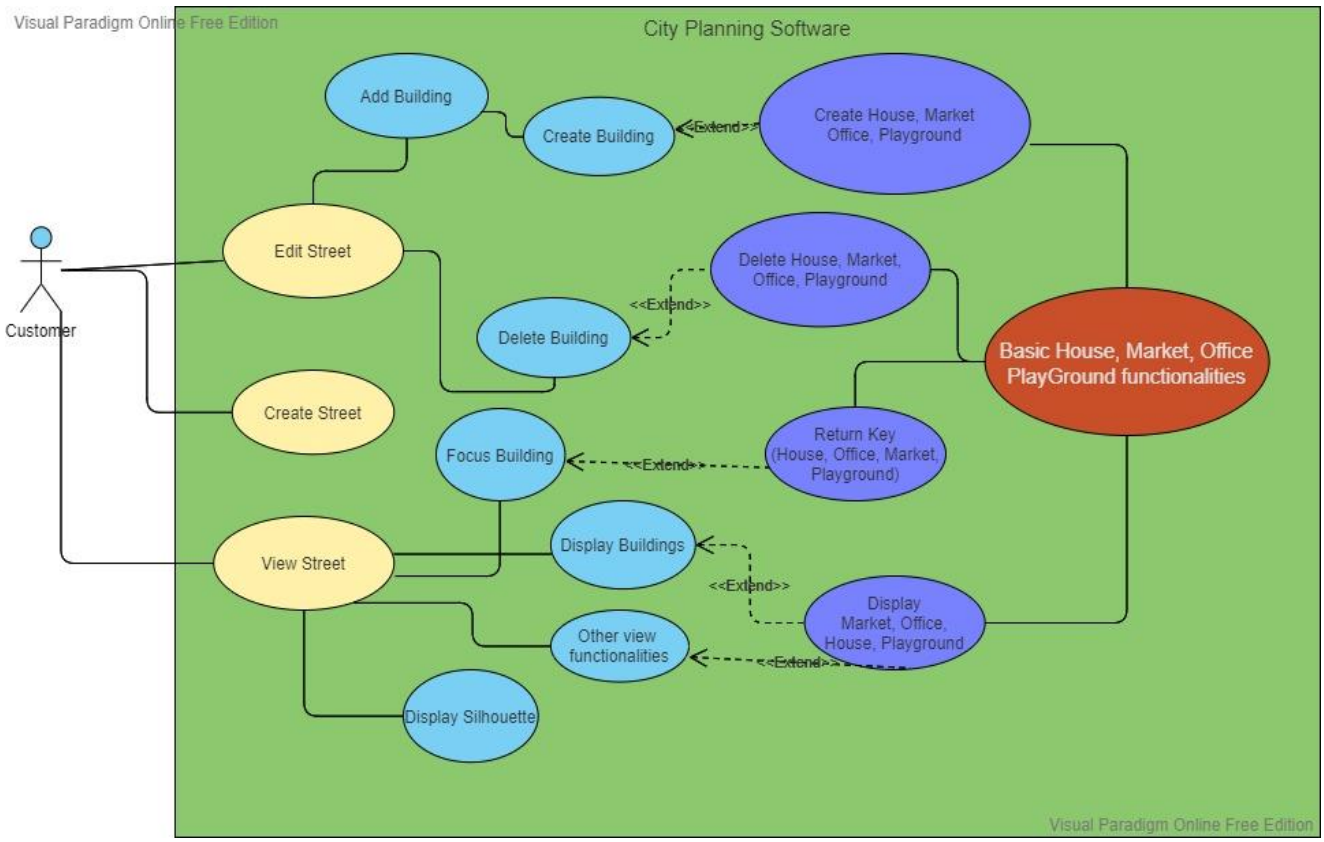
```
CityBuilding office1 = new Office( 10, 12, 20, "rachel", "consulting" );
```

A playground is a simple extension of building it does not require an extra information like others, but its height is automatically 1.

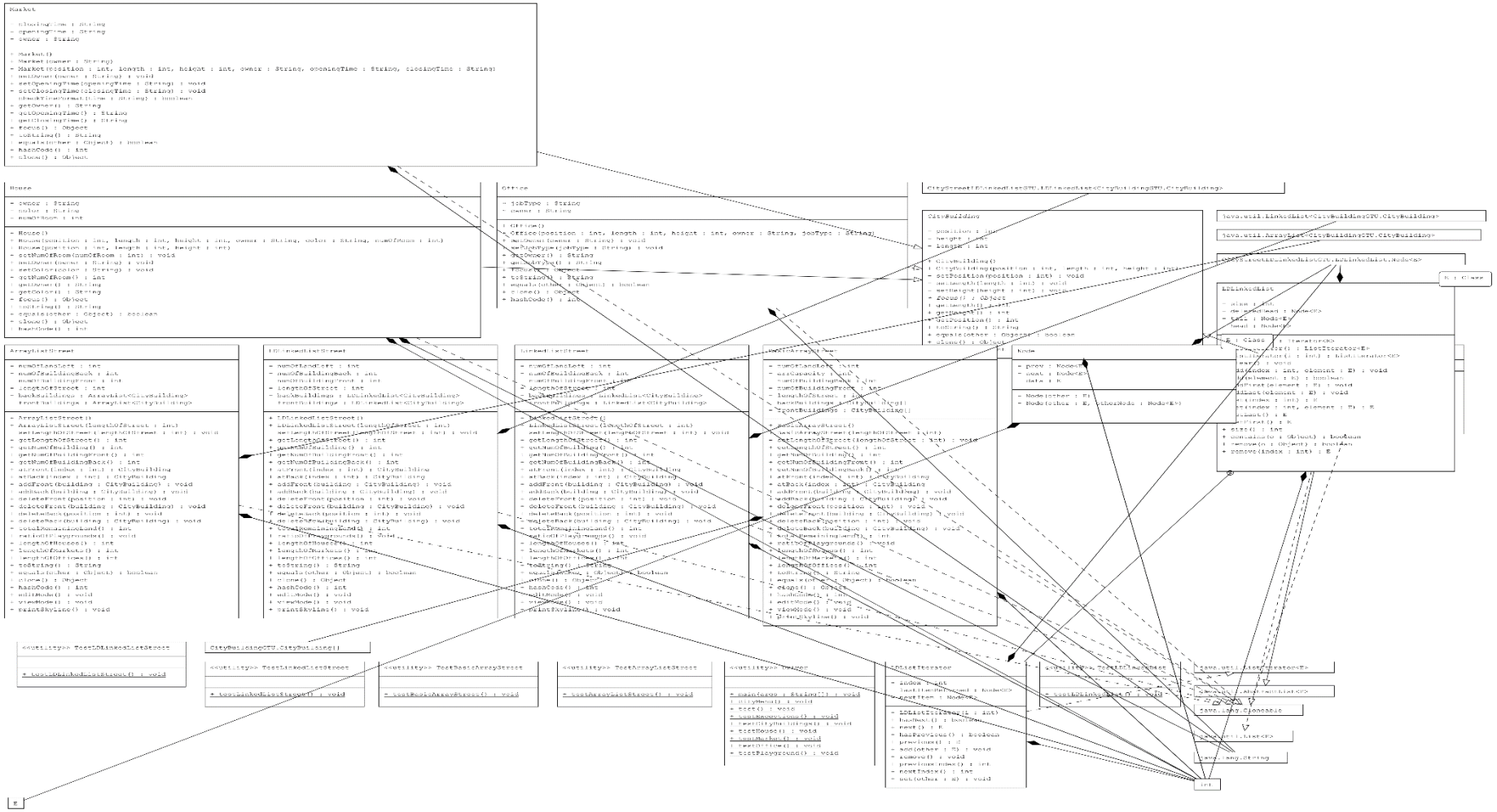
```
CityBuilding playground1 = new Playground ( 31, 4 );
```

2. **USE CASE AND CLASS DIAGRAMS**

USE CASE DIAGRAM



CLASS DIAGRAM (High resolution version is inside this directory as .png file)



3. PROBLEM SOLUTION APPROACH

In HW3 I decided to create different types of street classes and name them according to their container type. Therefore, I changed the name cityStreet to BasicArrayStreet, and named the others like ArrayListStreet, LinkedListStreet, LDLinkedListStreet. For LDLinkedListStreet, I made a class which is LDLinkedList, and it extends AbstractList of java collection. Inside this class I created a custom iterator class and it implements ListIterator class. I make all the remove and add operations with iterators. This class differs from LinkedList with lazy deletion strategy. Inside the iterators, remove function (even if it is inside iterator, when add and delete operation is made using list iterators will be called) adds the removed node to another linked list for lazy deletion.

```
private Node<E> deletedHead = null; // head of the deleted nodes.
```

when adding will be made, first this node is checked if it is null, then proceed.

