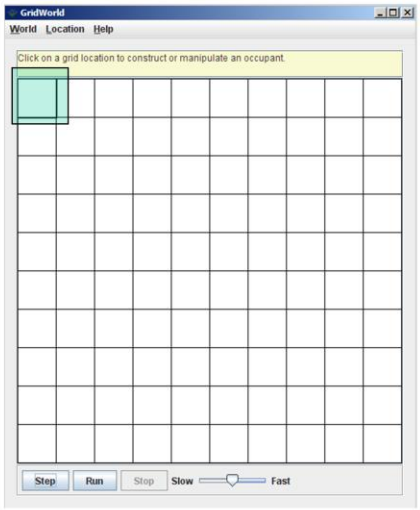

GridWorld

What is GridWorld?



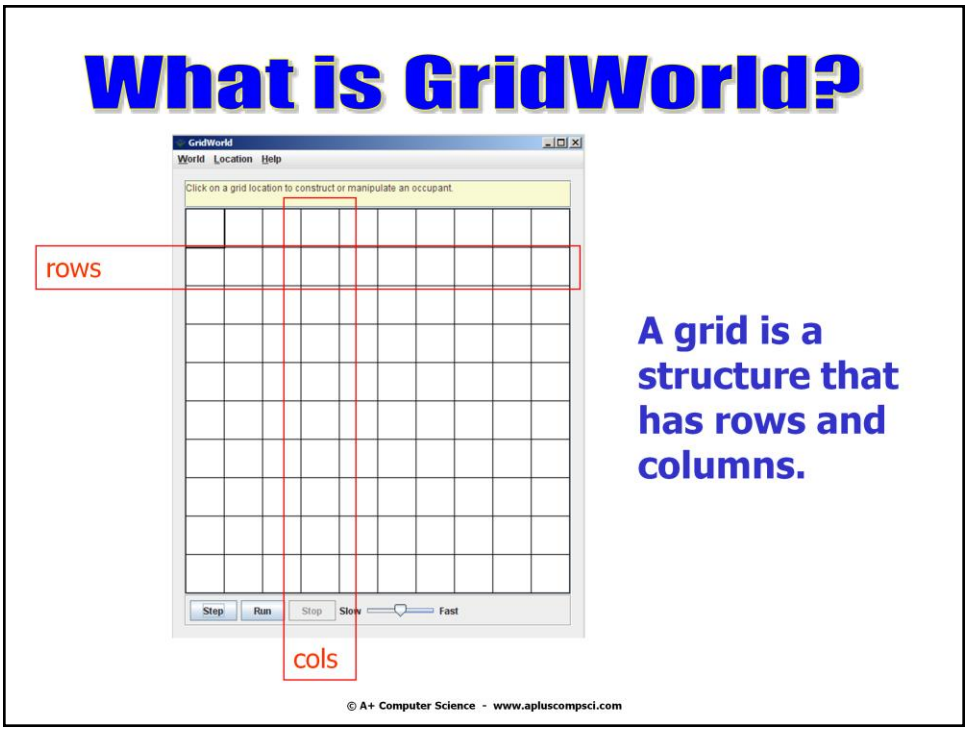
Row = 0

Column = 0

A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

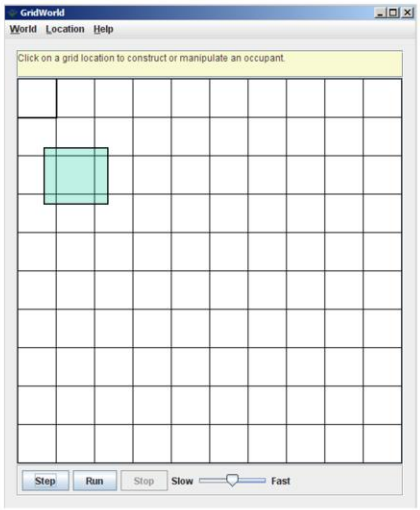


A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

What is GridWorld?



Row = 2

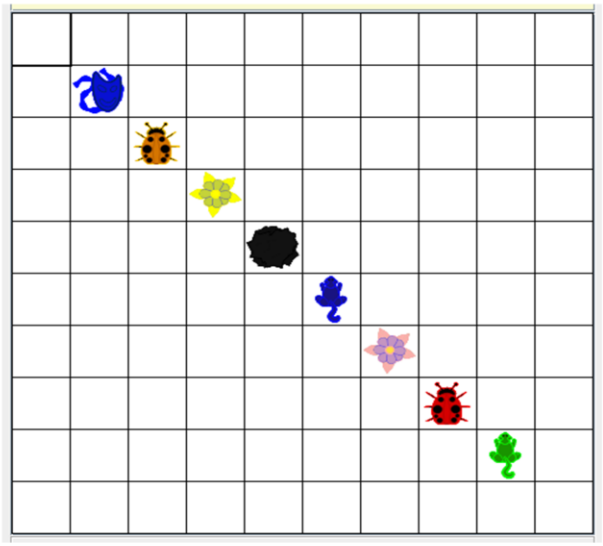
Column = 1

A grid is a structure that has rows and columns.

