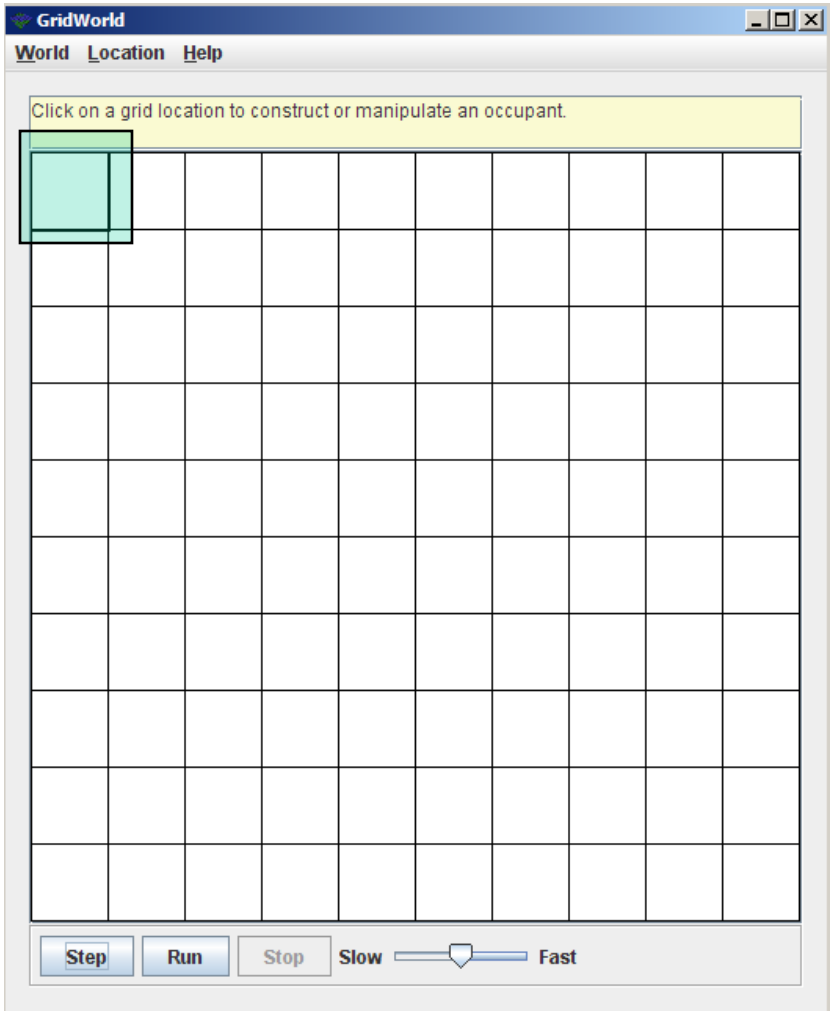
# GridWorld

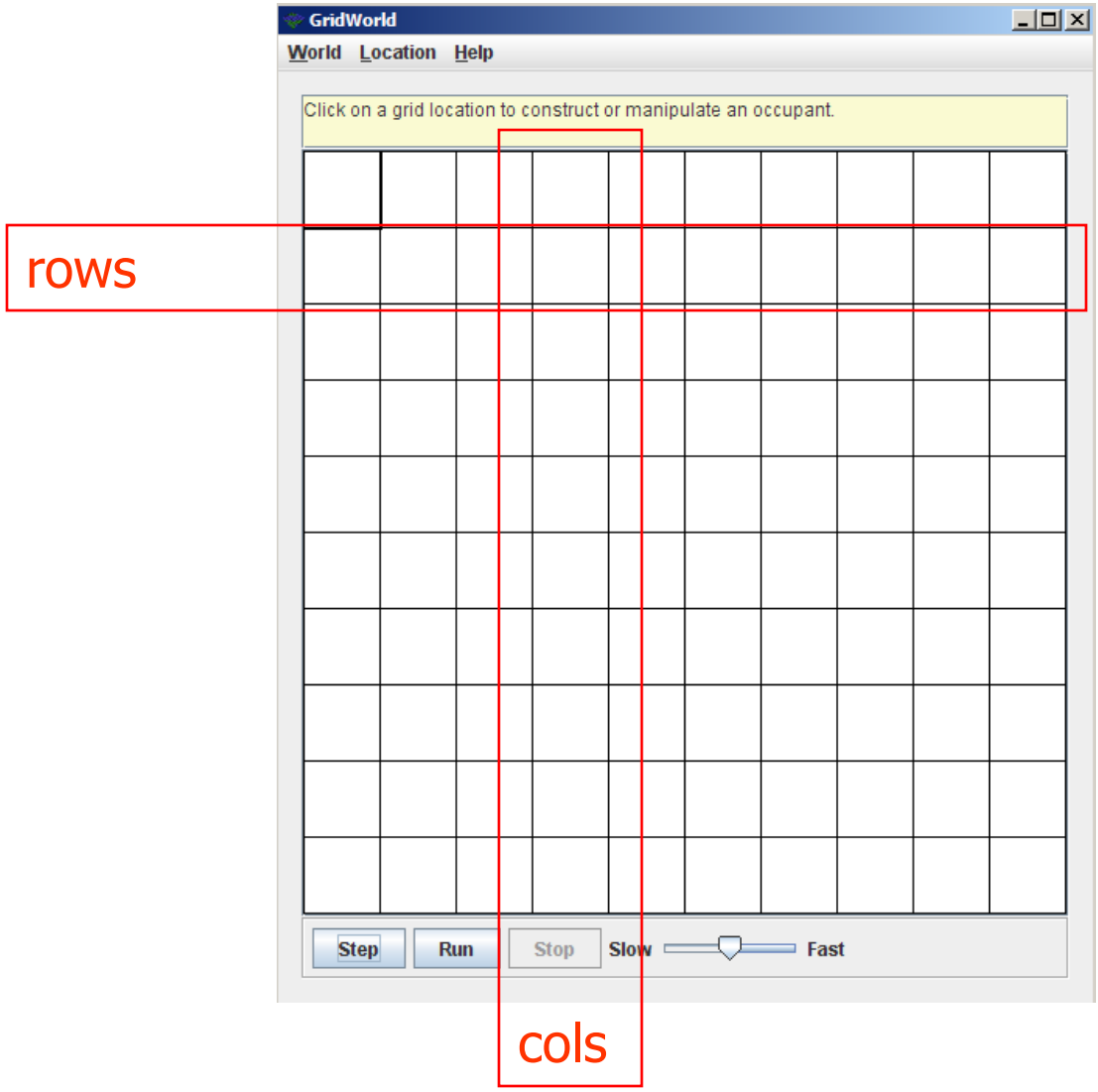
# What is GridWorld?