My design is pretty similar to first homework, There is an abstract building class which is superclass of house, market, office, and playground. In first homework only dynamically growing array is used to hold the data. This homework I made different kind of streets as I indicated above. Apart from first homework I decided to change printSkyline() function, because it is very complex and hard to understand. In first homework I printed it character by character, and in each step I checked all the conditions. So it has n^4 complexity. In this version, I updated it to n^2. First I hold the positions and heights, then printing them with some condition checks. Apart from it, everything is the same for BasicArrayStreet for the first version.

For new classes, ArrayList version is pretty similar to BasicArray but for ArrayList there is no need to handle dynamic growing. It handles it by itself. So, program became simpler with ArrayList. In LinkedList and LDLinkedList I changed get() methods which are inside loops to iterator version. If I didn't do that loops complexity were quadratic. With this change it is still linear.

4. Complexity Calculations

THEROTICAL RUNNING TIMES

Front and Back means sides of the street, not head and tail.

Methods/Classes	BasicArrayStreet	ArrayListStreet	LinkedListStreet	LDLinkedListStreet
atFront() atBack()	Θ(1)	Θ(1)	O(n)	O(n)
addFront() (tail) addBack() (tail)	Θ(n)	Θ(n)	Θ(n)	Θ(n)
deleteFront() deleteBack()	Θ(n)	Θ(n)	O(n)	O(n)
toString()(list buildings)	Θ(n)	Θ(n)	Θ(n)	Θ(n)
equals()	Θ(n^2)	Θ(n^2)	Θ(n^2)	Θ(n^2)
clone()	Θ(n)	Θ(n)	Θ(n)	Θ(n)
printSkyline()	Θ(n^2)	Θ(n^2)	Θ(n^2)	Θ(n^2)
ratioOfPlaygrounds()	Θ(n)	Θ(n)	Θ(n)	Θ(n)
lengthOfHouses()	Θ(n)	Θ(n)	Θ(n)	Θ(n)
lengthOfMarkets()	Θ(n)	Θ(n)	Θ(n)	Θ(n)
focus Part inside ViewMode()	Θ(1)	Θ(1)	O(n)	O(n)
getLengthOfStreet() getNumOfBuilding()	Θ(1)	Θ(1)	Θ(1)	Θ(1)

EXPERIMENTAL RUNNING TIMES

add methods (addFront(), addBack())

They are all Θ(n) because of the reference checking.

```
add methods running times(nanoseconds) for size = 1000
BasicArrayStreet = 14087900
LinkedListStreet = 22640300
ArrayListStreet = 25431300
LDLinkedListStreet = 31912500

add methods running times(nanoseconds) for size = 2000
BasicArrayStreet = 20367500
LinkedListStreet = 9853600
ArrayListStreet = 9864300
LDLinkedListStreet = 10184300

add methods running times(nanoseconds) for size = 4000
BasicArrayStreet = 30933700
LinkedListStreet = 40725000
ArrayListStreet = 40573300
LDLinkedListStreet = 44630500

add methods running times(nanoseconds) for size = 8000
BasicArrayStreet = 115847500
LinkedListStreet = 149102600
ArrayListStreet = 154124600
LDLinkedListStreet = 196987400

add methods running times(nanoseconds) for size = 16000
BasicArrayStreet = 496607300
LinkedListStreet = 638981700
ArrayListStreet = 616703800
LDLinkedListStreet = 761848600

add methods running times(nanoseconds) for size = 32000
BasicArrayStreet = 1964011500
LinkedListStreet = 2595465700
ArrayListStreet = 2368196000
LDLinkedListStreet = 3432239000
```

Delete methods (deleteFront(), deleteBack())

LinkedList and LDLinked is faster than arrays, and increasing is linear.

```
delete methods running times(nanoseconds) for size = 1000
BasicArrayStreet = 5704200
LinkedListStreet = 439400
ArrayListStreet = 578500
LDLinkedListStreet = 199000

delete methods running times(nanoseconds) for size = 2000
BasicArrayStreet = 10227500
LinkedListStreet = 120900
ArrayListStreet = 1086900
LDLinkedListStreet = 438500

delete methods running times(nanoseconds) for size = 4000
BasicArrayStreet = 25530500
LinkedListStreet = 296400
ArrayListStreet = 662800
LDLinkedListStreet = 232700

delete methods running times(nanoseconds) for size = 8000
BasicArrayStreet = 112146700
LinkedListStreet = 561300
ArrayListStreet = 2202800
LDLinkedListStreet = 319300

delete methods running times(nanoseconds) for size = 16000
BasicArrayStreet = 489883500
LinkedListStreet = 385000
ArrayListStreet = 7209000
LDLinkedListStreet = 457100

delete methods running times(nanoseconds) for size = 32000
BasicArrayStreet = 2011171900
LinkedListStreet = 742800
ArrayListStreet = 33753900
LDLinkedListStreet = 853000
```

At methods (atFront(), atBack())

```
at methods running times(nanoseconds) for size = 1000
BasicArrayStreet = 202700
LinkedListStreet = 1054900
ArrayListStreet = 295000
LDLinkedListStreet = 2391100

at methods running times(nanoseconds) for size = 2000
BasicArrayStreet = 118500
LinkedListStreet = 3277400
ArrayListStreet = 131300
LDLinkedListStreet = 9617000

at methods running times(nanoseconds) for size = 4000
BasicArrayStreet = 270900
LinkedListStreet = 15359800
ArrayListStreet = 272500
LDLinkedListStreet = 63150700

at methods running times(nanoseconds) for size = 8000
BasicArrayStreet = 410400
LinkedListStreet = 125976400
ArrayListStreet = 691700
LDLinkedListStreet = 251625200

at methods running times(nanoseconds) for size = 16000
BasicArrayStreet = 177300
LinkedListStreet = 605604300
ArrayListStreet = 603000
LDLinkedListStreet = 1068087200
```

```
at methods running times(nanoseconds) for size = 32000
BasicArrayStreet = 1948100
LinkedListStreet = 2686679400
ArrayListStreet = 1247600
LDLinkedListStreet = 7696702700
```

printSkyline() method

```
printSkyline() running times(nanoseconds) for size = 1000
BasicArrayStreet = 5198200
LinkedListStreet = 6378500
ArrayListStreet = 31018900
LDLinkedListStreet = 5802200

printSkyline() running times(nanoseconds) for size = 2000
BasicArrayStreet = 2744100
LinkedListStreet = 11566400
ArrayListStreet = 24842500
LDLinkedListStreet = 4027800

printSkyline() running times(nanoseconds) for size = 4000
BasicArrayStreet = 14413900
LinkedListStreet = 24940100
ArrayListStreet = 131445200
LDLinkedListStreet = 12729200

printSkyline() running times(nanoseconds) for size = 8000
BasicArrayStreet = 32170900
LinkedListStreet = 94153100
ArrayListStreet = 125702300
LDLinkedListStreet = 42398500

printSkyline() running times(nanoseconds) for size = 16000
BasicArrayStreet = 68025500
LinkedListStreet = 110954000
ArrayListStreet = 453394800
LDLinkedListStreet = 125757800

printSkyline() running times(nanoseconds) for size = 32000
BasicArrayStreet = 276360300
LinkedListStreet = 497009600
ArrayListStreet = 1959321000
LDLinkedListStreet = 520030000
```

