# Java Developer Position Assessment

Candidate,

Below is an assessments designed to test problem solving skills, logic, Java, and Object Oriented Design. You are highly encouraged to use any resources at your disposal to complete the tasks.

## Outbreak

Candidate, the Department of Disease Spotting needs your help finding outbreaks!

### Requirements

* Given a floor plan, consisting of rooms arranged in a grid-like pattern, determine if there is an outbreak.
* Connected rooms must share a wall.
* An outbreak exists if there are 5 or more connected rooms which have been infected; in a chain (the chain can be in any direction, but each room must be connected in the chain).
* Diagonal rooms are not connected.
* The Department of Disease Spotting has provided the following resources to assist you. You must use them; however, you may use any other functions or classes you wish to complete this task.
* The Department of Disease Spotting, takes no chances, and has requested that the isOutbreak method be thoroughly unit-tested using sample data (below) to prove that the routine will work correctly every time.
* Points will be awarded according to accuracy, testability, and performance.

public class Room {

private final boolean isInfected;

public Room(boolean infected) {

isInfected = infected;

}

}

public class Outbreak

{

public static boolean isOutbreak(Room[][] floor)

{

/\*

\* Your code here.

\*/

}

}

### Example of Rooms on a floor

|  |  |
| --- | --- |
| Outbreak  |X|X|X| | | | | | | |  | | |X|X| | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | | No Outbreak  | |X| | | | | | | | |  | | |X| | | | | | | |  | | | |X| | | | | | |  | | | | |X| | | | | |  | | | | | |X| | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | |  | | | | | | | | | | | |

### Test Data

Here are a few example test cases and what they should return:

Room[][] verticalTrue = new Room[][] {

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(true), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) }

};

Room[][] horizontalTrue = new Room[][] {

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(true), new Room(true), new Room(true), new Room(true), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) }

};

Room[][] noInfection = new Room[][] {

{new Room(true), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(true), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(true), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(true), new Room(false), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(true), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(true), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) },

{new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false), new Room(false) }

};