**Row = 0**

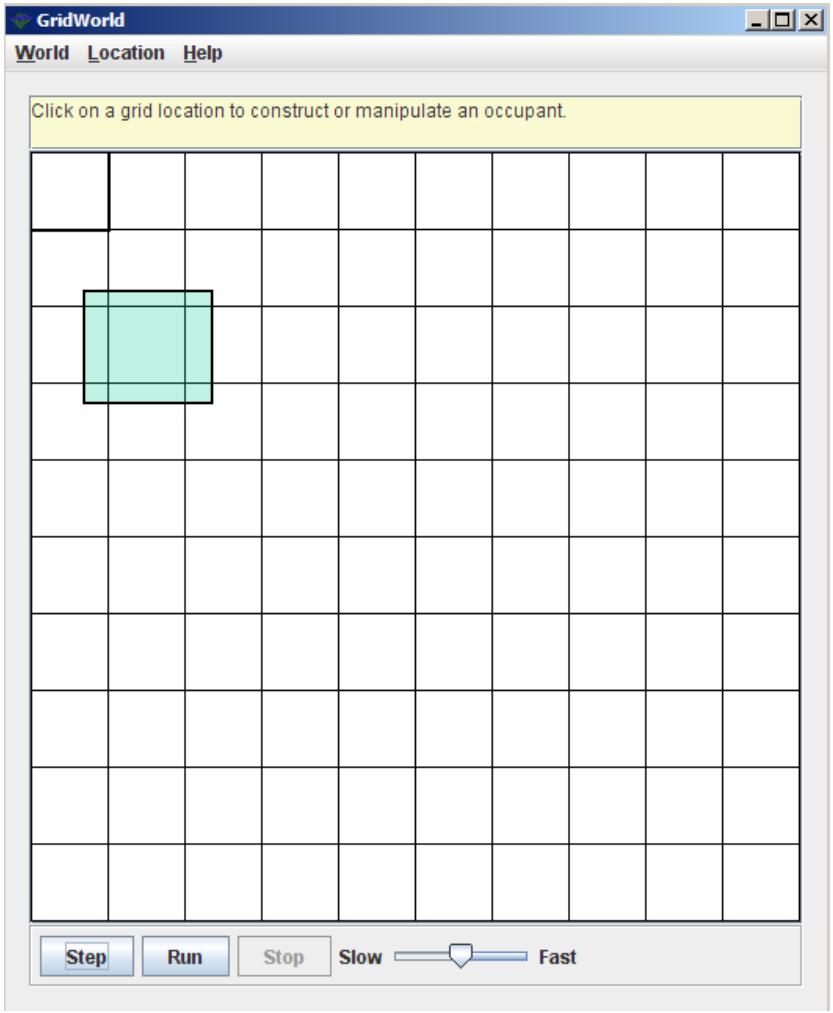
**Column = 0**

# What is GridWorld?




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

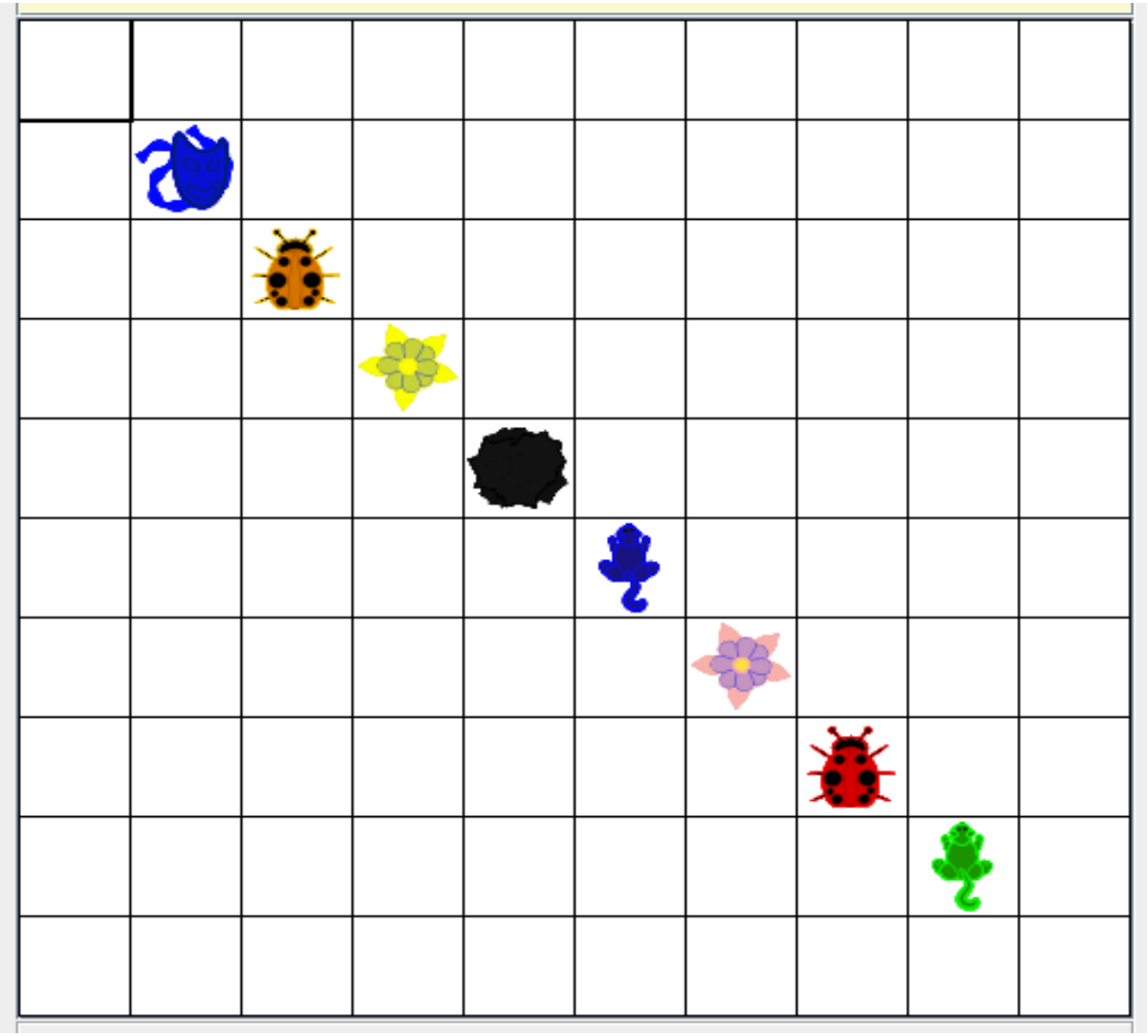
# What is GridWorld?




**Row = 2**

**Column = 1**

# What is GridWorld?



# Location

# Location

## frequently used methods

| Name | Use |
|---|---|
| Location(row, col) | creates a new row,col Location |
| getCol() | gets the column value for this location |
| getRow() | gets the row value for this location |

```
import info.gridworld.grid.Location;
```

