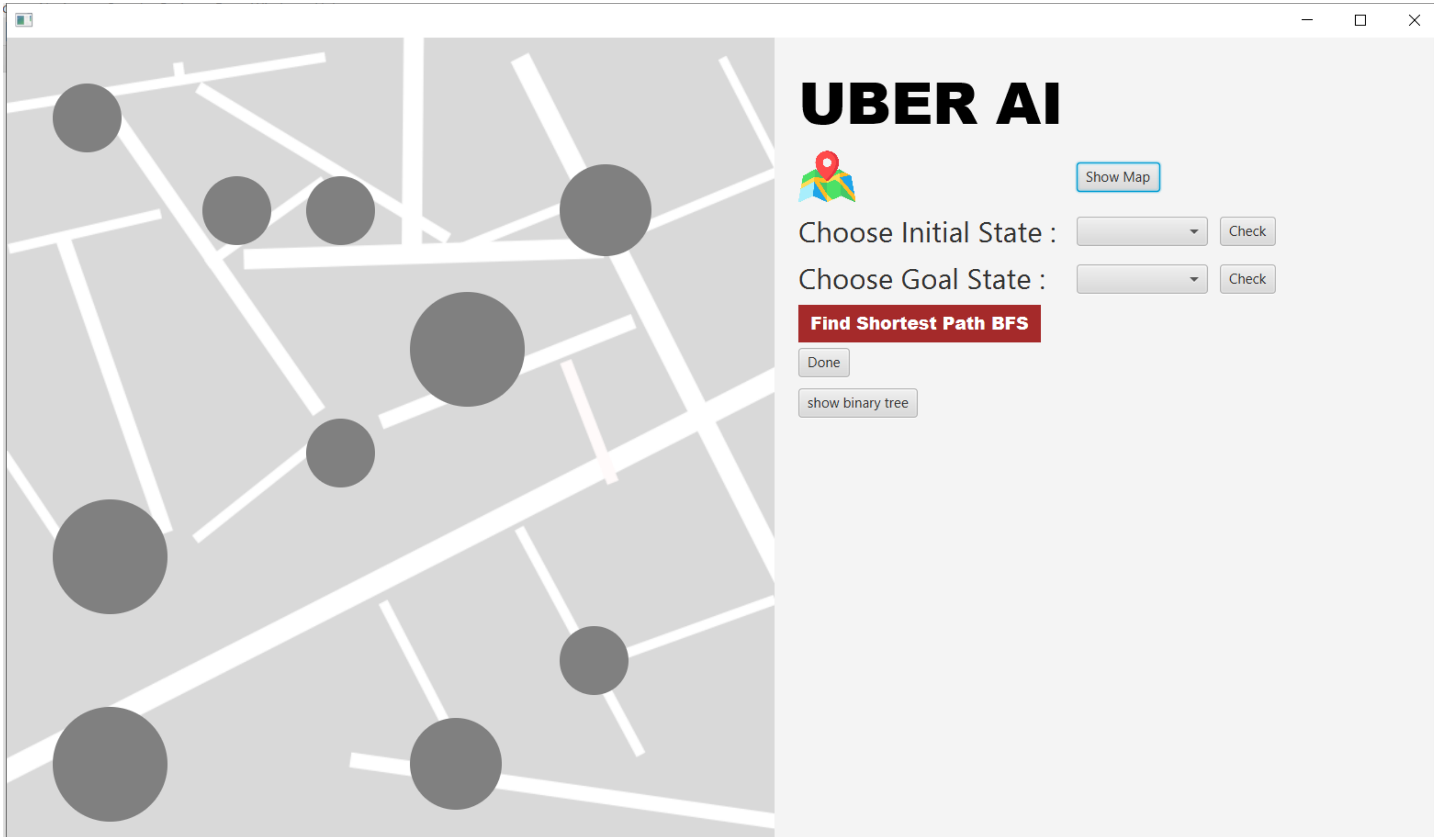
