GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework 1

YUSUF ARSLAN

200104004112

1. System Requirement

The system should have Java virtual machine (JVM), Java development kit (JDK) and 'at least' Java runtime environment – 11 (JRE 11).

2. Class Diagrams

Class diagrams attached to the and of the file.

3. Problem Solutions Approach

What is the problem?

We are asked to design and and implement a city planning software that will be used for designing a small street town.
Street consists of two rows, equal length, and every row keeps constructions.

Details:

There will be 4 type construction in the street, they are:
House, office, market and playground.
Every constructions has some properties beside the length and height.
House, represent with, room number, color and owner,
Office, job-type (business) and owner,
Market, opening and closing time and owner.
Playground is a simple construction and it has default height.

So, in program there will be 2 modes,

1- Editing mode, which allows to modify street. (add and remove buildings to the rows)
2- Viewing mode, which allows to display street and its properties.

What is my solution approach?

First of all, I brought House, Office, Market and Playground together under the roof of '`Building`' class.
I created a 'is a' relationship, and I extended classes for every construction from 'Building'.
They are, `House`, `Office`, `Market` and `Playground` classes.

All this sub-classes, besides the represent the super class <`Building`> they also has specific properties inside them.

Then,

I created 2 other class,
To keep rows and buildings on the rows: `Street`
To keep `Building` references and position: `BuildingOnStreet`

Although the `BuildingOnStreet` seems unnecessary at first, since I don't want the building to hold position as a field, I created the `BuildingOnStreet` that will act as some kind of container.

`Street` class basically is also a container class that has a `length` and two reference array of `BuildingOnStreet` which they are correspond to rows. (`BuildingOnStreet[] row1`, `BuildingOnStreet[] row2`)

Finally,

The user, will be able to create buildings, add position where it will be added to the street, and then add the building to the street if it is possible.

Exceptions and error handlings

To street make sense, it should be at least 20 meter long, and since we will display street to the terminal, it should be maximum 150 meter long.

Since a character in the terminal has x for width and 2x for height, due to scaling problem the height should be a 'even' number.

To buildings make sense with real life, house and office length should be between 4 and 40, and height between 6 and 60,
market length should be between 4 – 80 and height 4 – 12,
playground length should be between 4 – 120 and height should be default 2 meter.

Room number of building should be at least 1, and time format for the market should be [hh:mm].
Position should be at least 0 and smaller then length of the street.

All the exceptions handled properly.

Here the code where I add a building to the street:

```
public boolean addBuilding(BuildingOnStreet[] row,
                           BuildingOnStreet newBOnStreet)
{
      checkAddBuildingValidity(row, newBOnStreet);
      BuildingOnStreet[] temp;
      int index = 0;
      if (row != null){
            temp = new BuildingOnStreet[row.length + 1];
            for (int i = 0; i < row.length; i++){
                  temp[i] = row[i];
            }
            index = row.length;
      }
      else
            temp = new BuildingOnStreet[1];

      temp[index] = newBOnStreet;

      if      (row == row1)   setRow1(temp);
      else if (row == row2)   setRow2(temp);

      setSilhouette();
      return true;
}
```

`addBuilding` takes building as parameter and firstly checks validity of building:

1- If street length is not setted
2- If the building length is not setted
3- Is current building is exist in the any row
(Does not allow a add building to the street twice)
4- If the position that user wants to enter not available for setting building
Does not allow to set the building.

Otherwise, according to the row that user wants to add building, expands row dynamically and adds the new item to the row (building array).

If user want remove a building from street, just shrinks the row.

4. Test Cases

1- Create a `Street`
2- Create Building based objects (`House`, `Office`, `Market`, `Playground`)
3- Create `BuildingOnStreet` objects from buildings and their position on street
4- Add `BuildingOnStreet` objects to the first and second row of street
5- Display silhouette of street
6- Remove building from street
7- Find total remaining land in the street
8- Find number of playgrounds in the street
9- Calculate ratio of playgrounds to the total land
10- Occupied land by house, office and markets