5. TEST CASES

Testing ArrayListStreet class

```
public static void testArrayListStreet ( ) {
    try {
        System.out.println( "___TESTING ArrayListStreet class___\n" );

        ArrayListStreet street1 = new ArrayListStreet( 55 );

        System.out.println( "\nEmpty Street created.\n" );
        System.out.printf( "Length of Street is = %d", street1.getLengthOfStreet() );

        System.out.println( street1 );
        System.out.println( "--------------------------\n" );

        CityBuilding house1 = new House( 6, 7, 5, "burak", "green", 3 );
        System.out.println( "\nHouse created\n" );
        System.out.println( house1 );
        System.out.println( "--------------------------\n" );

        CityBuilding market1 = new Market( 0, 8, 20, "james" , "08:00" , "21:00" );
        System.out.println( "\nMarket created\n" );
        System.out.println( market1 );
        System.out.println( "--------------------------\n" );

        CityBuilding office1 = new Office( 10, 12, 20, "rachel", "consulting" );
        System.out.println( "\nOffice created\n" );
        System.out.println( office1 );
        System.out.println( "--------------------------\n" );

        CityBuilding playground1 = new Playground ( 31, 4 );
        System.out.println( "\nPlayground created\n" );
        System.out.println( playground1 );
        System.out.println( "--------------------------\n" );
```

```
    street1.addFront( house1 );
    street1.addBack( market1 );
    street1.addBack( office1 );
    street1.addFront( playground1 );

    System.out.println( "\nhouse, market, office, and playground are added to street.\n" );
    System.out.println( street1 );
    System.out.println( "--------------------------\n" );

    street1.deleteFront( house1 );
    street1.deleteFront( playground1 );
    street1.deleteBack( 10 );
    street1.deleteBack( 0 );

    System.out.println( "\nAll buildings are removed from class with 4 overloaded delete methods.\n" );
    System.out.println( street1 );
    System.out.println( "--------------------------\n" );

    street1.addFront( market1 );
    street1.addBack( house1 );
    street1.addBack( playground1 );
    street1.addFront( office1 );

    System.out.println( "\nBuildings are added to street oppositely.\n" );
    System.out.println( street1 );
    System.out.printf( "Total remaining lands = %d\n", street1.totalRemainingLand() );
    System.out.printf( "Total number of Buildings = %d\n", street1.getNumOfBuilding() );
    street1.ratioOfPlaygrounds();
    System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
            , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );
    System.out.println( "--------------------------\n" );

    street1.printSkyline();
```

```
street1.addFront( new House( 36, 10 , 10,  "olivia", "yellow", 3 ) );
street1.addFront( new House( 48, 7, 10, "ahmet", "gray", 2 ));
street1.addBack( new House( 17, 13 , 30, "elif", "black", 3  ) );
street1.addBack( new Market( 43, 8 , 20, "george", "06:30", "19:00" ) );
System.out.println( "\nNew buildings are added.\n" );

System.out.println( street1 );
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
          , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );

street1.printSkyline();

System.out.println( "Testing focus() method for every building. Called from CityBuilding array." );

for ( int i = 0; i < street1.getNumOfBuildingFront(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atFront(i).focus() );

for ( int i = 0; i < street1.getNumOfBuildingBack(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atBack(i).focus() );

System.out.println( "---------------------------\n" );

System.out.println( "\nTesting clone() method." );

ArrayListStreet street2 = ( ArrayListStreet ) street1.clone();

System.out.println( "Original street\n"  + street1 );
System.out.println( "Cloned street\n" + street2 );

System.out.println( "---------------------------\n" );

System.out.printf( "\nTesting equals() method\nresult of street1.equals(street2) is = " );

System.out.println( street1.equals(street2) );
```

```
System.out.println( "\nDeleting one building from street2\n" );
street2.deleteBack(playground1);
System.out.println( street2 );

System.out.println( "After deleting result of condition is: " + street1.equals(street2) );
System.out.println( "---------------------------\n" );

System.out.println( "street1.hashCode() = " + street1.hashCode() + "\n" );
```

Testing LDLinkedList class

```
System.out.println( "___TESTING LDLinkedList class___\n" );

System.out.println( "Creating and adding elements." );
LDLinkedList<String> list = new LDLinkedList<String>();
list.add("elma");
list.add("armut");
list.add("kavun");
list.add(2,"kahve");
ListIterator<String> itr = list.listIterator();

System.out.println( "size() returns = " + list.size());
while ( itr.hasNext() ) {
    String tool = itr.next();
    System.out.println(tool);
}
System.out.println( "\nadd methods of LDLinkedList and hasNext(), next() methods of custom iterator are tested." );
System.out.println( "Removing some elements with iterators remove() method and lists remove methods.\n" );
list.remove(0);
list.remove("kahve");
itr = list.listIterator();
itr.next();
itr.remove();

itr = list.listIterator();
System.out.println( "size() returns = " + list.size());
while ( itr.hasNext() ) {
    String tool = itr.next();
    System.out.println(tool);
}
```

```
System.out.println( "\nSome Elements removed successfully.\nAdding elements using other methods.\n" );

itr = list.listIterator();
itr.add("elma");
itr.add("armut");
list.addLast("muz");
list.addFirst("elma2");

System.out.print("\n");
System.out.println( "size() returns = " + list.size());
itr = list.listIterator();
while ( itr.hasNext() ) {
    String tool = itr.next();
    System.out.println(tool);
}
System.out.println( "add method of custom iterator, addLast, addFirst methods of LDLinkedList are tested.\n" );
System.out.println( "list will be printed backwards using previous, hasprevious." );

itr = list.listIterator ( list.size() );
System.out.println( "size() returns = " + list.size());
while ( itr.hasPrevious() ) {
    String tool = itr.previous();
    System.out.println(tool);
}
System.out.println( "previous, hasprevious methods are tested.\n\nset methods will be used." );
```

```
list.set(2, "armut2");
itr = list.listIterator();
itr.next();
itr.set("karpuz");
for( String obj : list )
    System.out.println ( obj );

System.out.println( "set methods are tested and range based for loop worked fine.\n" );


System.out.println( "list.getLast() = " + list.getLast() + "\nlist.getFirst() = " + list.getFirst() );
System.out.println( "\nlist.get(2) = " + list.get(2) + "\nlist.contains(\"karpuz\") = " + list.contains("karpuz") + "\n" );


Iterator<String> itr2 = list.iterator();
while ( itr2.hasNext() ) {
    String tool = itr2.next();
    System.out.println(tool);
}

System.out.println( "\nCalling list.clear()" );
list.clear();
System.out.println( "list.size() = " + list.size() );

System.out.println( "getters, iterator(), clear() methods are tested.\n" );

System.out.println( "Testing with different class which is Integer class.\n" );

LDLinkedList<Integer> list2 = new LDLinkedList<Integer>();

ListIterator<Integer> itr3 = list2.listIterator();

System.out.println( "\nAdd some elements using iterators." );
```

```
itr3.add (Integer.valueOf(3));
itr3.add (Integer.valueOf(8));
itr3.add (Integer.valueOf(11));
System.out.println( "size() returns = " + list2.size());
for ( Integer obj : list2 )
    System.out.print(obj + " ");

System.out.print("\n");

System.out.println( "\nAdd more numbers, and delete some using different scenarios." );
for ( int i = 0; i < 33; ++i )
    itr3.add(Integer.valueOf(i*3));

itr3 = list2.listIterator();
System.out.println( "\nBefore removing: size() returns = " + list2.size());
while (itr3.hasNext()) {
    Integer val = itr3.next();
    System.out.print( val + " " );
}

System.out.print("\n");

itr3 = list2.listIterator(list2.size());
itr3.previous();
itr3.next();
itr3.remove();

itr3 = list2.listIterator();

itr3.next();
itr3.remove();
```

```
itr3 = list2.listIterator(5);

itr3.next();
itr3.remove();

itr3.previous();
itr3.remove();


itr3 = list2.listIterator();
System.out.println( "\nAfter Removing: size() returns = " + list2.size());
while (itr3.hasNext()) {
    Integer val = itr3.next();
    System.out.print( val + " " );
}

System.out.println( "\n\n--All functionalities of LDLinkedList class are tested.\n" );
```