# Location

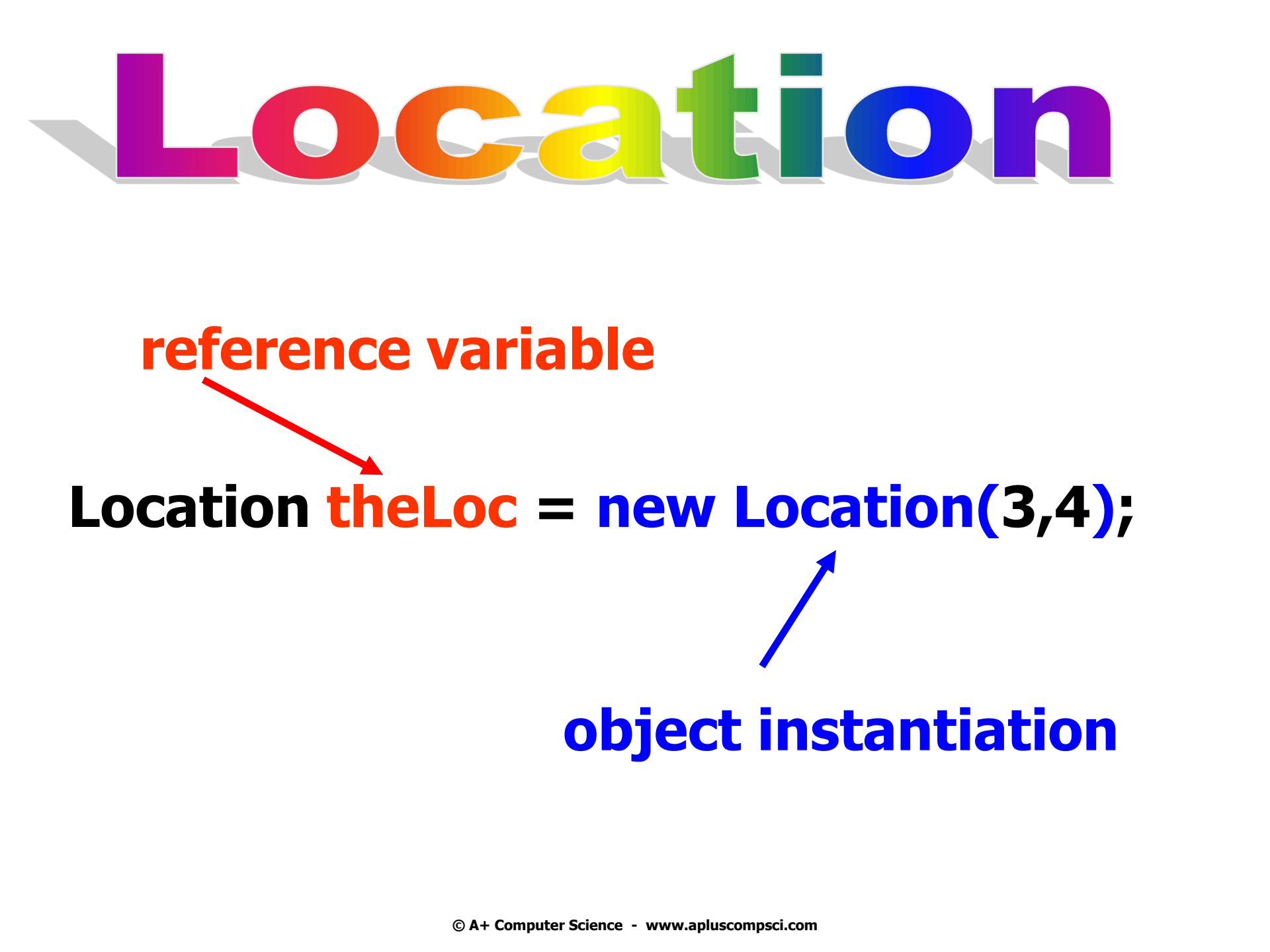
```
Location locTwo = new Location(3,5);
System.out.println(locTwo);

System.out.println(locTwo.getRow());
System.out.println(locTwo.getCol());
```

**OUTPUT**

```
(3, 5)
3
5
```

The Location class stores row and column information.