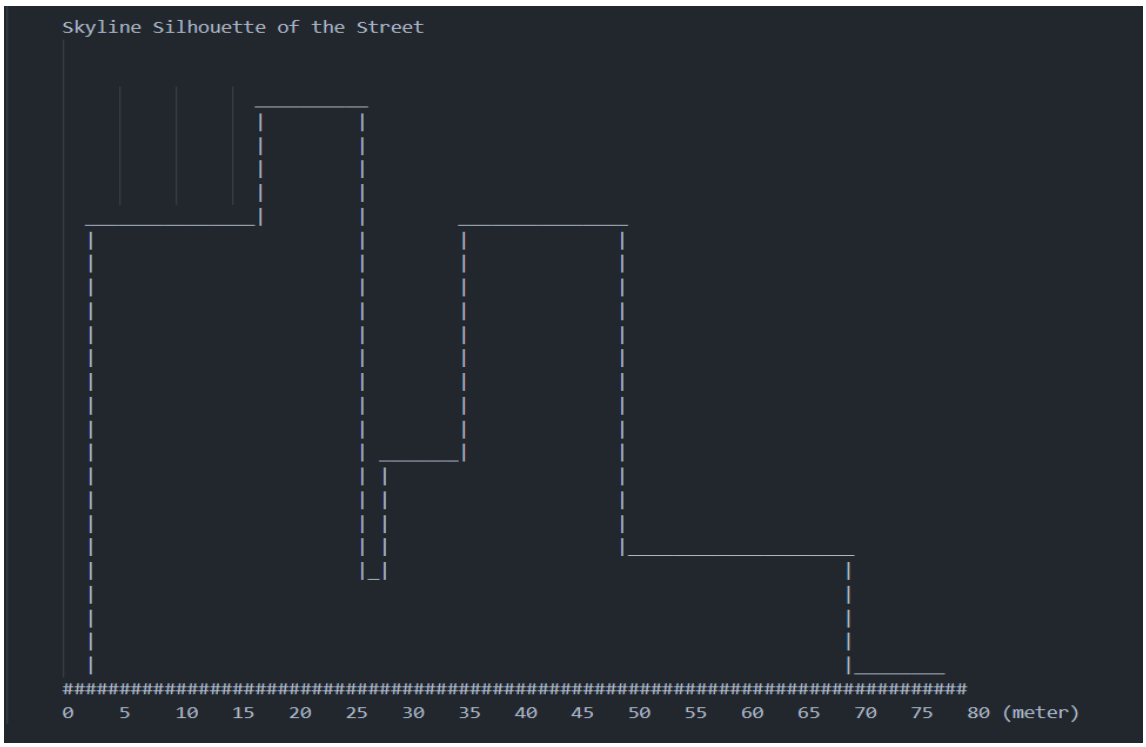
5. Running Command and Results

Result of driver (without user interaction)

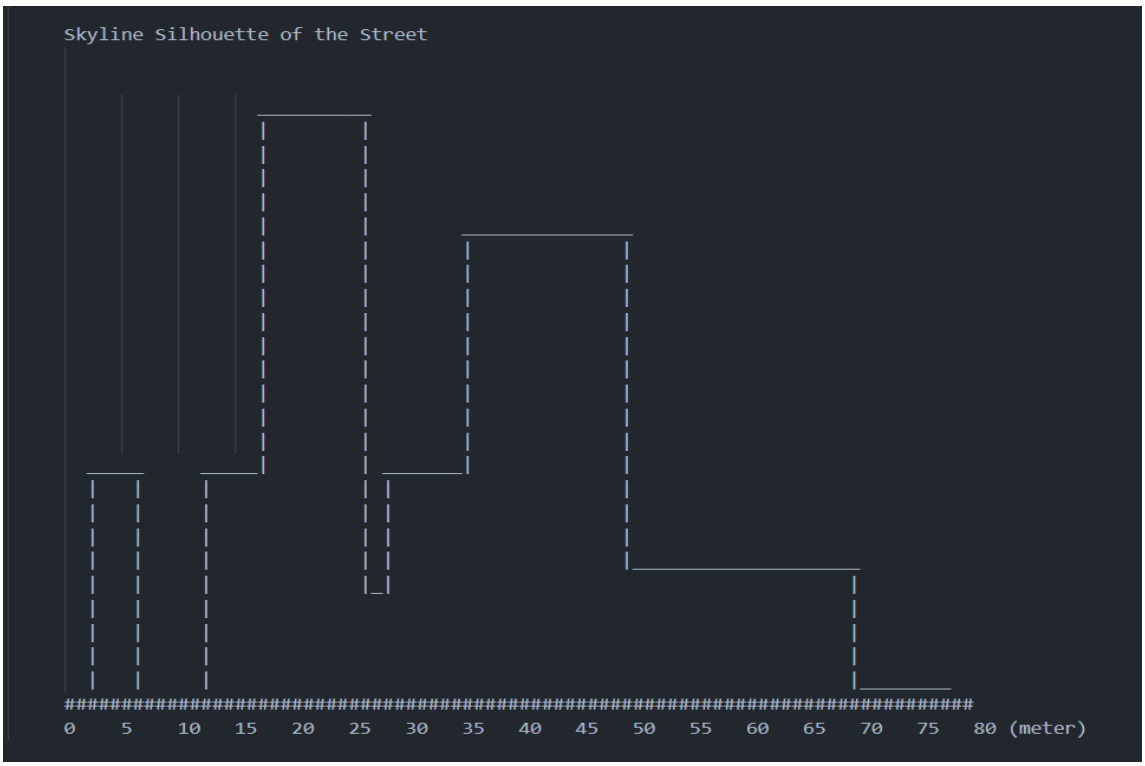
1- Creating building objects (to add them to the street):

```
Building 3/Market

     ┌─────────┐
     │         │
     │         │
     │         │
     │         │
     │         │
■■■■■■■■■■■■■■■■■■■■■

Length: 10
Height: 10
Owner: Lilly Aldrin
Opening Time: 10:00

CLOSING TIME: 21:00
```