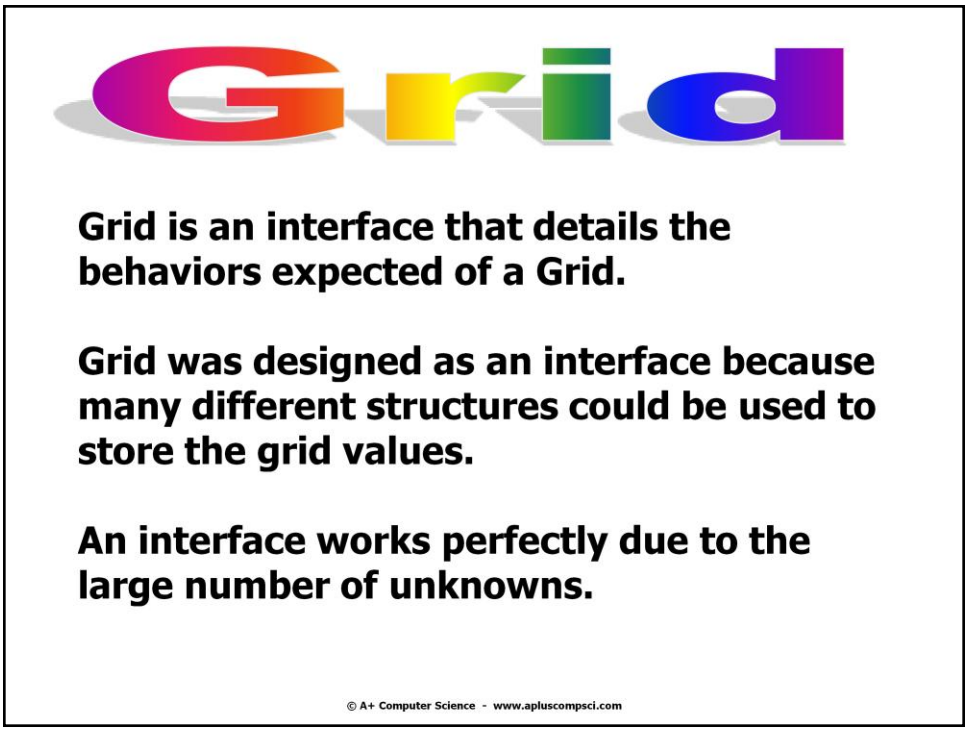
A spreadsheet is a grid.

A checker board is a grid.

What is GridWorld?

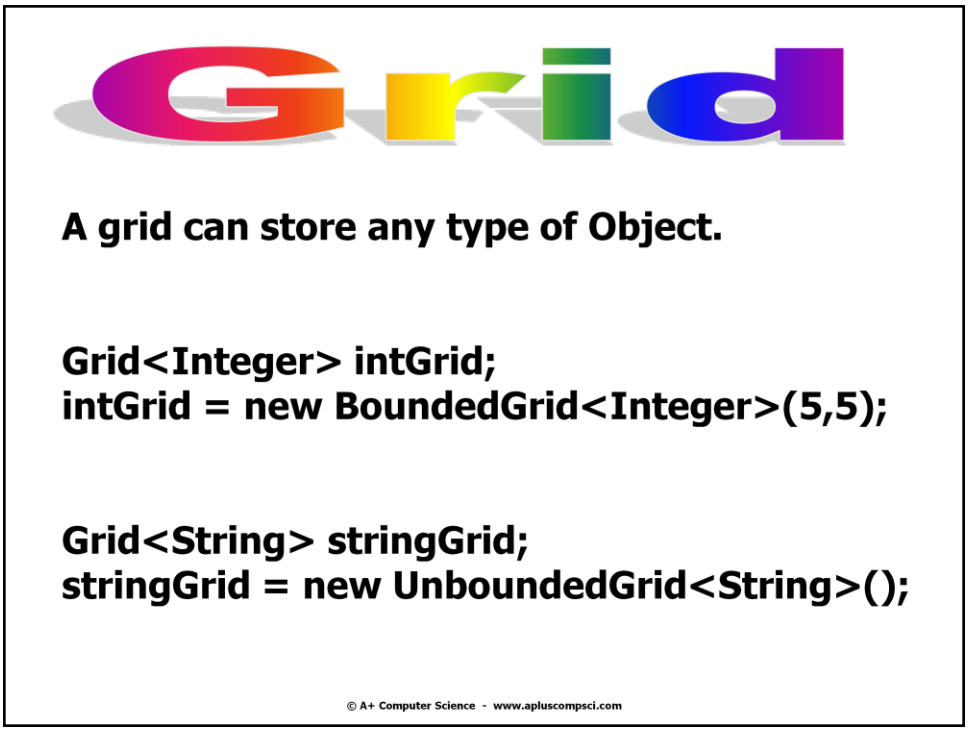



Grid
© A+ Computer Science - www.apluscompsci.com



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.