# Location



# Location

reference

command / method

```
theLoc.getRow();
```

# open locationone.java

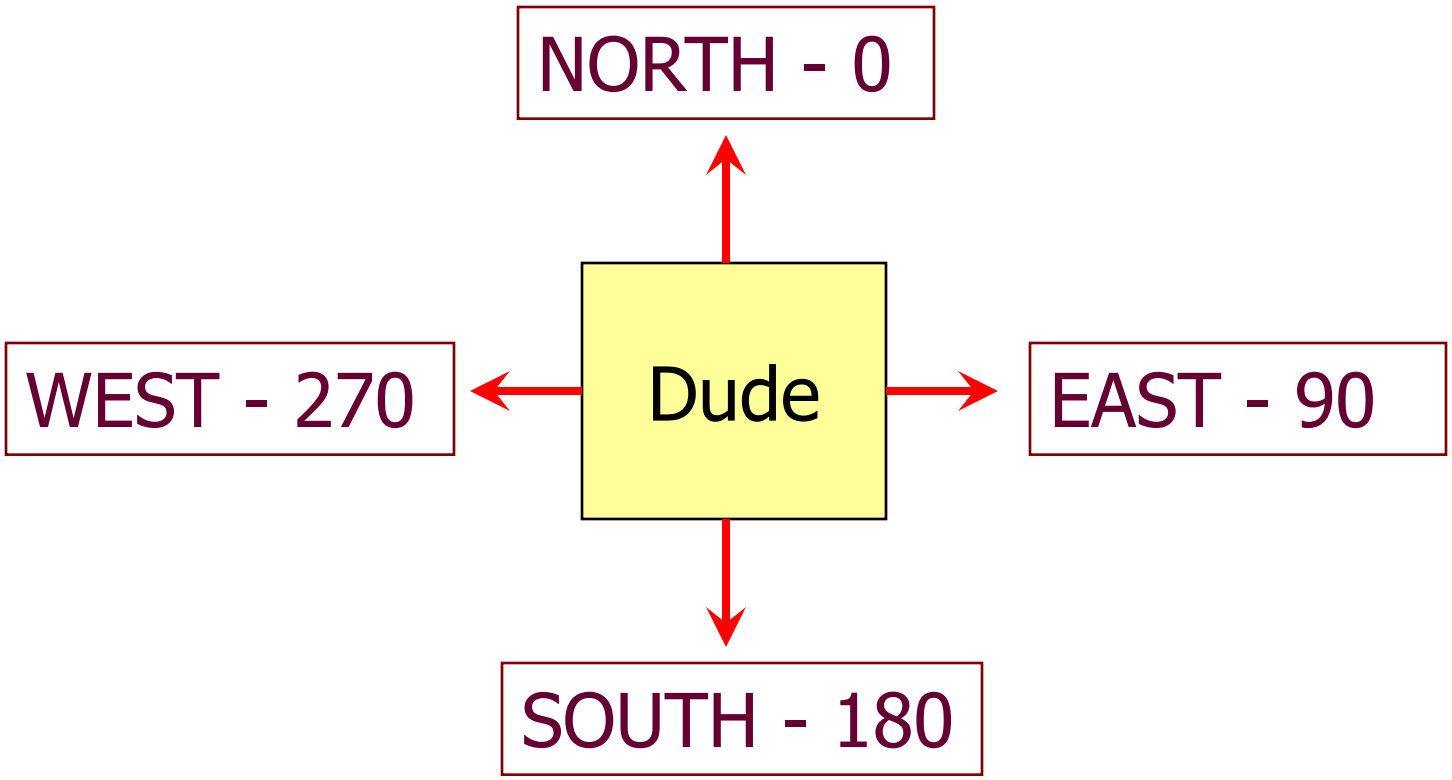
# Location

## frequently used fields

| Name | Use |
|---|---|
| NORTH | indicates going north – value of 0 |
| SOUTH | indicates going south – value of 180 |
| EAST | indicates going east – value of 90 |
| WEST | indicates going west – value of 270 |

```
import info.gridworld.grid.Location;
```

# Location



# Location

```
System.out.println(Location.NORTH);
System.out.println(Location.SOUTH);
System.out.println(Location.EAST);
System.out.println(Location.WEST);
```

**OUTPUT**

```
0
180
90
270
```

# open locationtwo.java

# Location

## frequently used methods

| Name | Use |
|---|---|
| getAdjacentLocation(dir) | get nearest loc in the dir |
| getDirectionToward(dest) | gives dir needed to reach dest |
| compareTo(thang) | compares this to thang |
| equals(thang) | test equality of this and thang |

```
import info.gridworld.grid.Location;
```

# Location

```
Location locOne = new Location(2,1);
Location locTwo = new Location(1,3);

out.println(locOne.getAdjacentLocation(Location.NORTH));
out.println(locOne.getAdjacentLocation(Location.SOUTH));
out.println(locOne.getAdjacentLocation(Location.EAST));
out.println(locOne.getAdjacentLocation(Location.WEST));

out.println(locOne.getDirectionToward(locTwo));
```

| | | | | |
|---|---|---|---|---|
| 0,0 | 0,1 | 0,2 | 0,3 | 0,4 |
| 1,0 | 1,1 | 1,2 | 1,3 | 1,4 |
| 2,0 | 2,1 | 2,2 | 2,3 | 2,4 |
| 3,0 | 3,1 | 3,2 | 3,3 | 3,4 |

**OUTPUT**

```
(1, 1)
(3, 1)
(2, 2)
(2, 0)
45
```

# open locationthree.java

# Location

```
Location locOne = new Location(9,1);
Location locTwo = new Location(3,6);

System.out.println(locOne.equals(locTwo));
System.out.println(locOne.compareTo(locTwo));
System.out.println(locTwo.compareTo(locOne));
```