2- Display skyline silhouette of the street



3- Display skyline silhouette after remove a building



4- Display properties of street after modifying street

```
Total remaining land (meter): 49
Number of playgrounds: 1
Ratio of playgrounds: 0.1625
Occupied land by markets: 35
Occupied land by houses: 30
Occupied land by offices: 20
```

Result of driver (with user interaction)

1- Enter street length

```
Enter street length [20, 150]
60
```

2- Select the mode

```
Enter which mode do you want to use:
1- Editing mode
2- Viewing mode
(0 to exit)
1
```

3- Select operation that you want to do

```
Editing Mode - Menu

1- Add building to first row
2- Add building to second row
3- Remove building from first row
4- Remove building from second row
5- Silhouette (Demo)
(0 to exit)

1
```

4- Add building

```
Select the building type you want to add.

1- House
2- Office
3- Market
4- Playground
(0 to exit)

2
Enter length [4, 40] meter
12
Enter height [4, 60 - even] meter
24
Enter business
Cleaning
Enter owner
Dwight Schrute
Enter position [>0]
2
Building successfully added to first row!
```

5- Add more building…

Building successfully added to second row!

Building successfully added to first row!

Building successfully added to first row!

Building successfully added to second row!

6- Remove a building

```
Select building you want to remove!

2

Building successfully removed!
```

7- Switch the viewing mode

```
Enter which mode do you want to use:
1- Editing mode
2- Viewing mode
(0 to exit)
2
```

8- Select operation that you want to do

```
Viewing Mode - Menu

1- Display total remaining length of lands
2- Display list of buildings
3- Display the number and ratio of length of playgrounds
4- Display total length of street occupied by the markets, houses or offices.
5- Display silhouette
(0 to exit)
```