TestingLDLinkedListStreet

```
System.out.println( "___TESTING LDLinkedListStreet class___\n" );

LDLinkedListStreet street1 = new LDLinkedListStreet( 55 );

System.out.println( "\nEmpty Street created.\n" );
System.out.printf( "Length of Street is = %d", street1.getLengthOfStreet() );

System.out.println( street1 );
System.out.println( "---------------------------\n" );

CityBuilding house1 = new House( 6, 7, 5, "burak", "green", 3 );
System.out.println( "\nHouse created\n" );
System.out.println( house1 );
System.out.println( "---------------------------\n" );

CityBuilding market1 = new Market( 0, 8, 20, "james" , "08:00" , "21:00" );
System.out.println( "\nMarket created\n" );
System.out.println( market1 );
System.out.println( "---------------------------\n" );

CityBuilding office1 = new Office( 10, 12, 20, "rachel", "consulting" );
System.out.println( "\nOffice created\n" );
System.out.println( office1 );
System.out.println( "---------------------------\n" );

CityBuilding playground1 = new Playground ( 31, 4 );
System.out.println( "\nPlayground created\n" );
System.out.println( playground1 );
System.out.println( "---------------------------\n" );
```

```
street1.addFront( house1 );
street1.addBack( market1 );
street1.addBack( office1 );
street1.addFront( playground1 );

System.out.println( "\nhouse, market, office, and playground are added to street.\n" );
System.out.println( street1 );
System.out.println( "---------------------------\n" );


street1.deleteFront( house1 );
street1.deleteFront( playground1 );
street1.deleteBack( 10 );
street1.deleteBack( 0 );

System.out.println( "\nAll buildings are removed from class with 4 overloaded delete methods.\n" );
System.out.println( street1 );
System.out.println( "---------------------------\n" );

street1.addFront( market1 );

street1.addBack( house1 );

street1.addBack( playground1 );
street1.addFront( office1 );

System.out.println( "\nBuildings are added to street oppositely.\n" );
```

```
street1.addFront( market1 );

street1.addBack( house1 );

street1.addBack( playground1 );
street1.addFront( office1 );

System.out.println( "\nBuildings are added to street oppositely.\n" );
System.out.println( street1 );
System.out.printf( "Total remaining lands = %d\n", street1.totalRemainingLand() );
System.out.printf( "Total number of Buildings = %d\n", street1.getNumOfBuilding() );
street1.ratioOfPlaygrounds();
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
            , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );
System.out.println( "---------------------------\n" );

street1.printSkyline();

street1.addFront( new House( 36, 10 , 10,  "olivia", "yellow", 3 ) );
street1.addFront( new House( 48, 7, 10, "ahmet", "gray", 2 ));
street1.addBack( new House( 17, 13 , 30, "elif", "black", 3  ) );
street1.addBack( new Market( 43, 8 , 20, "george", "06:30", "19:00" ) );
System.out.println( "\nNew buildings are added.\n" );

System.out.println( street1 );
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
            , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );

street1.printSkyline();

System.out.println( "Testing focus() method for every building. Called from CityBuilding array." );
```

```
System.out.println( "Testing focus() method for every building. Called from CityBuilding array." );

for ( int i = 0; i < street1.getNumOfBuildingFront(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atFront(i).focus() );

for ( int i = 0; i < street1.getNumOfBuildingBack(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atBack(i).focus() );

System.out.println( "---------------------------\n" );

System.out.println( "\nTesting clone() method." );

LDLinkedListStreet street2 = ( LDLinkedListStreet ) street1.clone();

System.out.println( "Original street\n"  + street1 );
System.out.println( "Cloned street\n" + street2 );

System.out.println( "---------------------------\n" );

System.out.printf( "\nTesting equals() method\nresult of street1.equals(street2) is = " );

System.out.println( street1.equals(street2) );
System.out.println( "---------------------------\n" );

System.out.println( "\nDeleting one building from street2\n" );
street2.deleteBack(playground1);
System.out.println( street2 );

System.out.println( "After deleting result of condition is: " + street1.equals(street2) );
System.out.println( "---------------------------\n" );

System.out.println( "street1.hashCode() = " + street1.hashCode() + "\n" );
```

Testing LinkedListStreet

```
System.out.println( "___TESTING LinkedListStreet class___\n" );

LinkedListStreet street1 = new LinkedListStreet( 55 );

System.out.println( "\nEmpty Street created.\n" );
System.out.printf( "Length of Street is = %d", street1.getLengthOfStreet() );

System.out.println( street1 );
System.out.println( "----------------------------\n" );

CityBuilding house1 = new House( 6, 7, 5, "burak", "green", 3 );
System.out.println( "\nHouse created\n" );
System.out.println( house1 );
System.out.println( "----------------------------\n" );

CityBuilding market1 = new Market( 0, 8, 20, "james" , "08:00" , "21:00" );
System.out.println( "\nMarket created\n" );
System.out.println( market1 );
System.out.println( "----------------------------\n" );

CityBuilding office1 = new Office( 10, 12, 20, "rachel", "consulting" );
System.out.println( "\nOffice created\n" );
System.out.println( office1 );
System.out.println( "----------------------------\n" );

CityBuilding playground1 = new Playground ( 31, 4 );
System.out.println( "\nPlayground created\n" );
System.out.println( playground1 );
System.out.println( "----------------------------\n" );
```

```
street1.addFront( house1 );
street1.addBack( market1 );
street1.addBack( office1 );
street1.addFront( playground1 );

System.out.println( "\nhouse, market, office, and playground are added to street.\n" );
System.out.println( street1 );
System.out.println( "--------------------------\n" );

street1.deleteFront( house1 );
street1.deleteFront( playground1 );
street1.deleteBack( 10 );
street1.deleteBack( 0 );

System.out.println( "\nAll buildings are removed from class with 4 overloaded delete methods.\n" );
System.out.println( street1 );
System.out.println( "----------------------------\n" );

street1.addFront( market1 );
street1.addBack( house1 );
street1.addBack( playground1 );
street1.addFront( office1 );

System.out.println( "\nBuildings are added to street oppositely.\n" );
System.out.println( street1 );
System.out.printf( "Total remaining lands = %d\n", street1.totalRemainingLand() );
System.out.printf( "Total number of Buildings = %d\n", street1.getNumOfBuilding() );
street1.ratioOfPlaygrounds();
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
        , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );
System.out.println( "--------------------------\n" );

street1.printSkyline();
```

```
    street1.addFront( new House( 36, 10 , 10,  "olivia", "yellow", 3 ) );
    street1.addFront( new House( 48, 7, 10, "ahmet", "gray", 2 ));
    street1.addBack( new House( 17, 13 , 30, "elif", "black", 3  ) );
    street1.addBack( new Market( 43, 8 , 20, "george", "06:30", "19:00" ) );
    System.out.println( "\nNew buildings are added.\n" );

    System.out.println( street1 );
    System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
                , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );

    street1.printSkyline();

    System.out.println( "Testing focus() method for every building. Called from CityBuilding array." );

    for ( int i = 0; i < street1.getNumOfBuildingFront(); ++i )
        System.out.printf( "focus() returned = %s\n", street1.atFront(i).focus() );

    for ( int i = 0; i < street1.getNumOfBuildingBack(); ++i )
        System.out.printf( "focus() returned = %s\n", street1.atBack(i).focus() );

    System.out.println( "--------------------------\n" );

    System.out.println( "\nTesting clone() method." );

    LinkedListStreet street2 = ( LinkedListStreet ) street1.clone();

    System.out.println( "Original street\n"  + street1 );
    System.out.println( "Cloned street\n" + street2 );

    System.out.println( "--------------------------\n" );

    System.out.printf( "\nTesting equals() method\nresult of street1.equals(street2) is = " );

    System.out.println( street1.equals(street2) );
```

```
System.out.printf( "\nTesting equals() method\nresult of street1.equals(street2) is = " );

System.out.println( street1.equals(street2) );
System.out.println( "--------------------------\n" );

System.out.println( "\nDeleting one building from street2\n" );
street2.deleteBack(playground1);
System.out.println( street2 );

System.out.println( "After deleting result of condition is: " + street1.equals(street2) );
System.out.println( "--------------------------\n" );

System.out.println( "street1.hashCode() = " + street1.hashCode() + "\n" );
```

