Set 10

The source code for the AbstractGrid class is in Appendix D.

1. Where is the isValid method specified? Which classes provide an implementation of this method?

Don't know what "specified" means... It is declared in Grid and implemented in BoundedGrid and UnboundedGrid.

2. Which AbstractGrid methods call the isValid method? Why don't the other methods need to call it?

getValidAdjacentLocations. because the other methods call getValidAdjacentLocations.

3. Which methods of the Grid interface are called in the getNeighbors method? Which classes provide implementations of these methods?

getOccupiedAdjacentLocations, AbstractGrid

4. Why must the get method, which returns an object of type E, be used in the getEmptyAdjacentLocations method when this method returns locations, not objects of type E?

Because the getEmptyAdjacentLocations method has to return an ArrayList of empty adjacent locations, and it uses get to check.

5. What would be the effect of replacing the constant Location.HALF_RIGHT with Location.RIGHT in the two places where it occurs in the getValidAdjacentLocations method?

It will only check and return valid adjacent locations horizonally or vertically, not diagonally.

Set 11

The source code for the BoundedGrid class is in Appendix D.

1. What ensures that a grid has at least one valid location?

```
if (rows <= 0)
   throw new IllegalArgumentException("rows <= 0");
if (cols <= 0)
   throw new IllegalArgumentException("cols <= 0");
```

2. How is the number of columns in the grid determined by the getNumCols method? What assumption about the grid makes this possible?

The length of the first row. numRows() > 0.

3. What are the requirements for a Location to be valid in a BoundedGrid?

```
0 <= loc.getRow() && loc.getRow() < getNumRows()
   && 0 <= loc.getCol() && loc.getCol() < getNumCols();
```

In the next four questions, let r = number of rows, c = number of columns, and n = number of occupied locations.

4. What type is returned by the getOccupiedLocations method? What is the time complexity (Big-Oh) for this method?

ArrayList<Location>
O(rc)

5. What type is returned by the get method? What parameter is needed? What is the time complexity (Big-Oh) for this method?

The generic type E.
Location of a valid grid.
O(1)

6. What conditions may cause an exception to be thrown by the put method? What is the time complexity (Big-Oh) for this method?

The argument loc is an invalid location or the object is null.
O(1)

7. What type is returned by the remove method? What happens when an attempt is made to remove an item from an empty location? What is the time complexity (Big-Oh) for this method?

The generic type E.
null is returned.
O(1)

8. Based on the answers to questions 4, 5, 6, and 7, would you consider this an efficient implementation? Justify your answer.

Yes, in some situations, it is efficient. But when the grid is big while the amount of objects in the grid is small, the getOccupiedLocations method will be unworthy slow and the two-dimensional array will be very big.

Set 12

The source code for the UnboundedGrid class is in Appendix D.

1. Which method must the Location class implement so that an instance of HashMap can be used for the map? What would be required of the Location class if a TreeMap were used instead? Does Location satisfy these requirements?

hashCode
compareTo
Yes

2. Why are the checks for null included in the get, put, and remove methods? Why are no such checks included in the corresponding methods for the BoundedGrid?

The prevent the argument from being null. Because in BoundedGrid null is used to represent an

empty grid, so putting a null does not matter, in UnboundedGrid every element in the Map represents an object, a null will really be "added" to the grid and cause error.

3. What is the average time complexity (Big-Oh) for the three methods: get, put, and remove? What would it be if a TreeMap were used instead of a HashMap?

	get	put	remove
HashMap	O(1)	O(1)	O(1)
TreeMap	O(log(n))	O(log(n))	O(log(n)) // n is the number of elements

4. How would the behavior of this class differ, aside from time complexity, if a TreeMap were used instead of a HashMap?

The objects in the grid will be automatically sorted by their location.

5. Could a map implementation be used for a bounded grid? What advantage, if any, would the two-dimensional array implementation that is used by the BoundedGrid class have over a map implementation?

Yes. Though the time complexity are both O(1) for two-dimensional array and hashMap, but the former is actually a little faster than the latter.

Exercises

Methods	SparseGridNode version	LinkedList<OccupantInCol> version	HashMap version	TreeMap version
getNeighbors	O(c)	O(c)	O(1)	O(log(n))
getEmptyAdjacentLocations	O(c)	O(c)	O(1)	O(log(n))
getOccupiedAdjacentLocations	O(c)	O(c)	O(1)	O(log(n))
getOccupiedLocations	O(n)	O(n)	O(n)	O(n)
get	O(c)	O(c)	O(1)	O(log(n))
put	O(1)	O(1)	O(1)	O(log(n))
remove	O(c)	O(c)	O(1)	O(log(n))

Implement the methods specified by the Grid interface using this data structure. What is the Big-Oh efficiency of the get method? What is the efficiency of the put method when the row and column index values are within the current array bounds? What is the efficiency when the array needs to be resized?

O(1), O(1), O(originalSize ^ 2)