**OUTPUT**

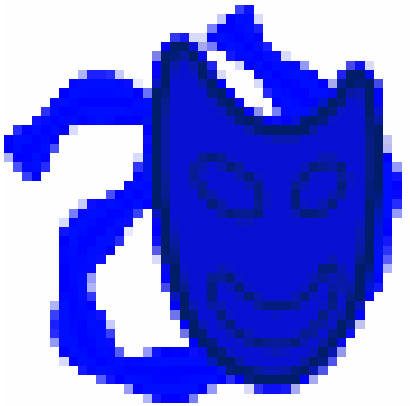
```
false
1
-1
```

# open locationfour.java

# Actor

**Actor is the basic object from which all other GridWorld actors will be built.**

**Each of the new actors created will extend the original actor class.**



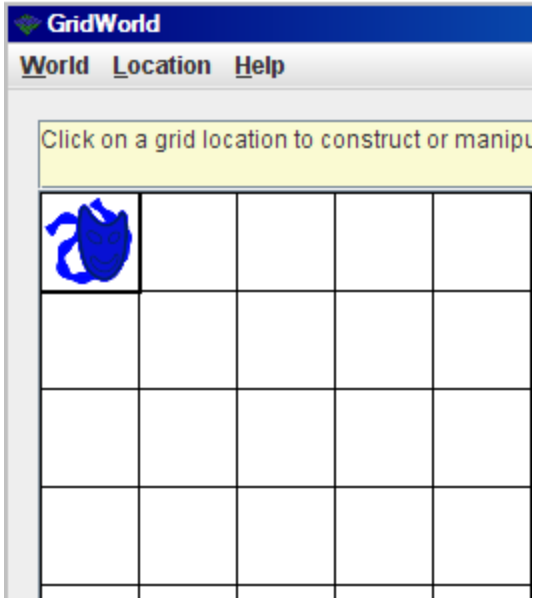
# Actor

## frequently used methods

| Name | Use |
|---|---|
| Actor() | creates new blue north bound actor |
| act() | reverses the direction for actor |
| getColor() | gets the actor's color |
| getDirection() | gets the actor's direction |
| getLocation() | gets the actor's location |
| setColor(col) | sets the actor's color to col |
| setDirection(dir) | sets the actor's direction to dir |
| moveTo(loc) | moves the actor to new location loc |

# Actor

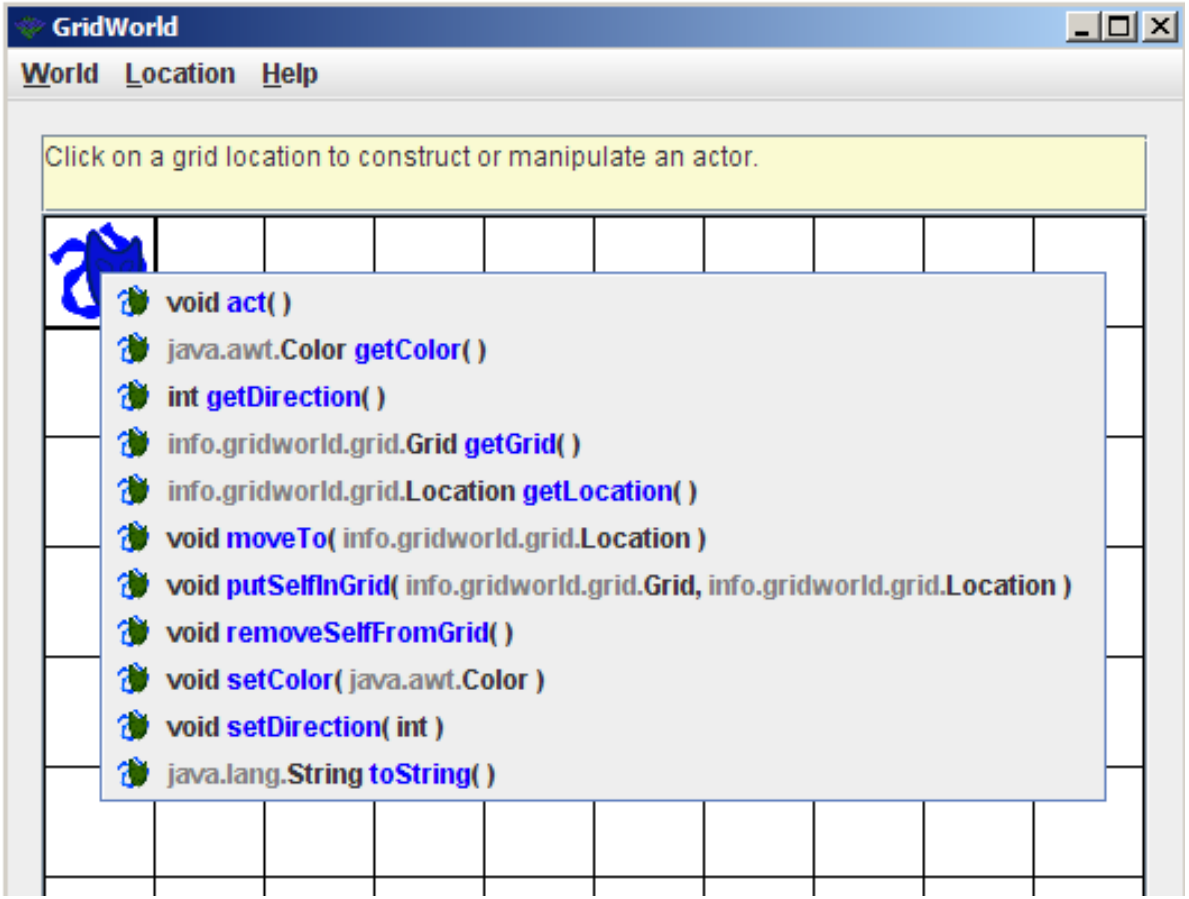
```
ActorWorld world = new ActorWorld();
Actor dude = new Actor();
world.add(new Location(0,0), dude);
world.show();
```




