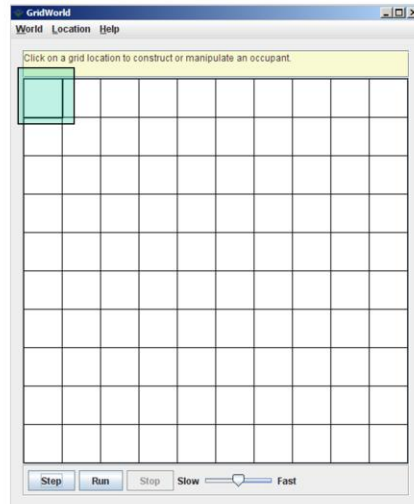




# What is GridWorld?



**Row = 0**

**Column = 0**

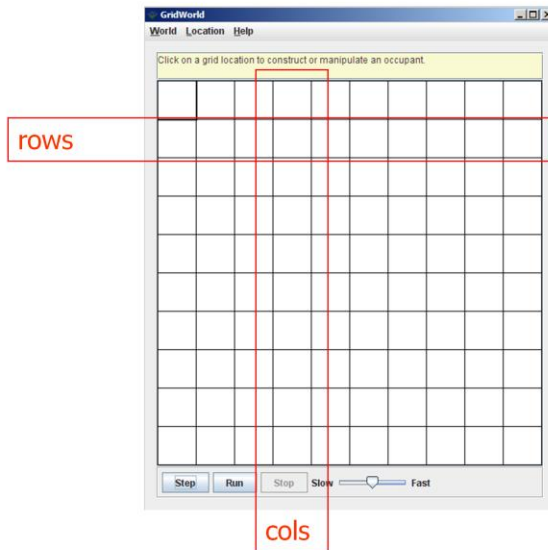
© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

# What is GridWorld?



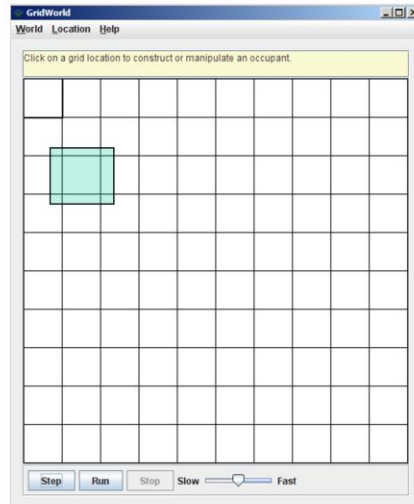
**A grid is a structure that has rows and columns.**

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

# What is GridWorld?



**Row = 2**

**Column = 1**

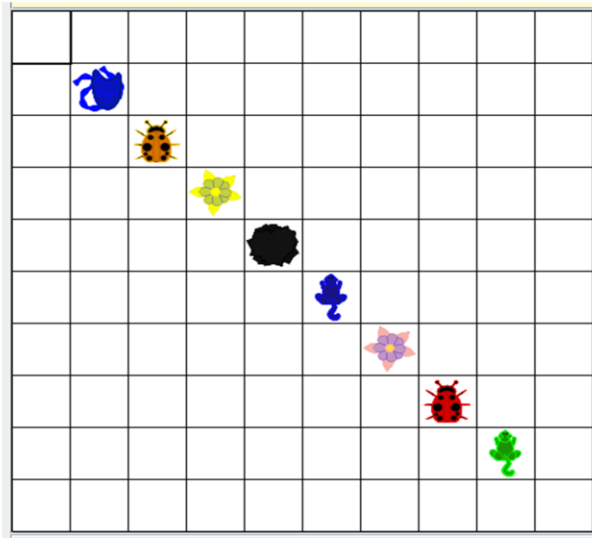
© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

# What is GridWorld?



© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



**Grid is an interface that details the behaviors expected of a Grid.**

**Grid was designed as an interface because many different structures could be used to store the grid values.**

**An interface works perfectly due to the large number of unknowns.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

Grid is a row / column structure that stores Objects.