Testing BasicArrayStreet

```
System.out.println( "___TESTING BasicArrayStreet class___\n" );

BasicArrayStreet street1 = new BasicArrayStreet( 55 );

System.out.println( "\nEmpty Street created.\n" );
System.out.printf( "Length of Street is = %d", street1.getLengthOfStreet() );

System.out.println( street1 );
System.out.println( "----------------------------\n" );

CityBuilding house1 = new House( 6, 7, 5, "burak", "green", 3 );
System.out.println( "\nHouse created\n" );
System.out.println( house1 );
System.out.println( "----------------------------\n" );

CityBuilding market1 = new Market( 0, 8, 20, "james" , "08:00" , "21:00" );
System.out.println( "\nMarket created\n" );
System.out.println( market1 );
System.out.println( "----------------------------\n" );

CityBuilding office1 = new Office( 10, 12, 20, "rachel", "consulting" );
System.out.println( "\nOffice created\n" );
System.out.println( office1 );
System.out.println( "----------------------------\n" );

CityBuilding playground1 = new Playground ( 31, 4 );
System.out.println( "\nPlayground created\n" );
System.out.println( playground1 );
System.out.println( "----------------------------\n" );
```

```
street1.addFront( house1 );
street1.addBack( market1 );
street1.addBack( office1 );
street1.addFront( playground1 );

System.out.println( "\nhouse, market, office, and playground are added to street.\n" );
System.out.println( street1 );
System.out.println( "---------------------------\n" );

street1.deleteFront( house1 );
street1.deleteFront( playground1 );
street1.deleteBack( 10 );
street1.deleteBack( 0 );

System.out.println( "\nAll buildings are removed from class with 4 overloaded delete methods.\n" );
System.out.println( street1 );
System.out.println( "---------------------------\n" );

street1.addFront( market1 );
street1.addBack( house1 );
street1.addBack( playground1 );
street1.addFront( office1 );

System.out.println( "\nBuildings are added to street oppositely.\n" );
System.out.println( street1 );
System.out.printf( "Total remaining lands = %d\n", street1.totalRemainingLand() );
System.out.printf( "Total number of Buildings = %d\n", street1.getNumOfBuilding() );
street1.ratioOfPlaygrounds();
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
          , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );
System.out.println( "---------------------------\n" );

street1.printSkyline();
```

```
street1.addFront( new House( 36, 10 , 10,  "olivia", "yellow", 3 ) );
street1.addFront( new House( 48, 7, 10, "ahmet", "gray", 2 ));
street1.addBack( new House( 17, 13 , 30, "elif", "black", 3  ) );
street1.addBack( new Market( 43, 8 , 20, "george", "06:30", "19:00" ) );
System.out.println( "\nNew buildings are added.\n" );

System.out.println( street1 );
System.out.printf("\nTotal length of Market(s) = %d\nTotal length of House(s) = %d\nTotal length of Office(s) = %d\n"
          , street1.lengthOfMarkets(), street1.lengthOfHouses(), street1.lengthOfOffices() );

street1.printSkyline();

System.out.println( "Testing focus() method for every building. Called from CityBuilding array." );

for ( int i = 0; i < street1.getNumOfBuildingFront(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atFront(i).focus() );

for ( int i = 0; i < street1.getNumOfBuildingBack(); ++i )
    System.out.printf( "focus() returned = %s\n", street1.atBack(i).focus() );

System.out.println( "---------------------------\n" );

System.out.println( "\nTesting clone() method." );

BasicArrayStreet street2 = ( BasicArrayStreet ) street1.clone();

System.out.println( "Original street\n"  + street1 );
System.out.println( "Cloned street\n" + street2 );

System.out.println( "---------------------------\n" );
```

```
System.out.printf( "\nTesting equals() method\nresult of street1.equals(street2) is = " );

System.out.println( street1.equals(street2) );
System.out.println( "---------------------------\n" );

System.out.println( "\nDeleting one building from street2\n" );
street2.deleteBack(playground1);
System.out.println( street2 );

System.out.println( "After deleting result of condition is: " + street1.equals(street2) );
System.out.println( "---------------------------\n" );

System.out.println( "street1.hashCode() = " + street1.hashCode() + "\n" );
```

Testing House

```
CityBuilding house1 = new House ( 3, 15, 20, "burak", "red", 4 );
CityBuilding house2 = new House ( );
System.out.println( "\nhouse1 and house2 created. house2 is created with no parameter constructor." );
System.out.println( "\nhouse1 and house2 is declared as CityBuilding class." );
System.out.println( "\nhouse1 = " + house1 + "\nhouse2 = " + house2 );

System.out.printf( "\nCalling CityBuilding class getters = %d %d %d", house1.getPosition(),
            house1.getLength(), house1.getHeight() );

System.out.printf( "\nChecking equality with equals method.\nhouse1.equals(house2) returns = %s\n",
    house1.equals(house2) );

House house3 = ( House ) house1.clone();
System.out.printf( "\nhouse3 created with using clone() method. And the result is downcasted to House class.\nhouse3 = %s\n",house3 );
System.out.printf( "house1.equals(house3) returns = %s\n", house1.equals(house3) );

house3.setNumOfRoom( 7 );
house3.setColor("black");
house3.setOwner("elma");
System.out.println( "house3 is changed with its accessors." );
System.out.printf("House classes getters returns = %s %s %d\n" , house3.getOwner(), house3.getColor(), house3.getNumOfRoom() );

System.out.printf( "house1.focus() returns(called from CityBuilding class) = %s\n", house1.focus() );
System.out.println( "house3.hashCode() = " + house3.hashCode() + "\n" );
```

Testing Market

```
CityBuilding market1 = new Market ( 21, 10, 17, "john", "08:00", "21:00" );
CityBuilding market2 = new Market ( );
System.out.println( "\nmarket1 and market2 created. market2 is created with no parameter constructor." );
System.out.println( "\nmarket1 and market2 is declared as CityBuilding class." );
System.out.println( "\nmarket1 = " + market1 + "\nhouse2 = " + market2 );

System.out.printf( "\nCalling CityBuilding class getters = %d %d %d", market1.getPosition(),
            market1.getLength(), market1.getHeight() );

System.out.printf( "\nChecking equality with equals method.\nmarket1.equals(market2) returns = %s\n",
    market1.equals(market2) );

Market market3 = ( Market ) market1.clone();
System.out.printf( "\nmarket3 created with using clone() method. And the result is downcasted to Market class.\nmarket3 = %s\n"
            , market3 );
System.out.printf( "market1.equals(market3) returns = %s\n", market1.equals(market3) );

market3.setOpeningTime( "13:00" );
market3.setClosingTime("23:30");
market3.setOwner("elma");
System.out.println( "market3 is changed with its accessors." );
System.out.printf("Market classes getters returns = %s %s %s\n"
     , market3.getOwner(), market3.getOpeningTime(), market3.getClosingTime() );

System.out.printf( "market1.focus() returns(called from CityBuilding class) = %s\n", market1.focus() );
System.out.println( "market3.hashCode() = " + market3.hashCode() + "\n" );
```