9- Display total remaining land

```
1
Total remaining land (meter): 30
```

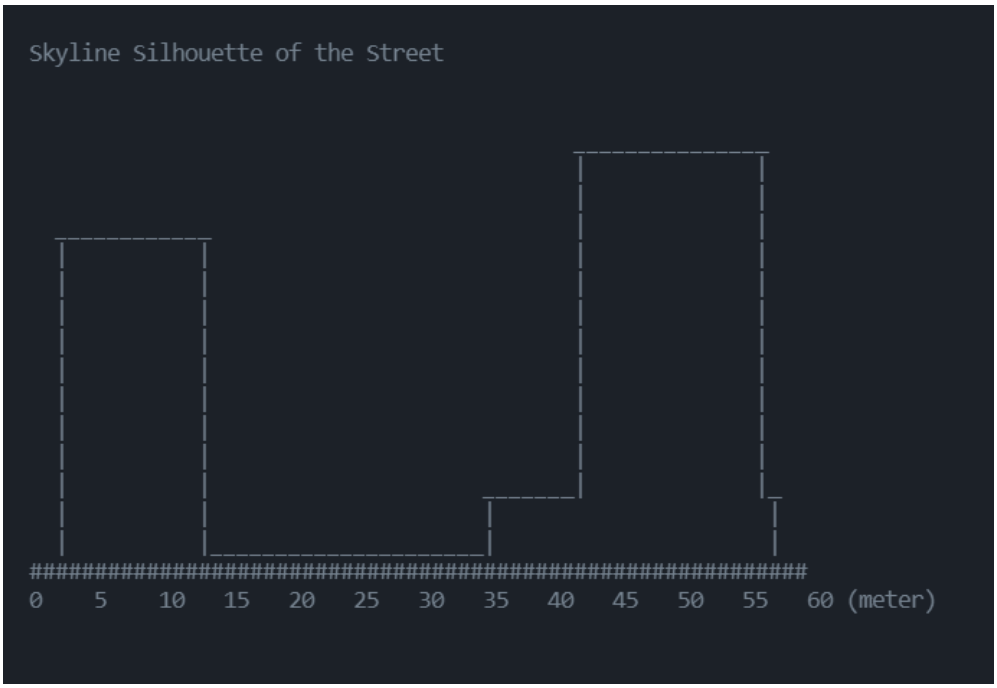
10- Display playground ratio and length

```
3
Number of playgrounds: 1
Ratio of playgrounds: 0.3333333333333333
```

11- Total land that occupied by different types of building

```
4
Occupied land by markets: 23
Occupied land by houses: 15
Occupied land by offices: 12
```