The location of each Object is determined by the Location provided when putting the Object in the grid.



**A grid can store any type of Object.**

```
Grid<Integer> intGrid;  
intGrid = new BoundedGrid<Integer>(5,5);
```

```
Grid<String> stringGrid;  
stringGrid = new UnboundedGrid<String>();
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

Grid is a row / column structure that stores Objects.

The location of each Object is determined by the Location provided when putting the Object in the grid.



## Grid

### abstract methods

Name	Use
<code>get(loc)</code>	returns the ref at location loc
<code>getEmptyAdjacentLocations(loc)</code>	gets the valid empty locs in 8 dirs
<code>getNeighbors(loc)</code>	returns the objs around this
<code>getNumCols()</code>	gets the # of cols for this grid
<code>getNumRows()</code>	gets the # of rows for this grid
<code>getOccupiedAdjacentLocations(loc)</code>	gets the valid locs in 8 dirs that contain objs
<code>getOccupiedLocations()</code>	gets locs that contain live objs
<code>getValidAdjacentLocations(loc)</code>	gets the valid locs in 8 dirs
<code>isValid(loc)</code>	checks to see if loc is valid
<code>put(loc, obj)</code>	put the obj in grid at location loc
<code>remove(loc)</code>	take the obj at location loc out of the grid

```
import info.gridworld.grid.Grid;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Grid

rows	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
rows	0	0	0	0	0

**A grid is a structure that has rows and columns.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

# Grid

**A grid is a structure that has rows and columns.**

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
cols			cols	

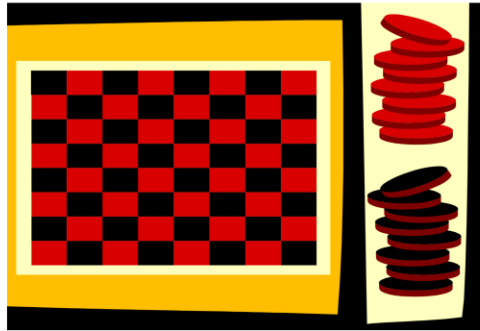
© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

# Grid

A screenshot of a software window titled "GridWorld" with a menu bar (World, Location, Help) and a toolbar (Step, Run, Stop, Slow, Fast). The main area displays a 3x3 grid representing Pascal's Triangle:

1		
1	1	
1	2	1

**A grid is a structure that has rows and columns.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.



© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# AbstractGrid

**AbstractGrid is an abstract class that implements five of the Grid interface methods that would be common to any type of grid implementation.**

**AbstractGrid works best as an abstract class as there are some things known and some things that are still unknown.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

AbstractGrid implements the methods from the Grid interfaces that will be the same for all types of grids.

There are five methods that will be the same no matter what type of data structure is used to store grid elements.

# AbstractGrid

implements Grid

## frequently used methods

Name	Use
<code>getNeighbors(loc)</code>	returns the objs around this
<code>getValidAdjacentLocations(loc)</code>	gets the valid locs in 8 dirs
<code>getEmptyAdjacentLocations(loc)</code>	gets the valid empty locs in 8 dirs
<code>getOccupiedAdjacentLocations()</code>	gets adjacent locs that contain live objs
<code>toString()</code>	returns a list of the stuff in the grid

```
import info.gridworld.grid.AbstractGrid;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# BoundedGrid

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



# BoundedGrid

**BoundedGrid extends AbstractGrid and implements the remaining Grid interface methods.**

**BoundedGrid stores everything in a matrix.**

**Object[][] occupantArray;**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

BoundedGrid implements the remaining Grid methods that AbstractGrid did not implement.

BoundedGrid uses a matrix to store all object references.

## BoundedGrid

extends AbstractGrid

### frequently used methods

Name	Use
BoundedGrid(rows, cols)	creates a new Grid[rows,cols]
getNumCols()	gets the # of cols for this grid
getNumRows()	gets the # of rows for this grid