Testing Office

```
CityBuilding office1 = new Office ( 19, 10, 15, "alice", "sales" );
CityBuilding office2 = new Office ( );
System.out.println( "\noffice1 and office2 created. office2 is created with no parameter constructor." );
System.out.println( "\noffice1 and office2 is declared as CityBuilding class." );
System.out.println( "\noffice1 = " + office1 + "\nhouse2 = " + office2 );

System.out.printf( "\nCalling CityBuilding class getters = %d %d %d", office1.getPosition(),
            office1.getLength(), office1.getHeight() );

System.out.printf( "\nChecking equality with equals method.\noffice1.equals(office2) returns = %s\n",
    office1.equals(office2) );

Office office3 = ( Office ) office1.clone();
System.out.printf( "\noffice3 created with using clone() method. And the result is downcasted to Office class.\noffice3 = %s\n"
    , office3 );
System.out.printf( "office1.equals(office3) returns = %s\n", office1.equals(office3) );

office3.setJobType( "medical" );
office3.setOwner("elma");
System.out.println( "office3 is changed with its accessors." );
System.out.printf("Office classes getters returns = %s %s\n"
     , office3.getOwner(), office3.getJobType() );

System.out.printf( "office1.focus() returns(called from CityBuilding class) = %s\n", office1.focus() );
System.out.println( "office3.hashCode() = " + office3.hashCode() + "\n" );
```

Testing Playground

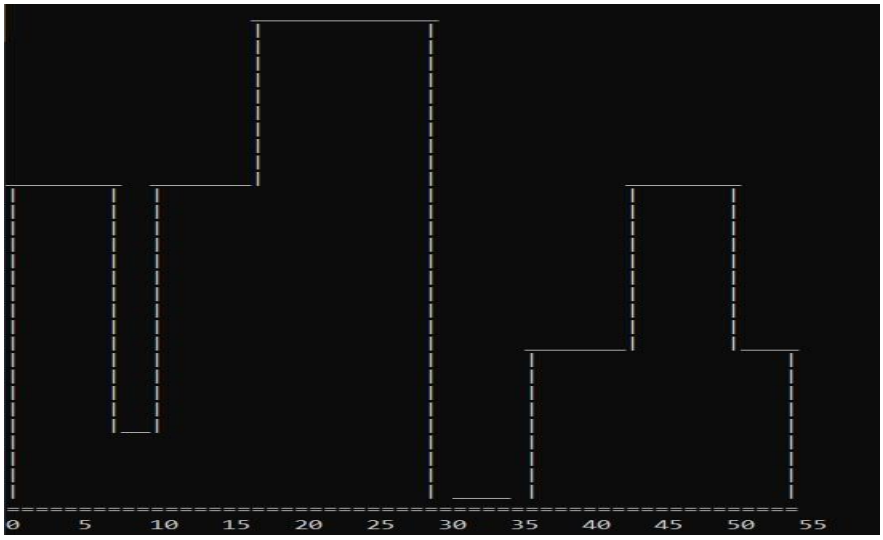
```
CityBuilding playground1 = new Playground ( 22, 11 );
CityBuilding playground2 = new Playground ( );
System.out.println( "\nplayground1 and playground2 created. playground2 is created with no parameter constructor." );
System.out.println( "\nplayground1 and playground2 is declared as CityBuilding class." );
System.out.println( "\nplayground1 = " + playground1 + "\nhouse2 = " + playground2 );

System.out.printf( "\nCalling CityBuilding class getters = %d %d %d\n", playground1.getPosition(),
           playground1.getLength(), playground1.getHeight() );

System.out.printf( "\nChecking equality with equals method.\nplayground1.equals(playground2) returns = %s\n",
    playground1.equals(playground2) );

Playground playground3 = ( Playground ) playground1.clone();
System.out.printf( "\nplayground3 created with using clone() method. And the result is downcasted to Playground class.\n" );
System.out.printf("playground3 = %s\n", playground3);
System.out.printf( "playground1.equals(playground3) returns = %s\n", playground1.equals(playground3) );

System.out.printf( "playground1.focus() returns(called from CityBuilding class) = %s\n", playground1.focus() );
System.out.println( "playground3.hashCode() = " + playground3.hashCode() + "\n" );
```

6. RUNNING AND RESULTS

Sample Street

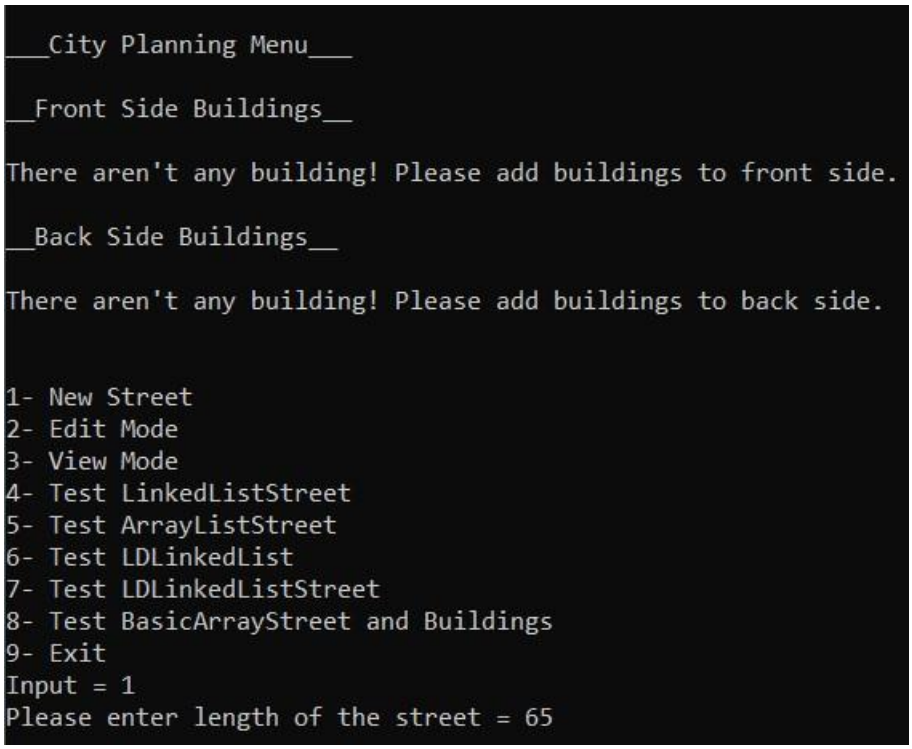
```
__Front Side Buildings__

1-  Market     0   8   20  james      08:00      21:00
2-  Office     10  12  20  rachel     consulting
3-  House      36  10  10  olivia     yellow     3
4-  House      48  7   10  ahmet      gray       2

__Back Side Buildings__

5-  House      6   7   5   burak      green      3
6-  Playground 31  4   0
7-  House      17  13  30  elif       black      3
8-  Market     43  8   20  george     06:30      19:00
```