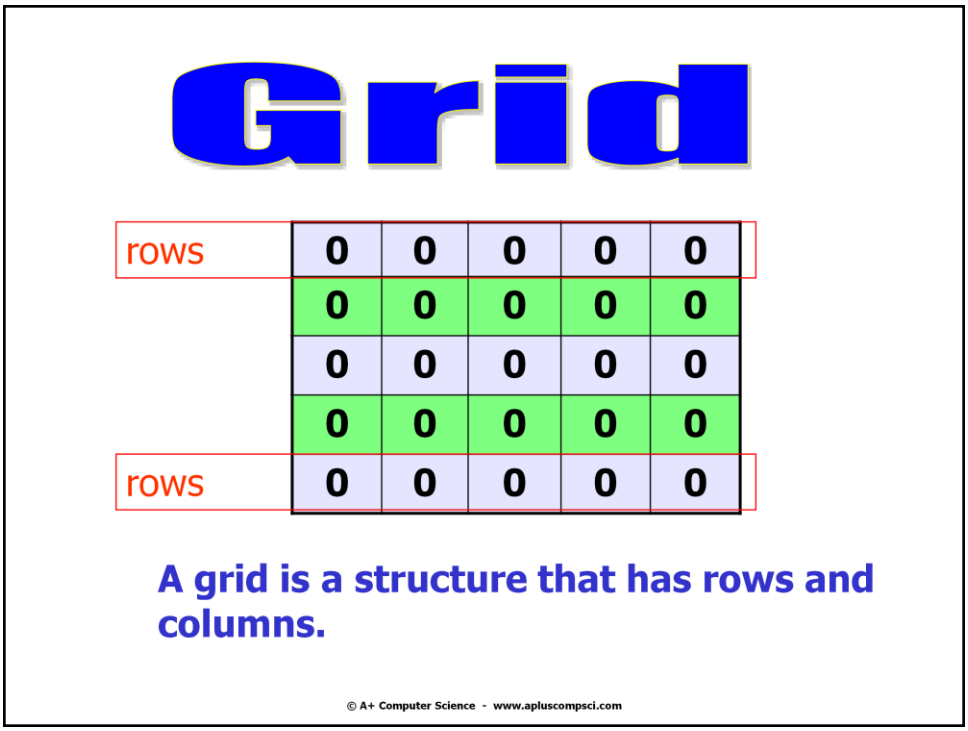
Grid is a row / column structure that stores Objects.

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

Grid
abstract methods

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

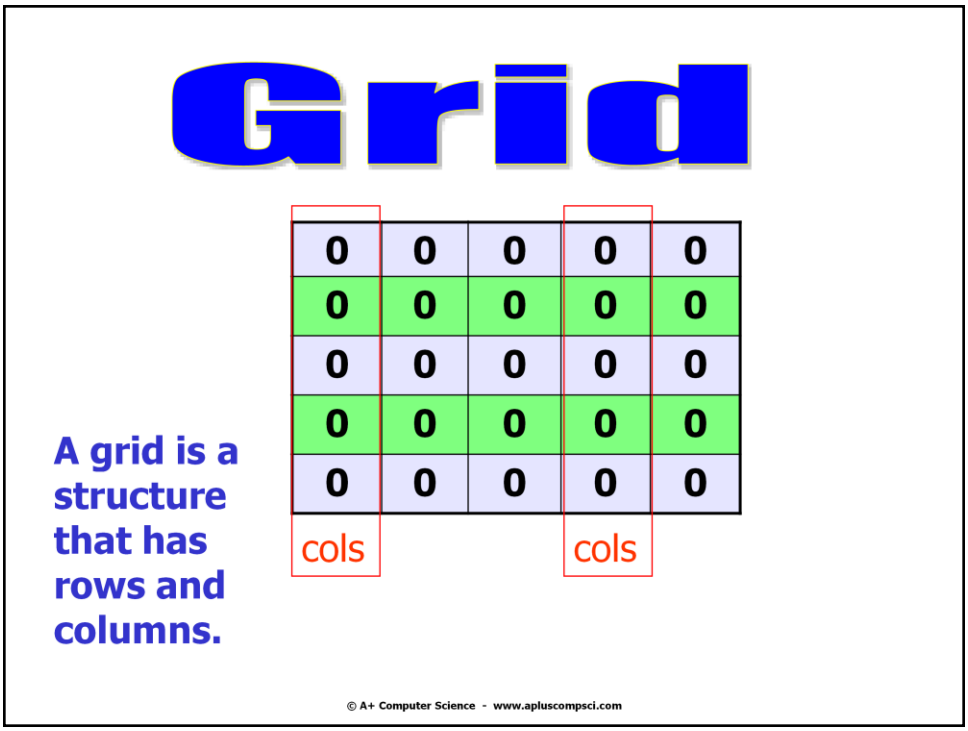
```
import info.gridworld.grid.Grid;
```



A grid is a structure that has rows and columns.

A spreadsheet is a grid.