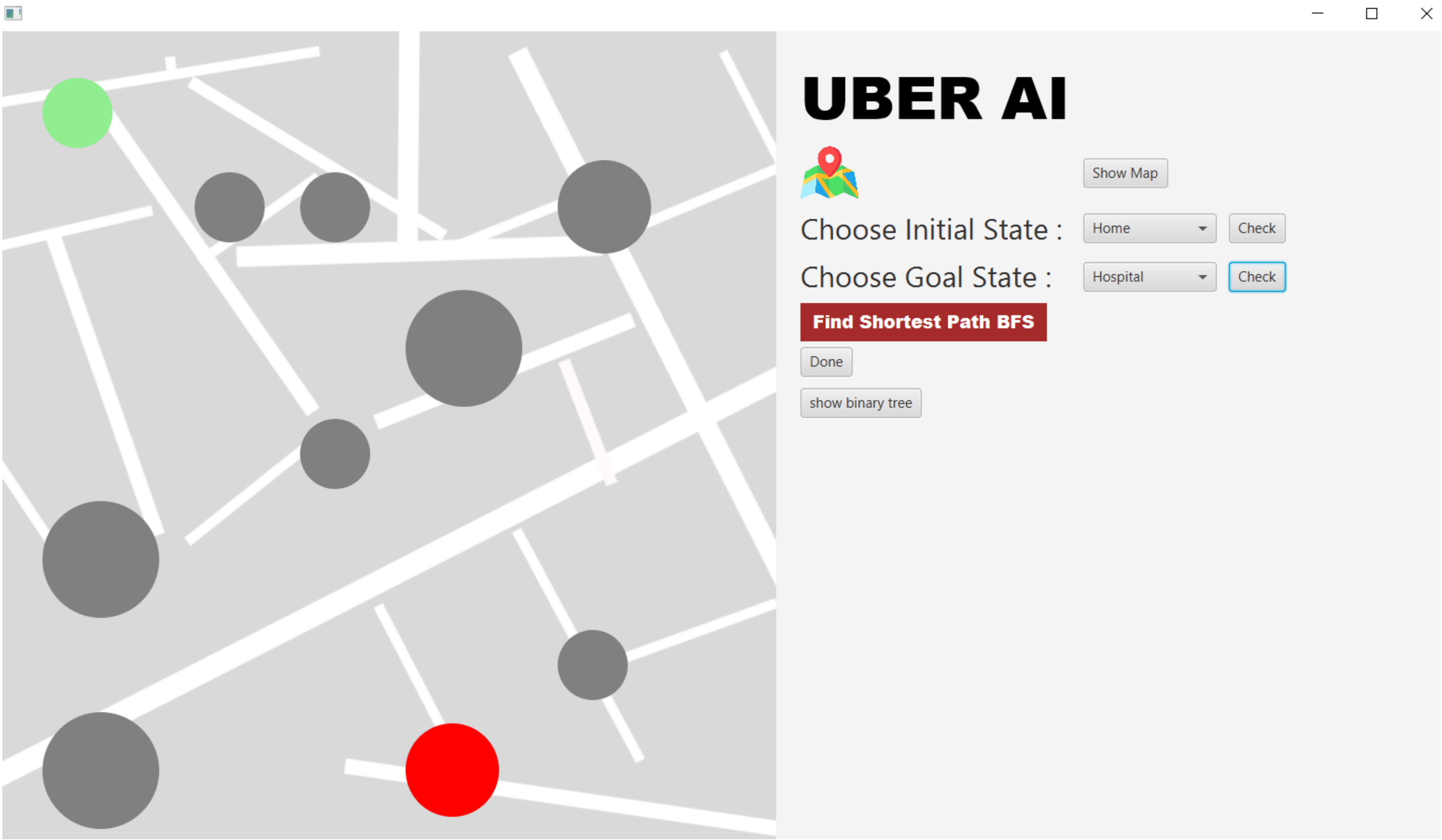
UBER AI - Report - Zeinab Ibrahim