```
import info.gridworld.grid.BoundedGrid;
```

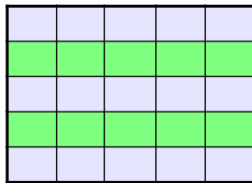
**BoundedGrid is a matrix of object references.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# BoundedGrid

```
BoundedGrid<Integer> grid;  
grid = new BoundedGrid<Integer>(5,5);
```

```
System.out.println(grid.getNumRows());  
System.out.println(grid.getNumCols());  
System.out.println(grid);
```



## OUTPUT

```
5  
5  
{ }
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

BoundedGrid is a grid that has a set number of rows and columns.

BoundedGrid has a limited number of storage locations.

**open  
bgridone.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

## **BoundedGrid**

extends **AbstractGrid**

### **frequently used methods**

<b>Name</b>	<b>Use</b>
<b>isValid(location)</b>	checks if location is inside of the grid
<b>get(location)</b>	gets the Object at specified location
<b>getOccupiedLocations()</b>	returns a list of occupied locs
<b>put(location, thang)</b>	put thang at spot location
<b>remove(location)</b>	removes thang at spot location

```
import info.gridworld.grid.BoundedGrid;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# BoundedGrid

```
BoundedGrid<Integer> grid;  
grid = new BoundedGrid<Integer>(5,5);  
grid.put(new Location(2,2),4);  
grid.put(new Location(1,1),11);  
System.out.println(grid);  
System.out.println(grid.isValid(new Location(-2,3)));  
System.out.println(grid.isValid(new Location(2,3)));
```

## **OUTPUT**

```
{(1, 1)=11, (2, 2)=4}  
false  
true
```

	11			
		4		

© A+ Computer Science - www.apluscompsci.com

BoundedGrid is a grid that has a set number of rows and columns.

BoundedGrid has a limited number of storage locations.

BoundedGrid has a top left storage location of (0,0).

BoundedGrid has a bottom right storage location of (length-1,length-1).

The isValid method will determine if a provided location is within the bounds of the provided grid.

# BoundedGrid

```
BoundedGrid<Integer> grid;  
grid = new BoundedGrid<Integer>(5,5);  
//add 4 at 2,2 - add 11 at 1,1 - add 3 at 0,2 - add 6 at 0,1  
System.out.println(grid);  
grid.remove(new Location(1,1));  
System.out.println(grid);  
grid.remove(new Location(1,1));  
System.out.println(grid);
```

## **OUTPUT**

```
{(0, 1)=6, (0, 2)=3, (1, 1)=11, (2, 2)=4}  
{(0, 1)=6, (0, 2)=3, (2, 2)=4}  
{(0, 1)=6, (0, 2)=3, (2, 2)=4}
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

When a grid is printed, each location that has a value is shown and the value stored in that location is also shown.

**open**  
**bgridtwo.java**  
**bgridthree.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



# UnboundedGrid

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# UnboundedGrid

**UnboundedGrid extends AbstractGrid and implements the remaining Grid interface methods.**

**UnboundedGrid stores locations and references in a Map.**

**Map<Location, E> occupantMap;**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

The UnboundedGrid example provided with GridWorld uses a map to store all object references.

A map is not the only data structure that could be used to store an unlimited number of objects.

## UnboundedGrid

extends AbstractGrid

### frequently used methods

Name	Use
UnboundedGrid()	creates an empty grid
getNumCols()	returns -1
getNumRows()	returns -1

```
import info.gridworld.grid.UnboundedGrid;
```

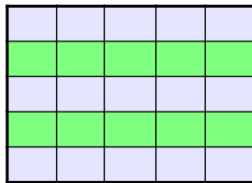
**UnboundedGrid stores loc and objs in a Map.**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# UnboundedGrid

```
UnboundedGrid<Integer> grid;  
grid = new UnboundedGrid<Integer>();
```

```
System.out.println(grid.getNumRows());  
System.out.println(grid.getNumCols());  
System.out.println(grid);
```



## **OUTPUT**

```
-1  
-1  
{}
```

© A+ Computer Science - www.apluscompsci.com

UnboundedGrid is a grid that does not have a set number of rows and columns.