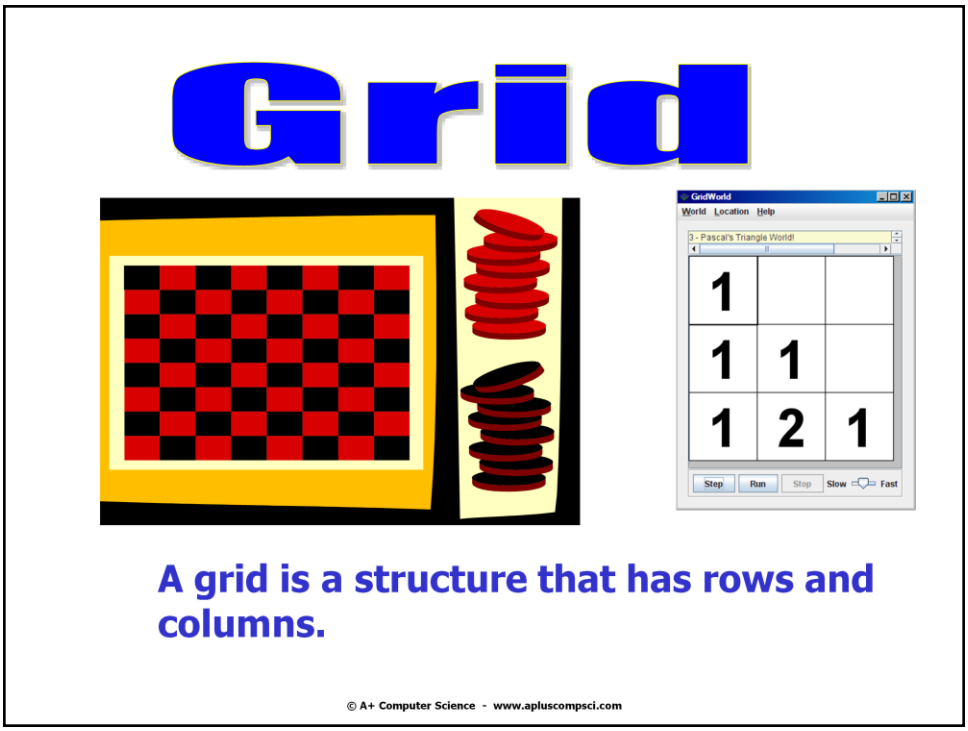
A checker board is a grid.



A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.

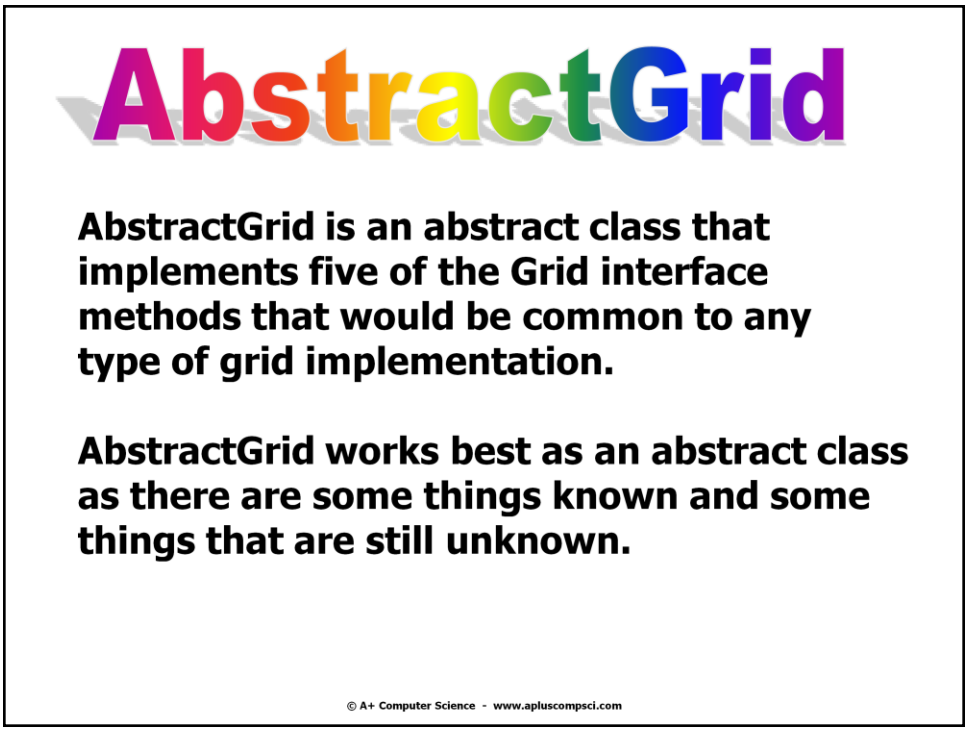


A grid is a structure that has rows and columns.

A spreadsheet is a grid.

A checker board is a grid.


AbstractGrid
© A+ Computer Science - 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.