First the user will open the map and have to select the Initial state which is the place he will start he destination from and the goal state is the place destination

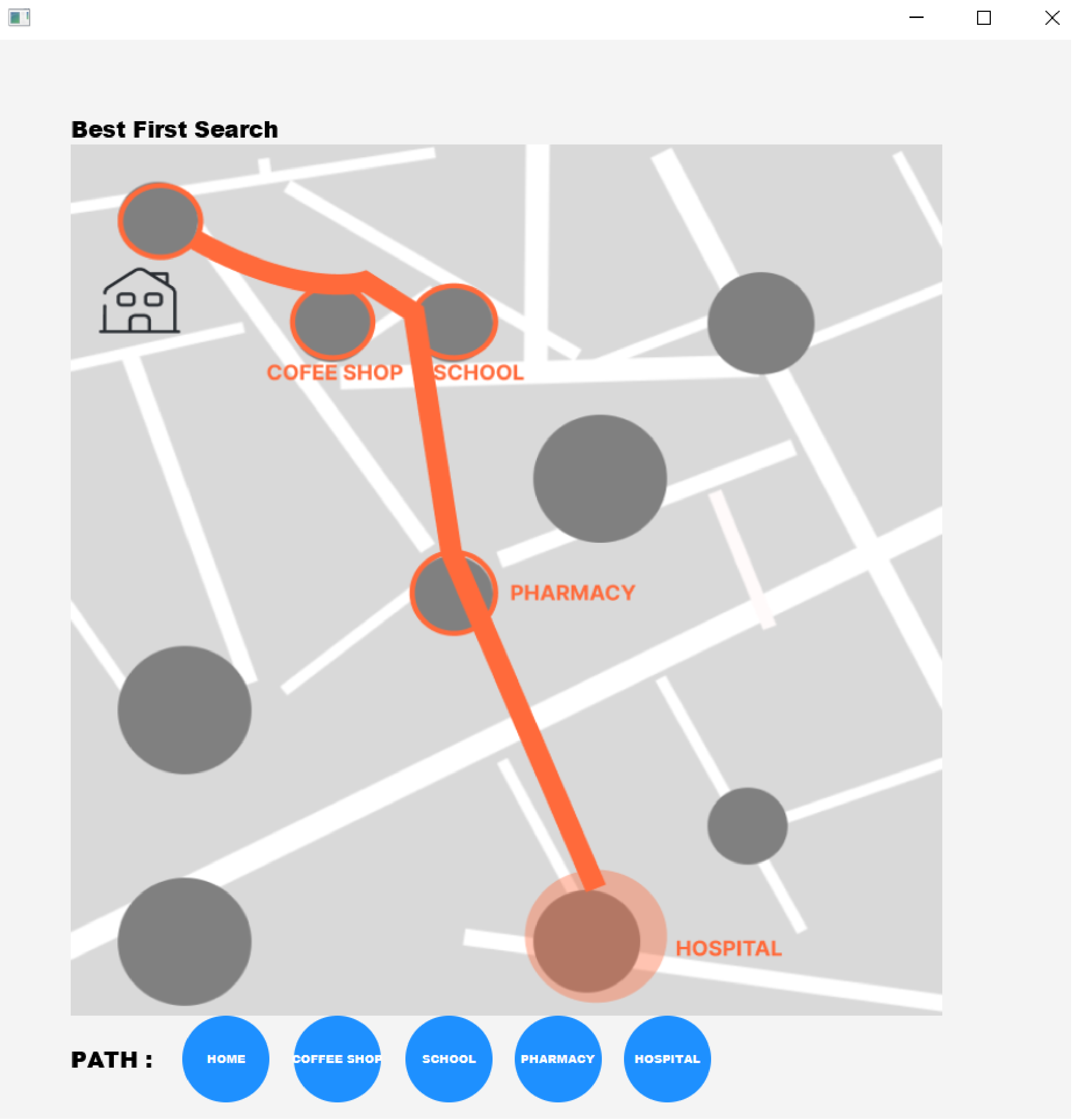


Choose Initial State : Home

Choose Goal State : Hospital

The green place is the initial state

the red place is the goal state



Best First Search

Breaking down the code

1. Initialize state for the user input using comboBox:

```
ComboBox<String> comboBox = new ComboBox<>();
comboBox.getItems().addAll(
"Airport",
"Home",
"Pharmacy",
"Restaurant",
"SuperMarket",
"University",
"Library",
"Hospital",
"CoffeeShop",
"School"
);
```

2. Get te value from the comboBox and assigned for user interactions...

```
cb2checkbtn.setOnMouseClicked(e->{
String selectedValue = comboBox2.getValue();
System.out.println("Selected Value: " + selectedValue);
if (selectedValue.equals("Home")) {
cA.setFill(Color.FIREBRICK);
...
.
.
.
}
```

3. After the user choose the BFS algorithm a boolean value bfs will be assigned to true and after clicking the done button an event is set to call Grapth class

```
bfsbtn.setOnMouseClicked(e->{
bfs = true;