**What happens if you click on the actor?**

# Actor

**When you click on an actor, a list of methods is shown.**

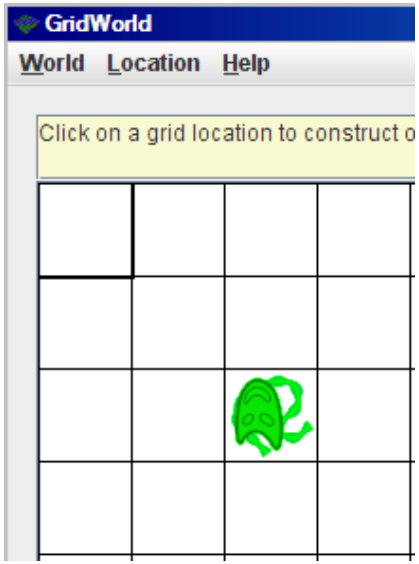


# open actorone.java

# Actor

```
ActorWorld world = new ActorWorld();
Actor dude = new Actor();
dude.setColor(Color.GREEN);
dude.setDirection(Location.SOUTH);
Location loc = new Location(2,2);
world.add(loc, dude);
world.show();
```

**What happens if you click on an empty location?**

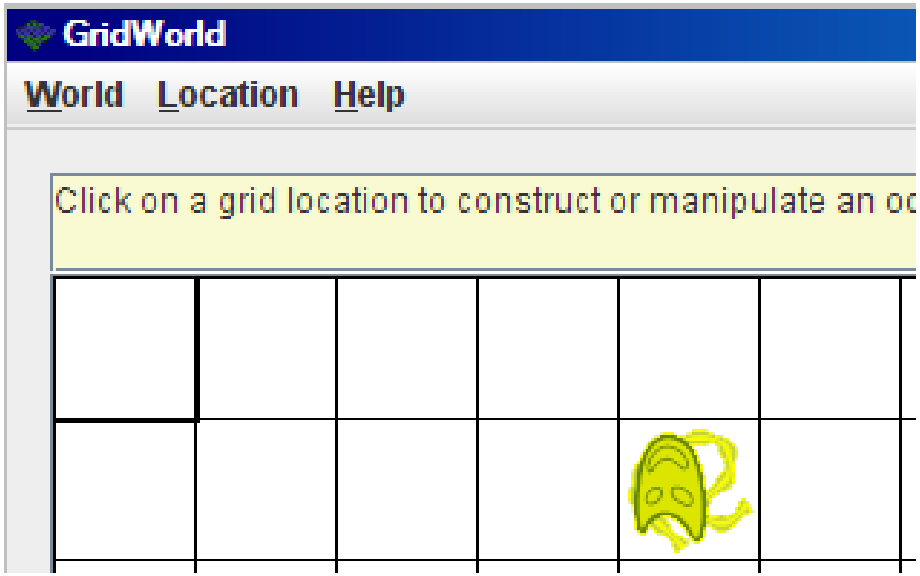


# open actortwo.java

# Actor

**What does an Actor do when its act() method is called ?**

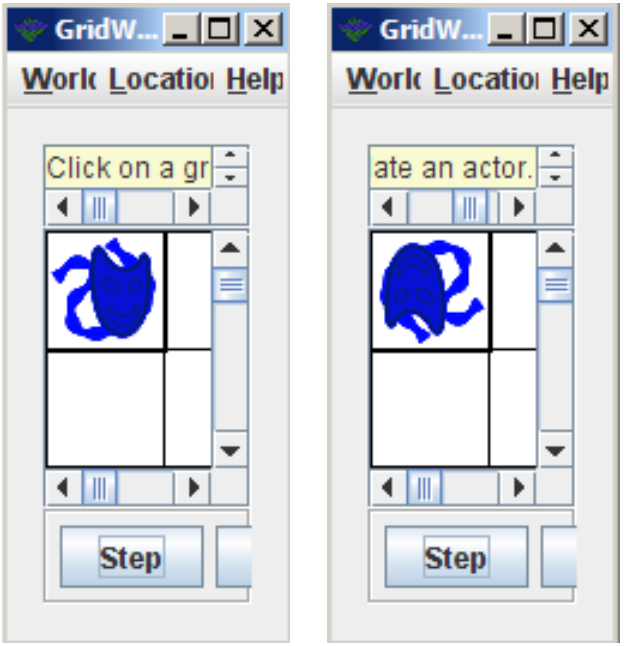
**How does the act() method get called?**