This street's Skyline Silhouette

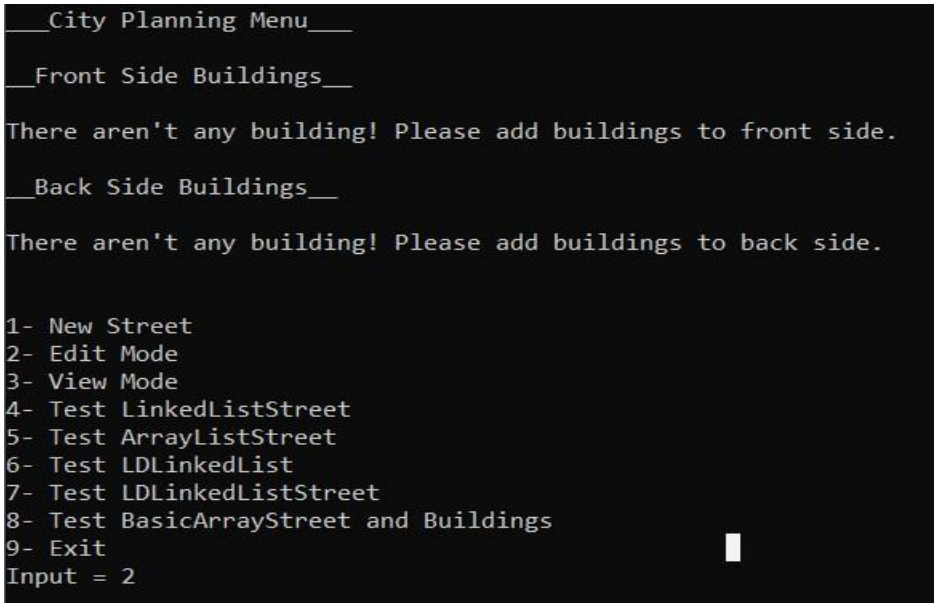


All classes Test's can be run inside this menu, It also has edit and view modes.

Creating a new street



Opening edit mode of street



Add house to street

```
1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 1

Which side you want to add building?
1- Front Side
2- Back Side
Input = 1

What kind of a building you want to add?
1- House
2- Market
3- Office
4- Playground
5- Exit
Input = 1

Please enter the position of the building(integer) = 3

Please enter the length of the building(integer) = 10

Please enter the height of the building(integer) = 15

Please enter the owner of the house = burak

Please enter the color of the house = green

Please enter the total number of room of the house(integer) = 3
```

Add market to street

```
___Street Editing Mode___


__Front Side Buildings__

1-  House      3   10  15  burak      green      3

__Back Side Buildings__

There aren't any building! Please add buildings to back side.


1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 1

Which side you want to add building?
1- Front Side
2- Back Side
Input = 1

What kind of a building you want to add?
1- House
2- Market
3- Office
4- Playground
5- Exit
Input = 2

Please enter the position of the building(integer) = 20

Please enter the length of the building(integer) = 10

Please enter the height of the building(integer) = 20

Please enter the owner of the market = elaine

Please enter the opening time of the market(Ex: 08:00) = 09:00

Please enter the closing time of the market(Ex: 18:00) = 19:00
```

Add office to street

```
1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 1

Which side you want to add building?
1- Front Side
2- Back Side
Input = 2

What kind of a building you want to add?
1- House
2- Market
3- Office
4- Playground
5- Exit
Input = 3

Please enter the position of the building(integer) = 0

Please enter the length of the building(integer) = 5

Please enter the height of the building(integer) = 17

Please enter the owner of the office = john

Please enter the job type of the office = medical
```

Add playground to street

```
___Street Editing Mode___


__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical


1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 1

Which side you want to add building?
1- Front Side
2- Back Side
Input = 2

What kind of a building you want to add?
1- House
2- Market
3- Office
4- Playground
5- Exit
Input = 4

Please enter the position of the building(integer) = 14

Please enter the length of the building(integer) = 3
```

Delete one building from street

```
1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 2

__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
4-  Playground 14  3   0

Which building you want to delete(enter building number) = 4

___Street Editing Mode___


__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
```

Add playground again

```
1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 1

Which side you want to add building?
1- Front Side
2- Back Side
Input = 2

What kind of a building you want to add?
1- House
2- Market
3- Office
4- Playground
5- Exit
Input = 4

Please enter the position of the building(integer) = 35

Please enter the length of the building(integer) = 5

___Street Editing Mode___


__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
4-  Playground 35  5   0


1- Add Building
2- Delete Building
3- Exit from Editing Mode
Input = 3
```

Open the view mode and display number of remaining lands

```
___Street Viewing Mode___
1- Display the total remaining length of lands on the street
2- Display the list of buildings on the street
3- Display the number and ratio of length of playgrounds in the street.
4- Calculate the total length of street occupied by the markets, houses or offices.
5- Display the skyline silhouette of the street
6- Focus on a spesific building( test polimorphism )
7- Exit from Viewing Mode
Input = 1

Total remaining Lands = 102
```

Display the list of buildings

```
Input = 2

__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
4-  Playground 35  5   0
```

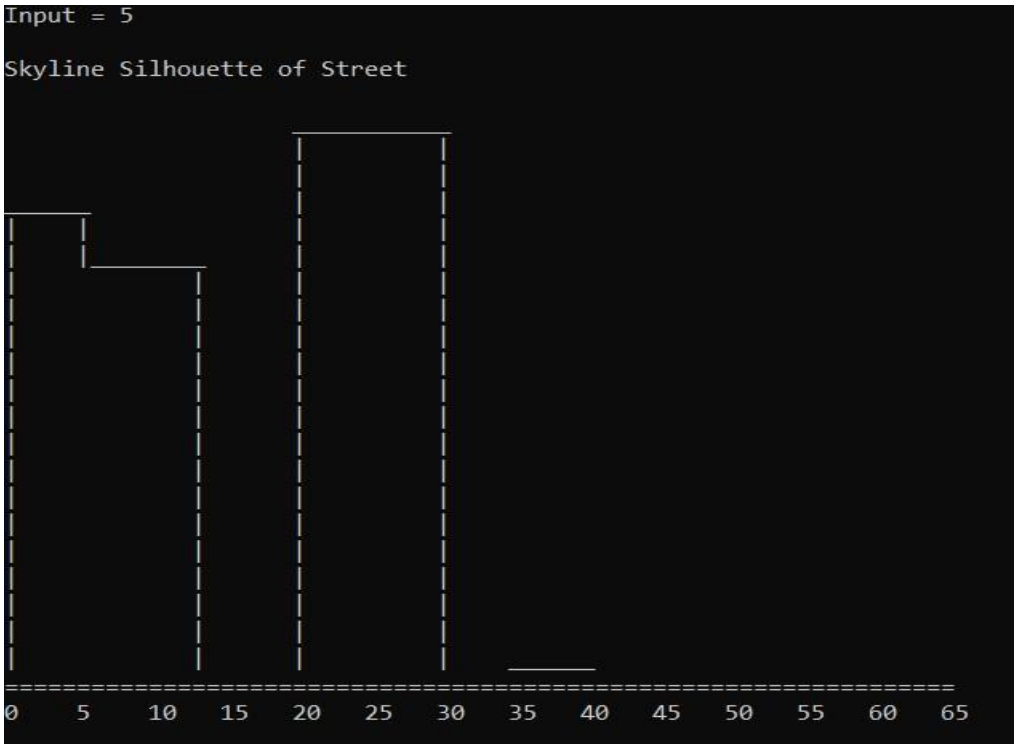
Display the number and ratio of playgrounds

```
Input = 3

Number of playgrounds = 1
Ratio of playgrounds = % 7.692
```