System.out.println(bfs);
});



donebtn.setOnMouseClicked(e->{
if (bfs == true) {
System.out.println("true");
Graph g = new Graph(initialstate,4);
}
});
```

4.Graph class

4. 1. Initialize variable

```
private int V;
private LinkedList<Integer> adj[]; // Adjacency Lists
public LinkedList<Integer> queue
= new LinkedList<Integer>();
```

```
public LinkedList<Integer> queuecontent
= new LinkedList<Integer>();
```

4. 2. add edges is a function that make the link between places
5. The Graph constructur take the start state(initial) and the goal state from the user and passed to the BFS function

```
Graph(int startstate, int goalstate){
System.out.println("I am in graph function");
Graph g = new Graph(10);
//home
g.addEdge(0, 8); //

//school
g.addEdge(1, 2); //
//pharmacy
g.addEdge(2, 4);
g.addEdge(2, 3);


//supermarket
g.addEdge(3, 5);
g.addEdge(3, 9);

//library
g.addEdge(5, 9);

//uni
g.addEdge(6, 2);

//coffeeshop
g.addEdge(8, 1);

//airport
System.out.println(
"Using Breadth First Search "
);
g.BFS(startstate, goalstate, startstate);
}
```

```
// Function to add an edge into the graph
void addEdge(int v, int w) {
adj[v].add(w);
}
```