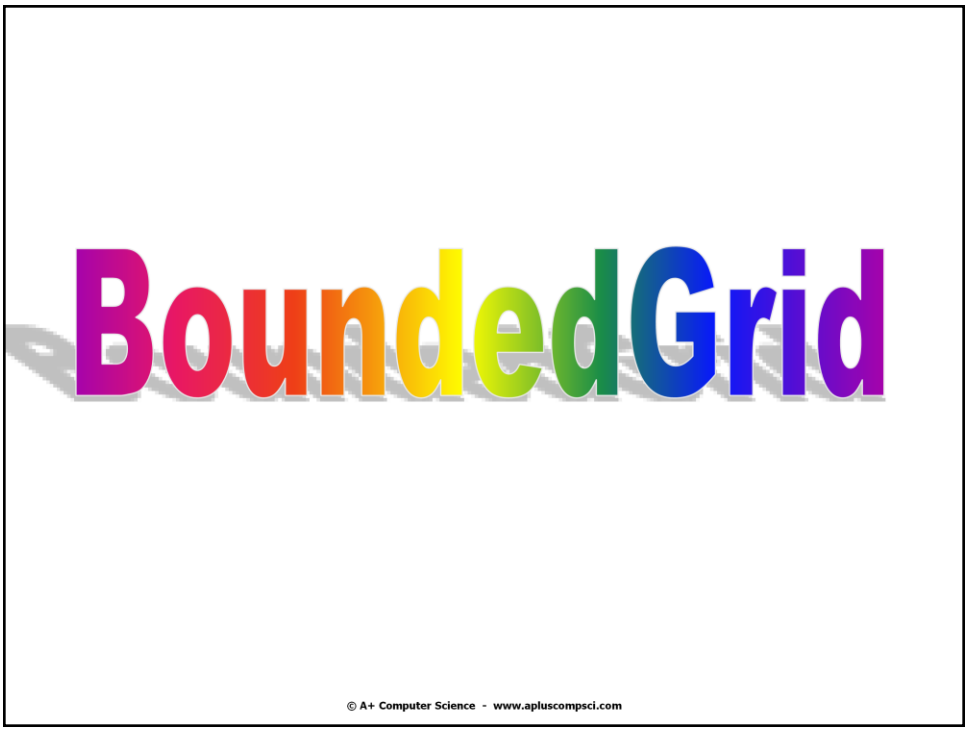
AbstractGrid

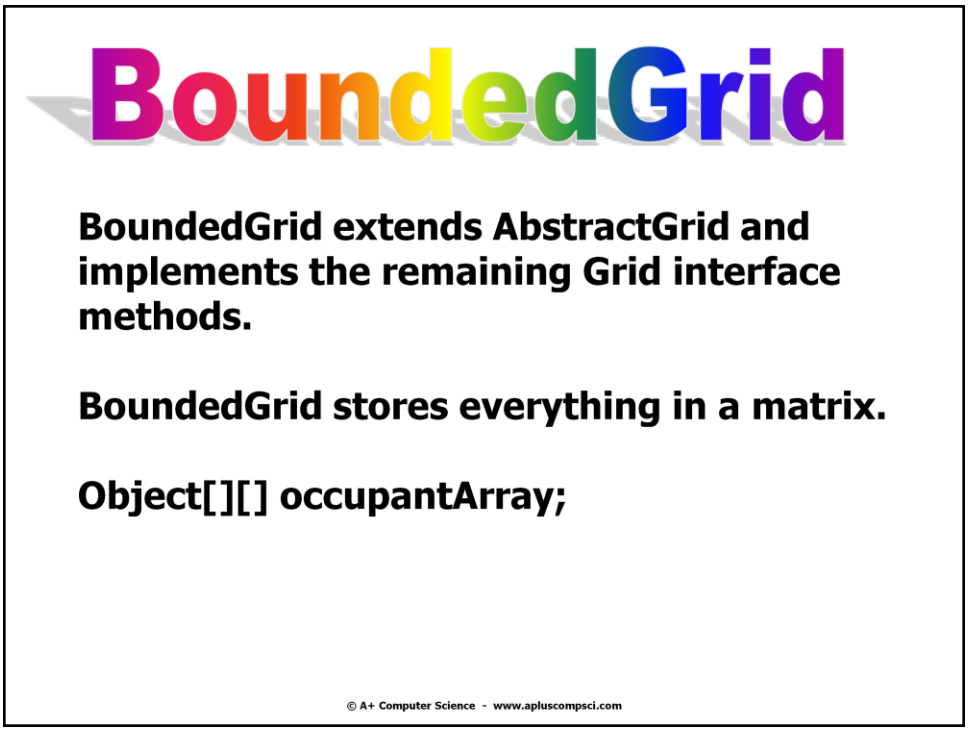
implements Grid

frequently used methods

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

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


BoundedGrid
© A+ Computer Science - 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

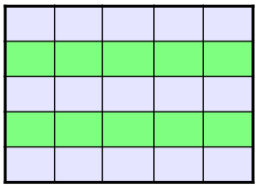
Name	Use
BoundedGrid(rows, cols)	creates a new Grid[rows,cols]
getNumCols()	gets the # of cols for this grid
getNumRows()	gets the # of rows for this grid

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

BoundedGrid is a matrix of object references.