UnboundedGrid has an unlimited number of storage locations.

**open  
ubgridone.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

## UnboundedGrid

extends AbstractGrid

### frequently used methods

Name	Use
isValid(location)	checks if location is inside of the grid
get(location)	gets the Object at specified location
getOccupiedLocations()	returns a list of occupied locs
put(location, thang)	put thang at spot location
remove(location)	removes thang at spot location

```
import info.gridworld.grid.UnboundedGrid;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# UnboundedGrid

```
UnboundedGrid<Integer> grid;  
grid = new UnboundedGrid<Integer>();  
grid.put(new Location(2,2),4);  
grid.put(new Location(1,1),11);  
System.out.println(grid);  
System.out.println(grid.isValid(new Location(-2,3)));  
System.out.println(grid.isValid(new Location(2,3)));
```

## **OUTPUT**

```
{(1, 1)=11, (2, 2)=4}  
true  
true
```

	11			
		4		

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

UnboundedGrid is a grid that does not have a set number of rows and columns.

UnboundedGrid has an unlimited number of storage locations.

UnboundedGrid has no set top left storage location.

UnboundedGrid has no set bottom right storage location.

The isValid method will determine if a provided location is within the bounds of the provided grid.

All locations are valid for an unbounded grid.

# UnboundedGrid

```
UnboundedGrid<Integer> grid;  
grid = new UnboundedGrid<Integer>();  
//add 4 at 2,2 - add 11 at 1,1 - add 3 at 0,2 - add 6 at 0,1  
System.out.println(grid);  
grid.remove(new Location(1,1));  
System.out.println(grid);  
grid.remove(new Location(1,1));  
System.out.println(grid);
```

## **OUTPUT**

```
{(0, 1)=6, (0, 2)=3, (1, 1)=11, (2, 2)=4}  
{(0, 1)=6, (0, 2)=3, (2, 2)=4}  
{(0, 1)=6, (0, 2)=3, (2, 2)=4}
```

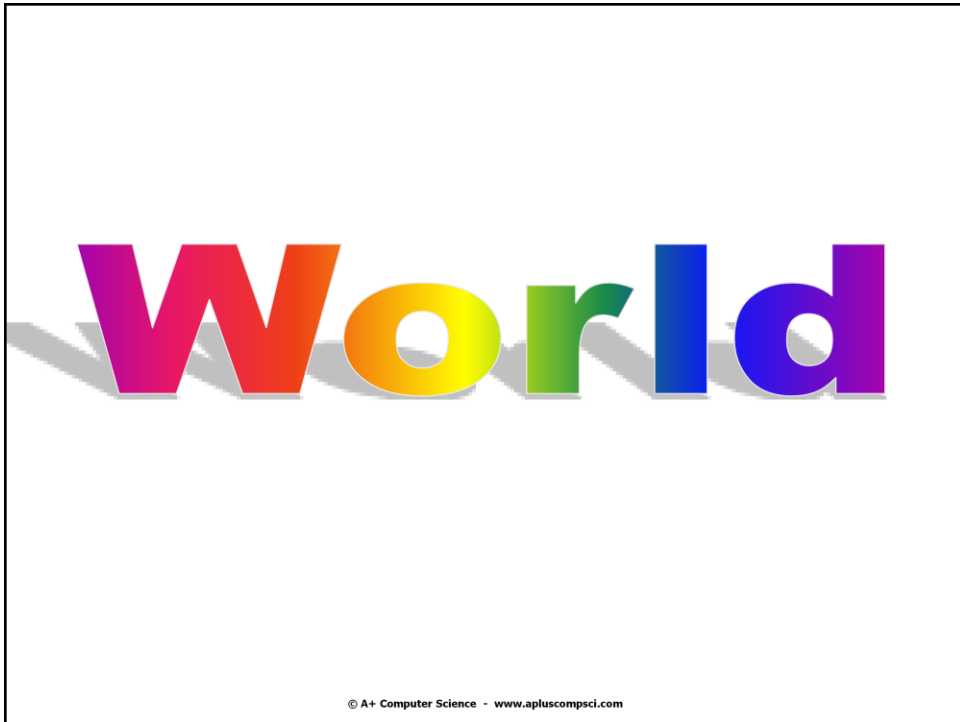
© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

When a grid is printed, each location that has a value is shown and the value stored in that location is also shown.



**open**  
**ubgridtwo.java**  
**ubgridthree.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)



© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

## **World**

### **frequently used methods**

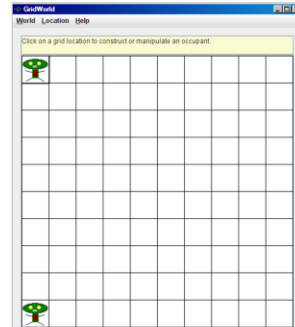
<b>Name</b>	<b>Use</b>
<b>World()</b>	creates a new world using 10X10 grid
<b>World(grid)</b>	creates a new world using grid
<b>add(loc, thang)</b>	add thang at spot loc
<b>show()</b>	makes the world visible