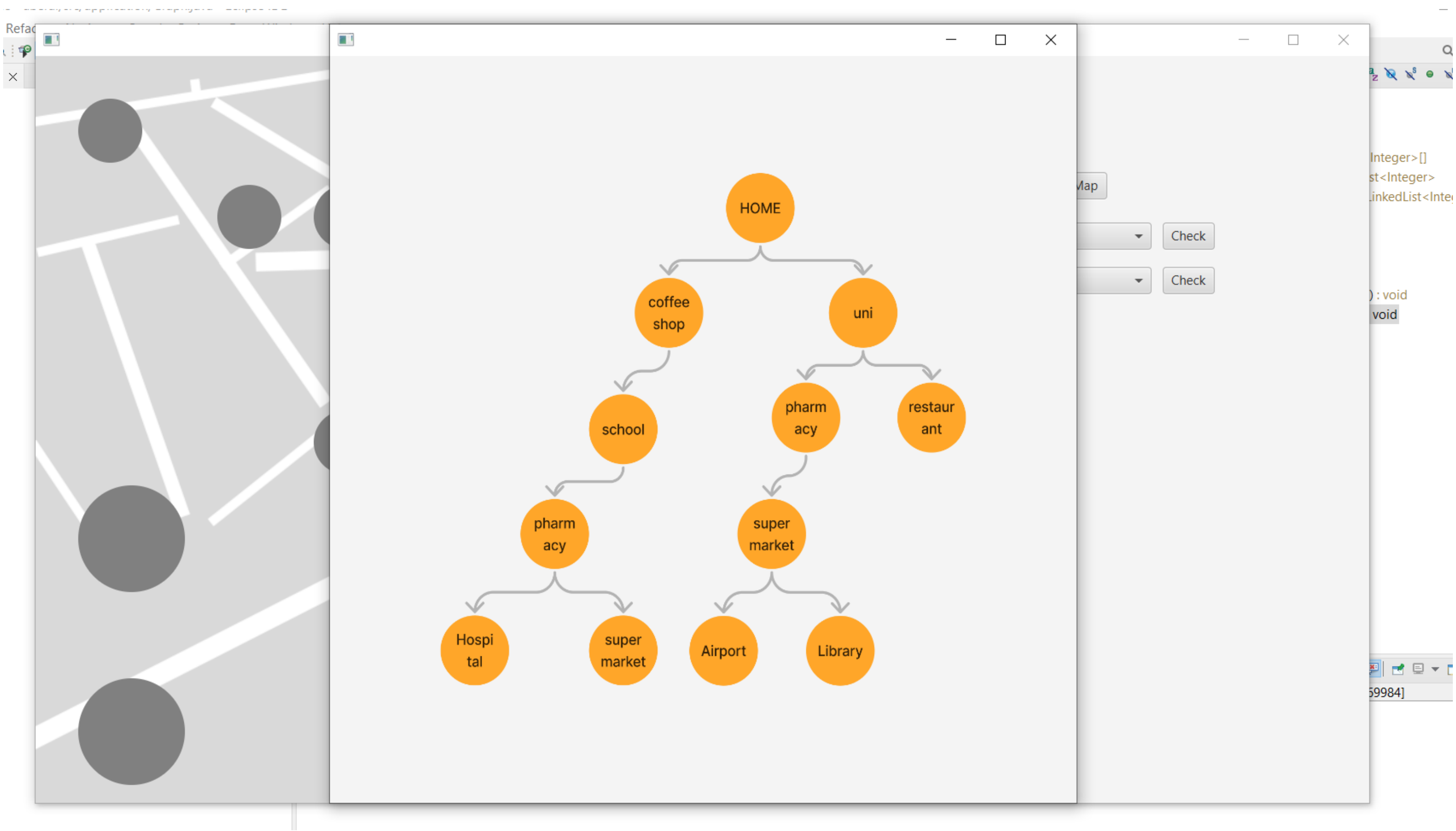
7. The second constructur called by the first one above take size v as parameter and create the matrix

```
Graph(int v)
{
this.V = v;
adj = new LinkedList[v];
for (int i = 0; i < v; ++i)
adj[i] = new LinkedList();
}
```

8. BFS Funtion

```
void BFS(int s, int goalstate, int initialstate)
{
// Mark all the vertices as not visited(By default
// set as false)
boolean visited[] = new boolean[V];
// Create a queue for BFS
/*LinkedList<Integer> queue
= new LinkedList<Integer>();*/
// Mark the current node as visited and enqueue it
visited[s] = true;
queue.add(s);
while (queue.size() != 0) {
// Dequeue a vertex from queue and print it
s = queue.poll();
System.out.print(s + " ");
queuecontent.add(s);
// Get all adjacent vertices of the dequeued
// vertex s If a adjacent has not been visited,
// then mark it visited and enqueue it
Iterator<Integer> i = adj[s].listIterator();
int n = 0;
while (i.hasNext()) {
n = i.next();
if (n == goalstate) {
visited[n] = true;
queue.add(n);
break;
}
if (!visited[n]) {
visited[n] = true;
queue.add(n);
}
}
System.out.print('\n'+ "path n : "+n + " ");
}
System.out.println(goalstate);
queuecontent.add(goalstate);
System.out.println("queuecontent :"+ queuecontent);
```

The show binary tree button



> You can find the full code with graphics code in the zip file