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
{}
```


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

BoundedGrid has a limited number of storage locations.

open bgridone.java

BoundedGrid

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

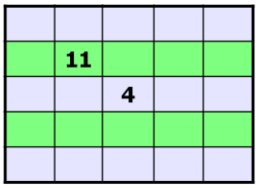
```
import info.gridworld.grid.BoundedGrid;
```

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
```



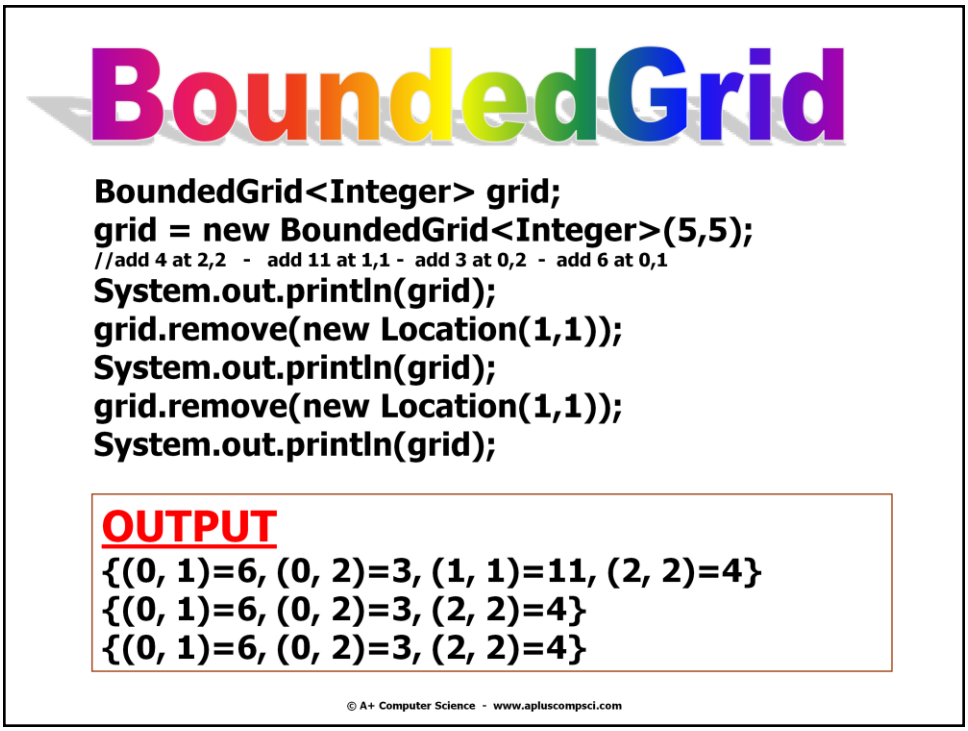
BoundedGrid is a grid that has a set number of rows and columns.

BoundedGrid has a limited number of storage locations.

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

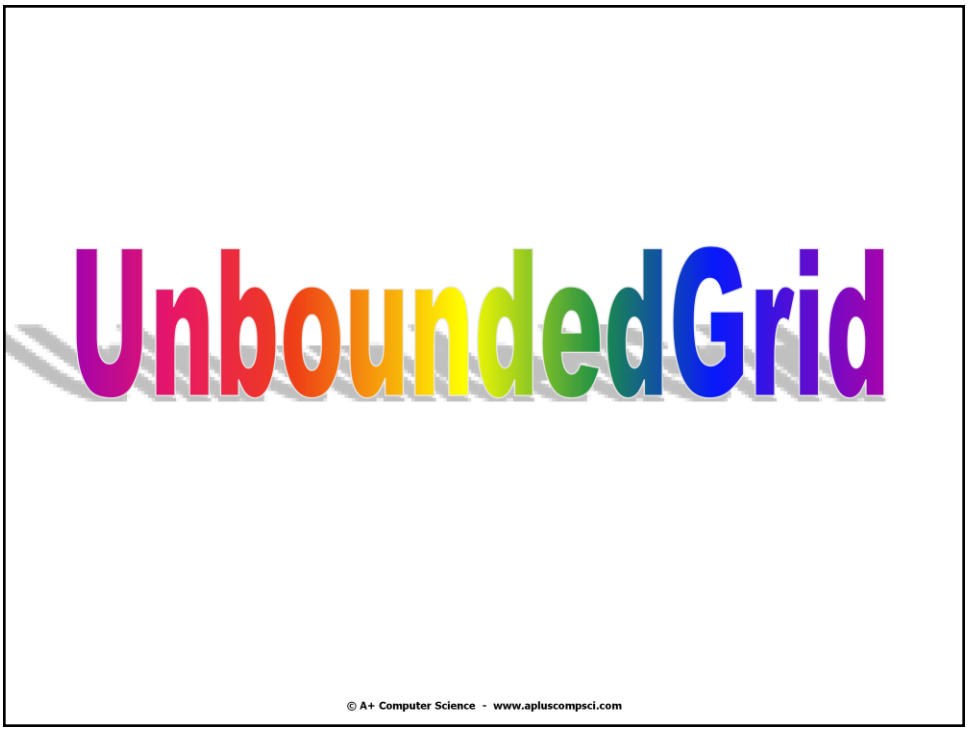
BoundedGrid has a bottom right storage location of (length-1,length-1).