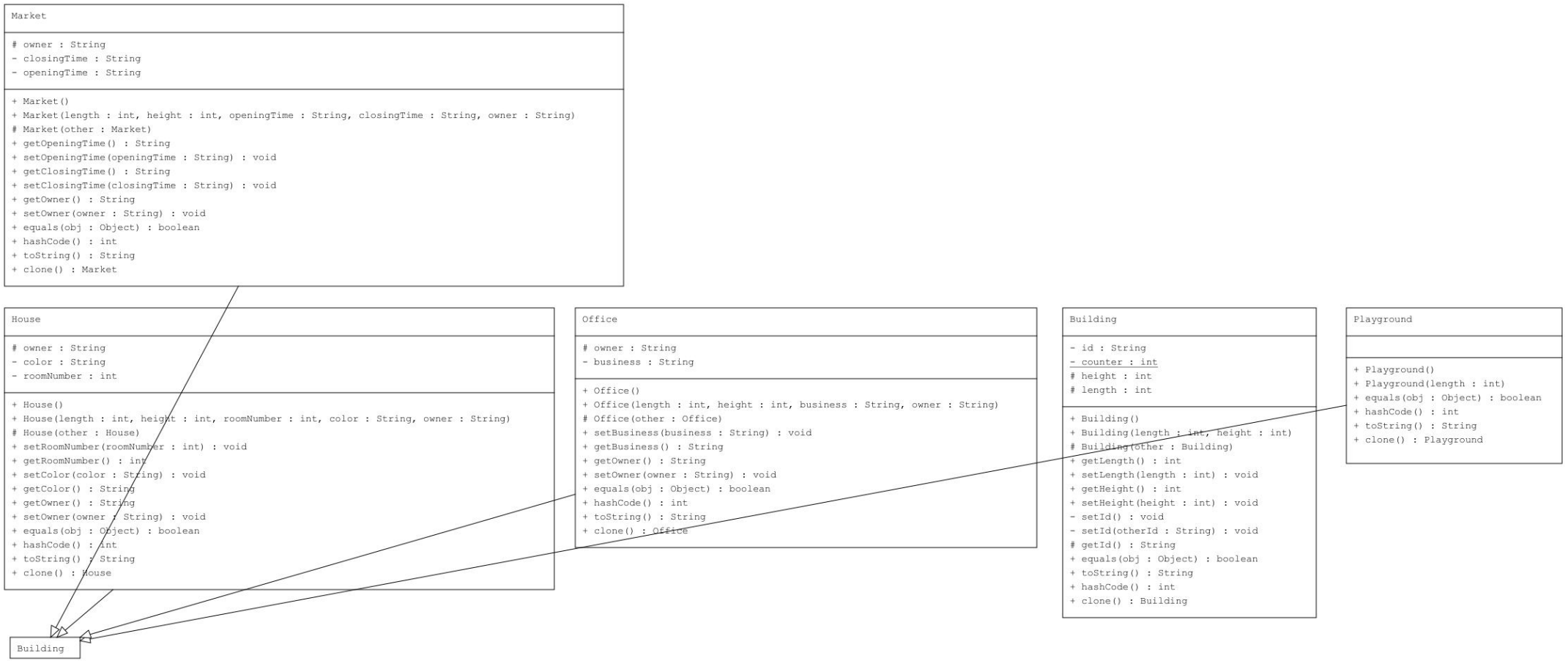
11- Display silhouette



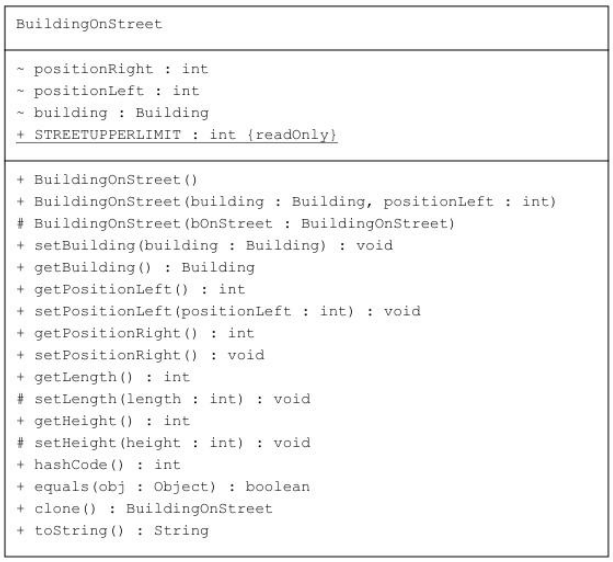
12- Display list of buildings

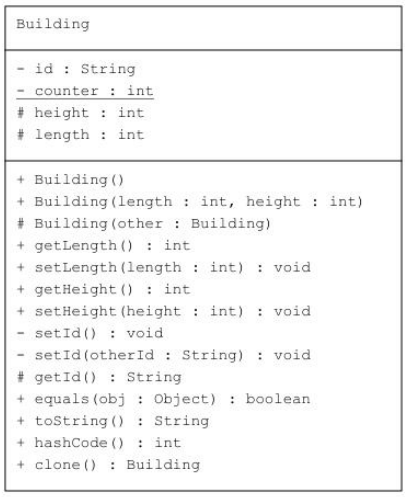
```
      List of building on the street

------------------------------

Length of street: 60

Buildings on first row:
-----------------------

Building 10/Office

     ┌──────────┐
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
     │          │
■■■■■■■■■■■■■■■■■■■■■■

Length: 12
Height: 24
Owner: Dwight Schrute

BUSINESS: Cleaning

Position: 2
Building 14/Market

     ┌──────────────────────┐
     │                      │
     │                      │
     │                      │
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

Length: 23
Height: 6
Owner: Pam
Opening Time: 16:00

CLOSING TIME: 02:00

Position: 35
```

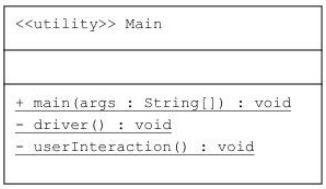
```
Buildings on second row:
------------------------

Building 12/Playground

     ┌──────────────────────────────────────┐


Length: 40
Height: 2

Position: 2
Building 13/House

     ┌──────────────┐
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │
     │              │


Length: 15
Height: 30
Room Number: 10
Color: lilac

OWNER: Kevin

Position: 42
```


