## Background: Location location =new Location(0, 0);

In this lab, you complete the implemention for a class called MyBoundedGrid<E> to represent a grid of objects of type E. Throughout the year, we will put various objects into this grid: bugs, critters, chess pieces, blocks, etc. (The grid is "bounded" in the sense that it is not infinite in size.) The MyBoundedGrid<E> class makes use of the Location class, which simply remembers a row and column index. A summary of the Location class is given below, but you will probably only use getRow, getCol, and equals in this lab.

**Location Class (implements Comparable)**

public **Location**(int r, int c)

public int **getRow**()

public int **getCol**()

public Location **getAdjacentLocation**(int direction)

public int **getDirectionToward**(Location target)

public boolean **equals**(Object other)

public int **hashCode**()

public int **compareTo**(Object other)

public String **toString**()

NORTH, EAST, SOUTH, WEST, NORTHEAST, SOUTHEAST, NORTHWEST, SOUTHWEST, LEFT, RIGHT, HALF\_LEFT, HALF\_RIGHT, FULL\_CIRCLE, HALF\_CIRCLE, AHEAD

Go ahead and download all the posted lab files.

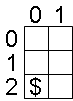
## Background: MyBoundedGrid<E>

The MyBoundedGrid<E> class allows us to place objects into a grid, as shown in the following example.

MyBoundedGrid grid = new MyBoundedGrid<String>(3, 2);

grid.put(new Location(2, 0), "$");

This code creates a grid with 3 rows and 2 columns, containing a "$" in the lower left corner, as the following diagram shows.



Here is a summary of MyBoundedGrid<E>'s constructor and methods.

**MyBoundedGrid<E> class**

public **MyBoundedGrid<E>**(int rows, int cols)

public int **getNumRows**()

public int **getNumCols**()

public boolean **isValid**(Location loc)

public E **put**(Location loc, E obj)

public E **remove**(Location loc)

public E **get**(Location loc)

public ArrayList<Location> **getOccupiedLocations**()

In this lab, we'll use a 2-dimensional array of Objects to represent the grid. Your MyBoundedGrid<E> class therefore has only a single instance variable. (Don't add any others!)

private Object[][] occupantArray;

The following instance diagram shows the result of executing the code segment shown earlier.

## Exercise 1: Growing a Grid

Go ahead and complete the MyBoundedGrid<E> class you downloaded. Test your code as you go by running the GridMonster. Beware – it is rude.

## Background: The Block Class

The first kind of thing we'll place in our grid is a colored Block—an object which keeps track of its color, location, and the MyBoundedGrid<Block> that contains it. When the Block is *not* in a grid, its grid and location instance variables are both null. When the Block *is* in a grid, its location instance variable must match the Block's location within the MyBoundedGrid associated with its grid instance variable. This means that both the Block and its grid are keeping track of the Block's location. It's up to the Block class to keep these consistent.

The following example shows how to place a Block in a MyBoundedGrid<Block>.

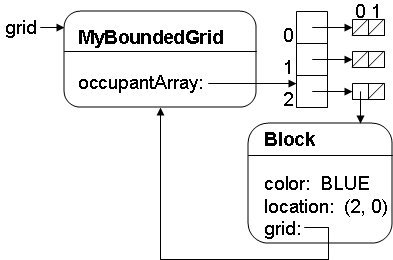
MyBoundedGrid<Block> grid;

grid = new MyBoundedGrid<Block>(3, 2);

Block block = new Block();

block.putSelfInGrid(grid, new Location(2, 0));

The following instance diagram shows the result of executing this code.



Here is a summary of the Block class.

**Block Class**

public **Block**()

public Color **getColor**()

public void **setColor**(Color newColor)

public MyBoundedGrid<Block> **getGrid**()

public Location **getLocation**()

public void **putSelfInGrid**(MyBoundedGrid<Block> gr, Location loc)

public void **removeSelfFromGrid**()

public void **moveTo**(Location newLocation)

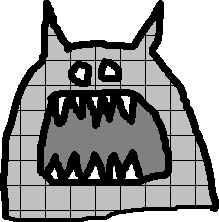
public **String** toString()

## Exercise 2: Coder's Block

Go ahead and complete the Block class you downloaded. You may test your work if you like or wait to use the automated tester in the next exercise.

## Exercise 3: Monster Test

Now that you've completed the MyBoundedGrid<E> and Block classes, it's time to see if you have what it takes to beat the dreaded GridMonster!



Go ahead and run the GridMonster class you downloaded. Your code must pass each of the GridMonster's tests in order to complete the lab. But beware! The GridMonster has a nasty habit of insulting any code that displeases him (or her). Good luck!