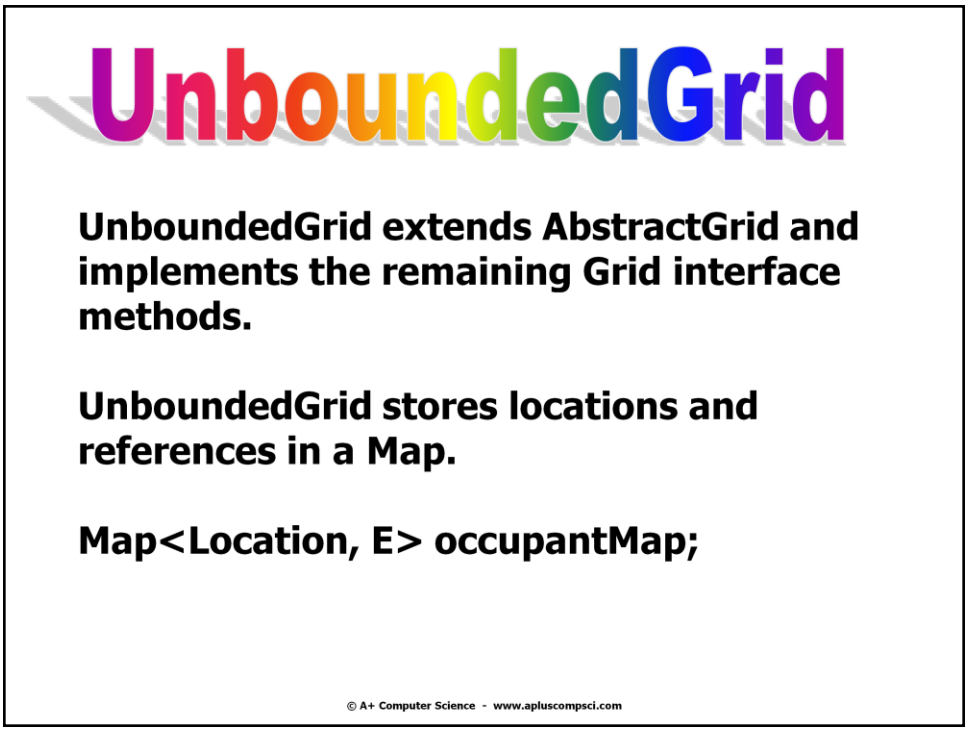
The isValid method will determine if a provided location is within the bounds of the provided grid.



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


UnboundedGrid
© A+ Computer Science - 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

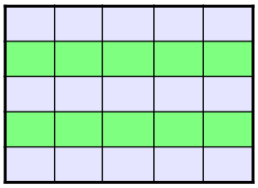
Name	Use
UnboundedGrid()	creates an empty grid
getNumCols()	returns -1
getNumRows()	returns -1

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

UnboundedGrid stores loc and objs in a Map.

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
{}
```


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

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

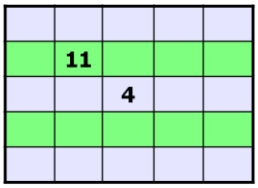
```
import info.gridworld.grid.UnboundedGrid;
```

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
```



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

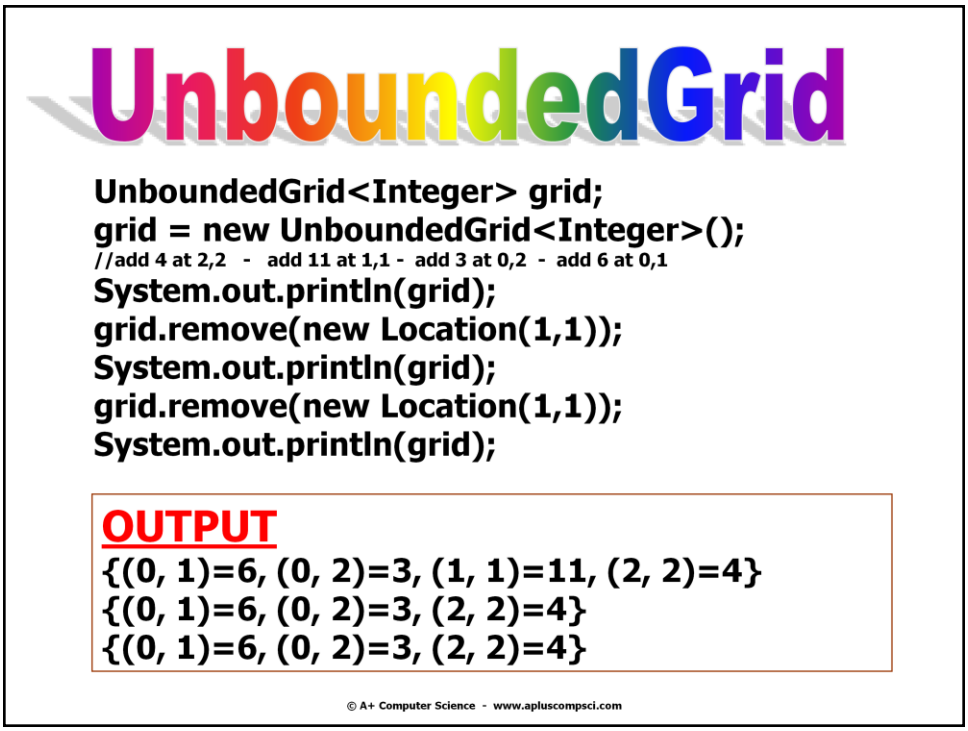
UnboundedGrid has an unlimited number of storage locations.