Market
owner : String
- closingTime : String
- openingTime : String
+ Market()
+ Market(length : int, height : int, openingTime : String, closingTime : String, owner : String)
# Market(other : Market)
+ getOpeningTime() : String
+ setOpeningTime(openingTime : String) : void
+ getClosingTime() : String
+ setClosingTime(closingTime : String) : void
+ getOwner() : String
+ setOwner(owner : String) : void
+ equals(obj : Object) : boolean
+ hashCode() : int
+ toString() : String
+ clone() : Market
House
owner : String
- color : String
- roomNumber : int
+ House()
+ House(length : int, height : int, roomNumber : int, color : String, owner : String)
# House(other : House)
+ setRoomNumber(roomNumber : int) : void
+ getRoomNumber() : int
+ setColor(color : String) : void
+ getColor() : String
+ getOwner() : String
+ setOwner(owner : String) : void
+ equals(obj : Object) : boolean
+ hashCode() : int
+ toString() : String
+ clone() : House
Office
owner : String
- business : String
+ Office()
+ Office(length : int, height : int, business : String, owner : String)
# Office(other : Office)
+ setBusiness(business : String) : void
+ getBusiness() : String
+ getOwner() : String
+ setOwner(owner : String) : void
+ equals(obj : Object) : boolean
+ hashCode() : int
+ toString() : String
+ clone() : Office
Building
- id : String
- counter : int
height : int
length : int
+ Building()
+ Building(length : int, height : int)
# Building(other : Building)
+ getLength() : int
+ setLength(length : int) : void
+ getHeight() : int
+ setHeight(height : int) : void
- setId() : void
- setId(otherId : String) : void
getId() : String
+ equals(obj : Object) : boolean
+ toString() : String
+ hashCode() : int
+ clone() : Building
Playground
+ Playground()
+ Playground(length : int)
+ equals(obj : Object) : boolean
+ hashCode() : int
+ toString() : String
+ clone() : Playground
Building


Street
- silhouette : String
- heightsInRange : int[]
- row2 : BuildingOnStreet[]
- row1 : BuildingOnStreet[]
- length : int
+ UPPERLIMIT : int {readOnly}
+ LOWERLIMIT : int {readOnly}
+ Street()
+ Street(length : int)
+ Street(row1 : BuildingOnStreet[], row2 : BuildingOnStreet[], length : int)
# Street(other : Street)
+ getLength() : int
# setLength(length : int) : void
+ setRow1(row1 : BuildingOnStreet[]) : void
+ getRow1() : BuildingOnStreet[]
+ setRow2(row2 : BuildingOnStreet[]) : void
+ getRow2() : BuildingOnStreet[]
+ setSilhouette() : void
+ getSilhouette() : String
- setHeightsInRange() : void
- findMaxHeight() : int
+ addBuilding(row : BuildingOnStreet[], newBOnStreet : BuildingOnStreet) : boolean
+ removeBuilding(row : BuildingOnStreet[], bOnStreet : BuildingOnStreet) : boolean
- checkRemoveBuildingValidity(row : BuildingOnStreet[], bOnStreet : BuildingOnStreet) : void
- checkAddBuildingValidity(row : BuildingOnStreet[], newBOnStreet : BuildingOnStreet) : void
+ isBuildingExist(row : BuildingOnStreet[], newBOnStreet : BuildingOnStreet) : boolean
+ editingMode() : void
- scanBuilding() : BuildingOnStreet
+ viewingMode() : void
+ findTotalOccupiedLandBy(className : String) : int
+ findNumberOfPlaygrounds() : int
+ findRatioOfPlaygrounds() : double
+ findTotalRemainingLand() : int
+ hashCode() : int
+ equals(obj : Object) : boolean
+ clone() : Street
+ toString() : String


BuildingOnStreet
~ positionRight : int
~ positionLeft : int
~ building : Building
+ STREETUPPERLIMIT : int {readOnly}
+ BuildingOnStreet()
+ BuildingOnStreet(building : Building, positionLeft : int)
# BuildingOnStreet(bOnStreet : BuildingOnStreet)
+ setBuilding(building : Building) : void
+ getBuilding() : Building
+ getPositionLeft() : int
+ setPositionLeft(positionLeft : int) : void
+ getPositionRight() : int
+ setPositionRight() : void
+ getLength() : int
# setLength(length : int) : void
+ getHeight() : int
# setHeight(height : int) : void
+ hashCode() : int
+ equals(obj : Object) : boolean
+ clone() : BuildingOnStreet
+ toString() : String


Building
- id : String
- counter : int
height : int
length : int
+ Building()
+ Building(length : int, height : int)
# Building(other : Building)
+ getLength() : int
+ setLength(length : int) : void
+ getHeight() : int
+ setHeight(height : int) : void
- setId() : void
- setId(otherId : String) : void
getId() : String
+ equals(obj : Object) : boolean
+ toString() : String
+ hashCode() : int
+ clone() : Building


<<utility>> Main
+ main(args : String[]) : void
- driver() : void
- userInteraction() : void