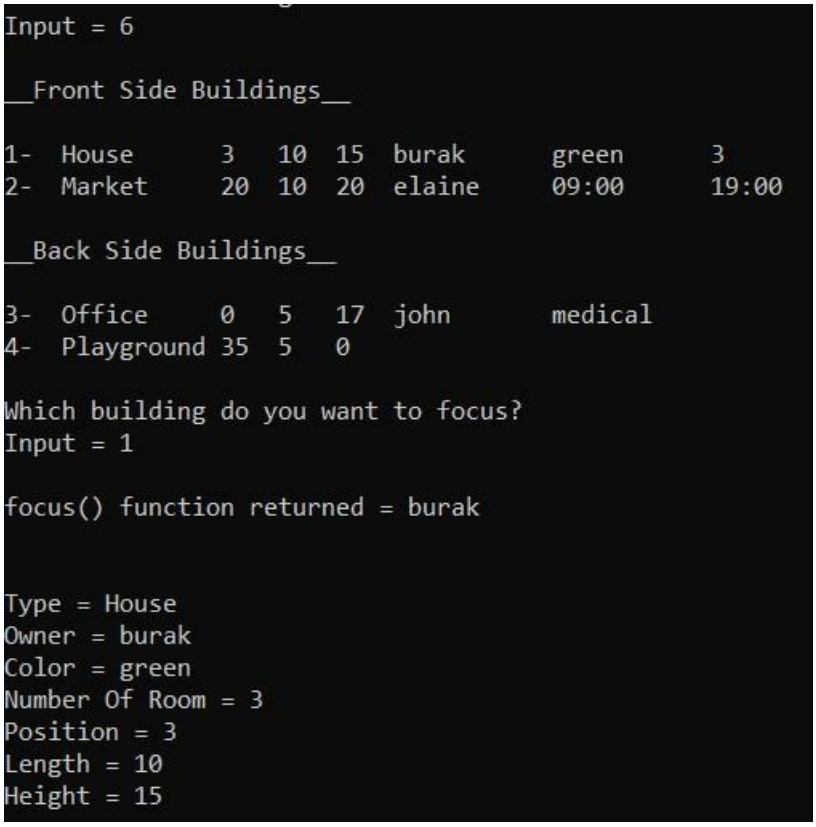
After choosing 4th option

```
Input = 4

Total length of Market(s) = 10
Total length of House(s) = 10
Total length of Office(s) = 5
```

Display the skyline silhouette



Focus a spesific building



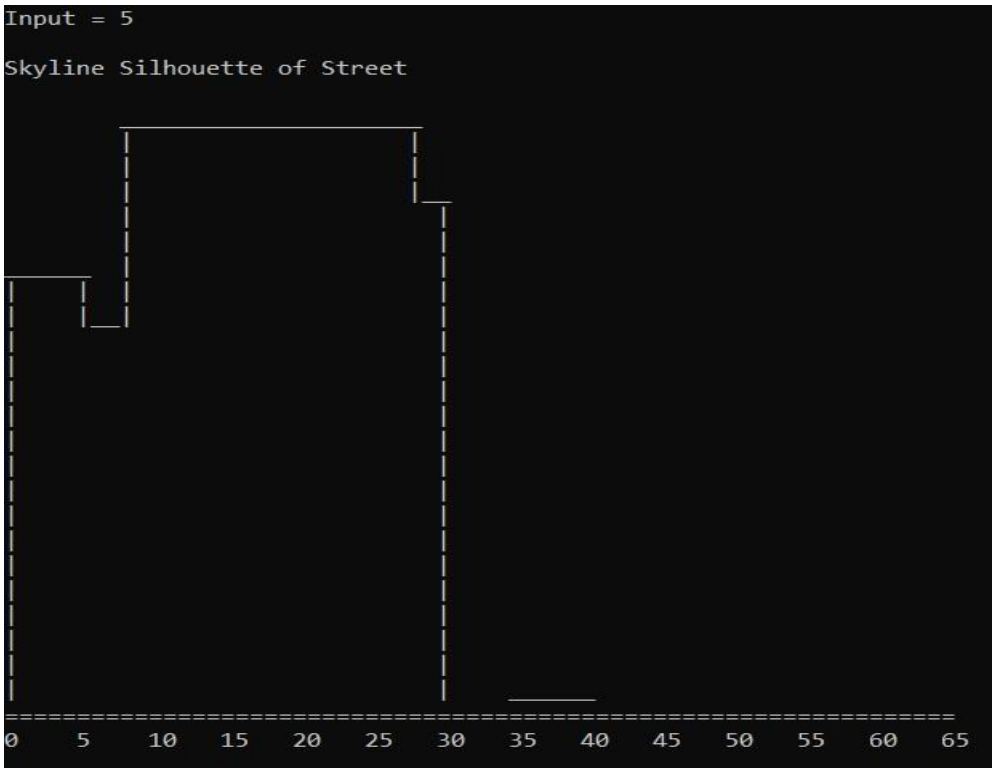
Focus another building

```
Input = 6

__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
4-  Playground 35  5   0

Which building do you want to focus?
Input = 3

focus() function returned = medical


Type = Office
Job Type = medical
Owner = john
Position = 0
Length = 5
Height = 17
```

After adding a house

```
__Front Side Buildings__

1-  House      3   10  15  burak      green      3
2-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

3-  Office     0   5   17  john       medical
4-  Playground 35  5   0
5-  House      8   20  23  laura      red        4
```

Silhouette after addition



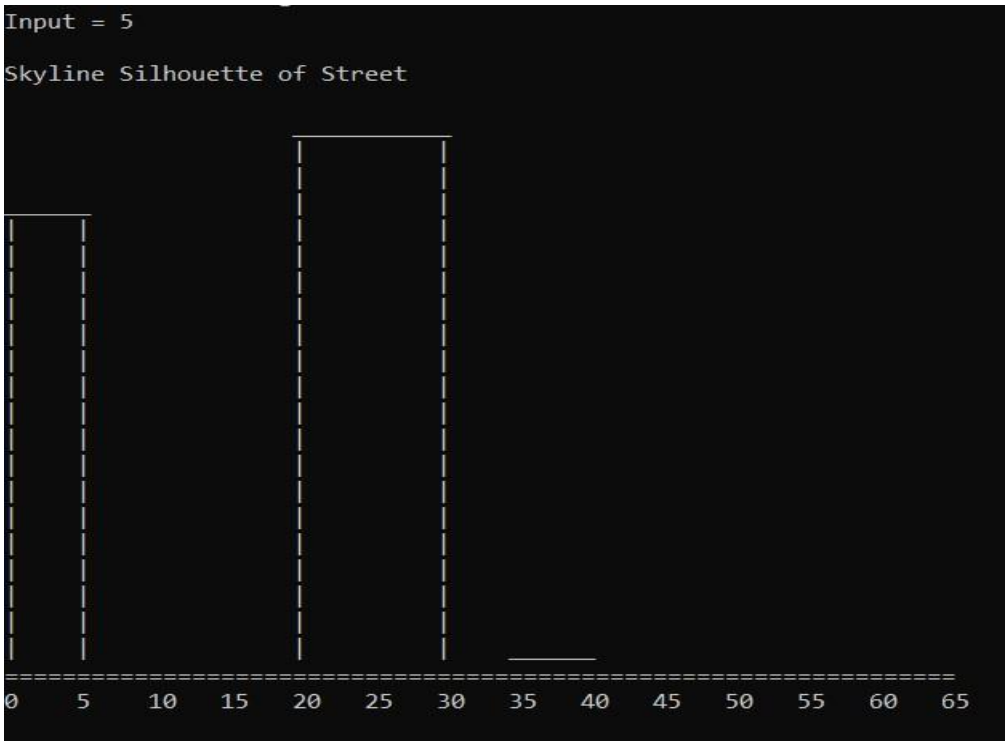
Delete one building

```
__Front Side Buildings__

1-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

2-  Office     0   5   17  john       medical
3-  Playground 35  5   0
4-  House      8   20  23  laura      red        4
```

Delete another building

```
__Front Side Buildings__

1-  Market     20  10  20  elaine     09:00      19:00

__Back Side Buildings__

2-  Office     0   5   17  john       medical
3-  Playground 35  5   0
```

Silhouette after deleting



Test Case results are inside test.txt, and these cases are run before menu shows up. And you can run them again inside menu. “make” command compiles, and “make run” command runs the program.