UnboundedGrid has no set top left storage location.

UnboundedGrid has no set bottom right storage location.

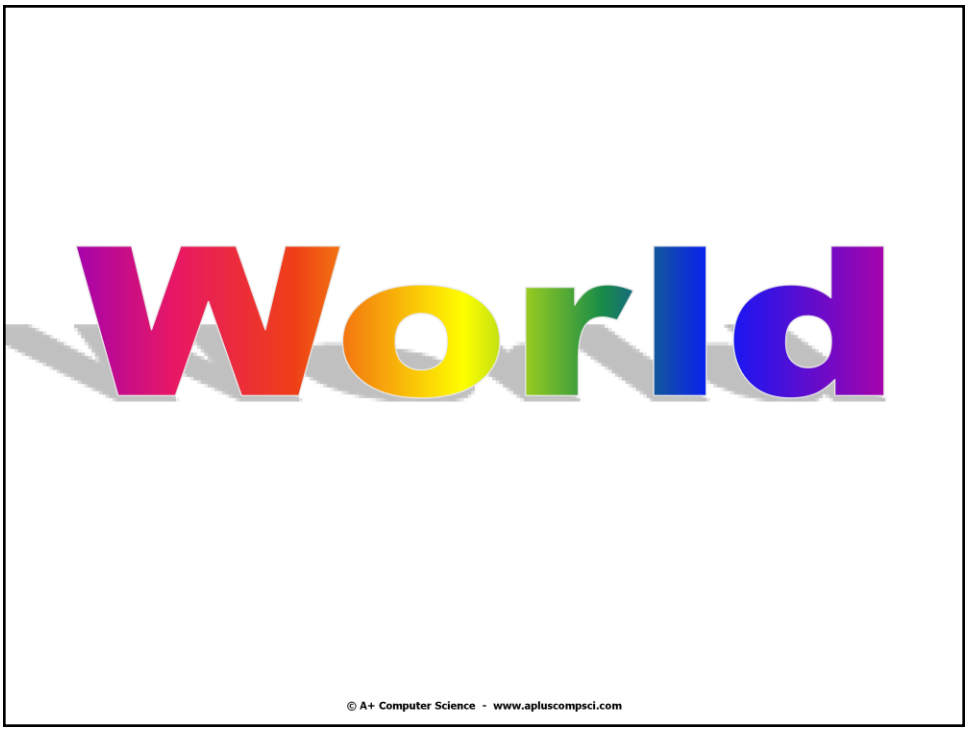
The isValid method will determine if a provided location is within the bounds of the provided grid.

All locations are valid for an unbounded grid.



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


World
© A+ Computer Science - www.apluscompsci.com

World

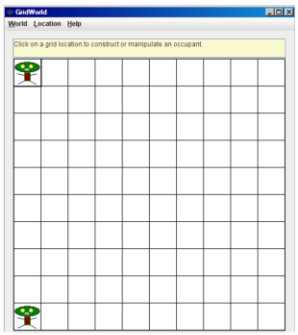
frequently used methods

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

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

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();
```



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

open worldone.java

World

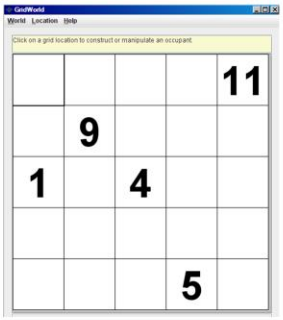
frequently used methods

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

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

World

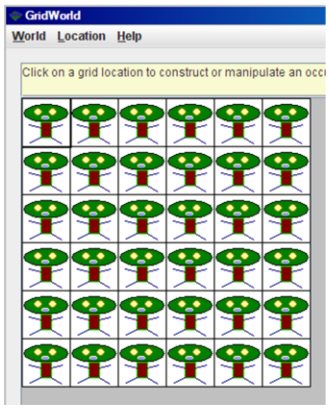
```
BoundedGrid<Integer> grid;
grid = new BoundedGrid<Integer>(5,5);
grid.put(new Location(2,2),4);
grid.put(new Location(1,1),9);
grid.put(new Location(0,4),11);
grid.put(new Location(4,3),5);
grid.put(new Location(2,0),1);

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



World 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));
  }
}
```



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

open worldtwo.java worldthree.java

Start work on Grid Labs