```
import info.gridworld.world.World;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# World

```
World<Monster> world = new World<Monster>();  
world.add(new Location(0,0), new Monster());  
world.add(new Location(9,0), new Monster());  
world.show();
```



© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

World is used to house a Grid. World is used to display the Grid graphically.

**open  
worldone.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

## **World**

### **frequently used methods**

<b>Name</b>	<b>Use</b>
<b>getGrid()</b>	returns a ref to the world's grid
<b>remove(location)</b>	removes the thang at location
<b>locationClicked(loc)</b>	activated by a mouse click in the grid

```
import info.gridworld.world.World;
```

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# World

```
BoundedGrid<Integer> grid;  
grid = new BoundedGrid<Integer>(5,5);  
grid.put(new Location(2,2),4);  
grid.put(new Location(1,1),9);  
grid.put(new Location(0,4),11);  
grid.put(new Location(4,3),5);  
grid.put(new Location(2,0),1);
```

```
World<Integer> world;  
world = new World<Integer>(grid);  
world.show();
```

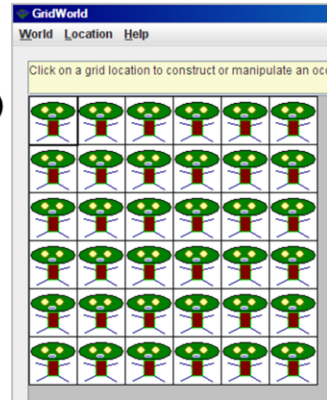
World	Location	Value
0	4	11
1	1	9
2	2	4
4	3	5
2	0	1

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

World is used to house a Grid. World is used to display the Grid graphically.

# World

```
World<Monster> world;  
world = new World<Monster>(new BoundedGrid(6,6));  
world.show();  
Grid<Monster> grid = world.getGrid();  
for(int r=0; r<grid.getNumRows(); r++)  
{  
    for(int c=0; c<grid.getNumCols(); c++)  
    {  
        grid.put(new Location(r,c),  
                new Monster(r,c,2));  
    }  
}
```



© A+ Computer Science - www.apluscompsci.com

World is used to house a Grid. World is used to display the Grid graphically.



**open**  
**worldtwo.java**  
**worldthree.java**

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)

# Start work on Grid Labs

© A+ Computer Science - [www.apluscompsci.com](http://www.apluscompsci.com)