# act

**The actor act method calls methods to make the actor do something.**

**Each time the act method for the default actor is called, the actor changes to the opposite direction.**



# moveTo

**The moveTo method is essentially a setLocation method.**

**The moveTo method is used to make an actor move to a new location.**

```
chucky.moveTo(new Location(3,3));
```

# Actor

```
ActorWorld world = new ActorWorld();
Actor dude = new Actor();
dude.setColor(Color.ORANGE);
dude.setDirection(Location.WEST);
world.add(new Location(1,2), dude);
dude.moveTo(new Location(6,7));
dude.moveTo(new Location(8,7));
world.show();
```

**Where does dude show up?**

# open actorthree.java

# Extending Actor

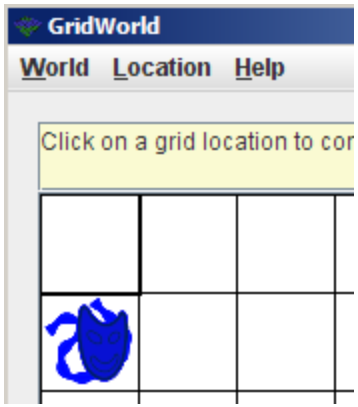
# Extending Actor

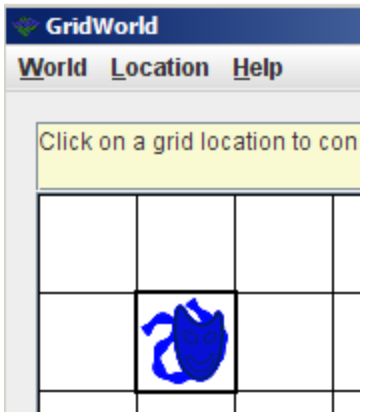
**To make a new actor, you must extend the Actor class and override the act method to give the new actor its own unique behavior.**

**What would have to be done to make a new actor that only moved to the right?**

# Extending Actor

```
public class SideWaysActor extends Actor
{
  public void act()
  {
    //move to the right

  }
}
```



# open sidewaysactor.java sidewaysactorrunner.java

# Start work on Actor Labs

# Actor

## frequently used methods

| Name | Use |
|---|---|
| putSelfInGrid(grid, loc) | put this actor in grid at loc |
| removeSelfFromGrid() | takes this actor out of the grid |
| getGrid() | gets the grid which contains this actor |
| toString() | gets actor data as a String |

```
import info.gridworld.actor.Actor;
```

# putSelfInGrid

The putSelfInGrid method puts an actor into a grid at a specified location.

The world add method calls putSelfInGrid when adding an actor to the grid.

world.add(loc, chucky);

chucky.putSelfInGrid(grid, loc);

# removeSelfFromGrid

**The removeSelfFromGrid method removes an actor from its grid.**

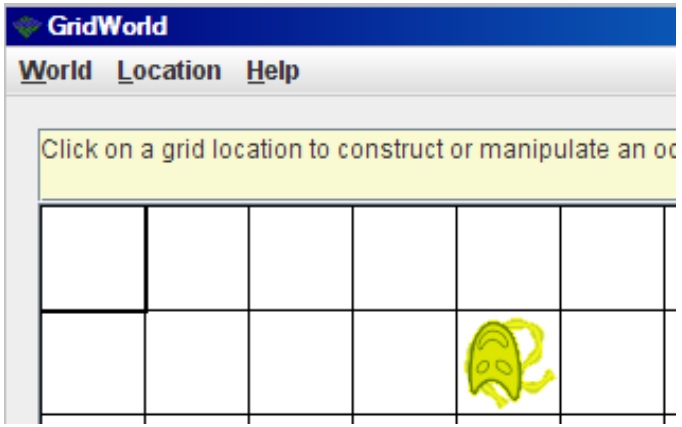
**When it is time to do away with an actor, call removeSelfFromGrid and the actor will disappear.**

```
chucky.removeSelfFromGrid();
```

# Actor

```
ActorWorld world = new ActorWorld();
Actor dude = new Actor();
dude.setColor(Color.YELLOW);
dude.setDirection(Location.SOUTH);
Location loc = new Location(1,4);
dude.putSelfInGrid(world.getGrid(),loc);
world.show();
```

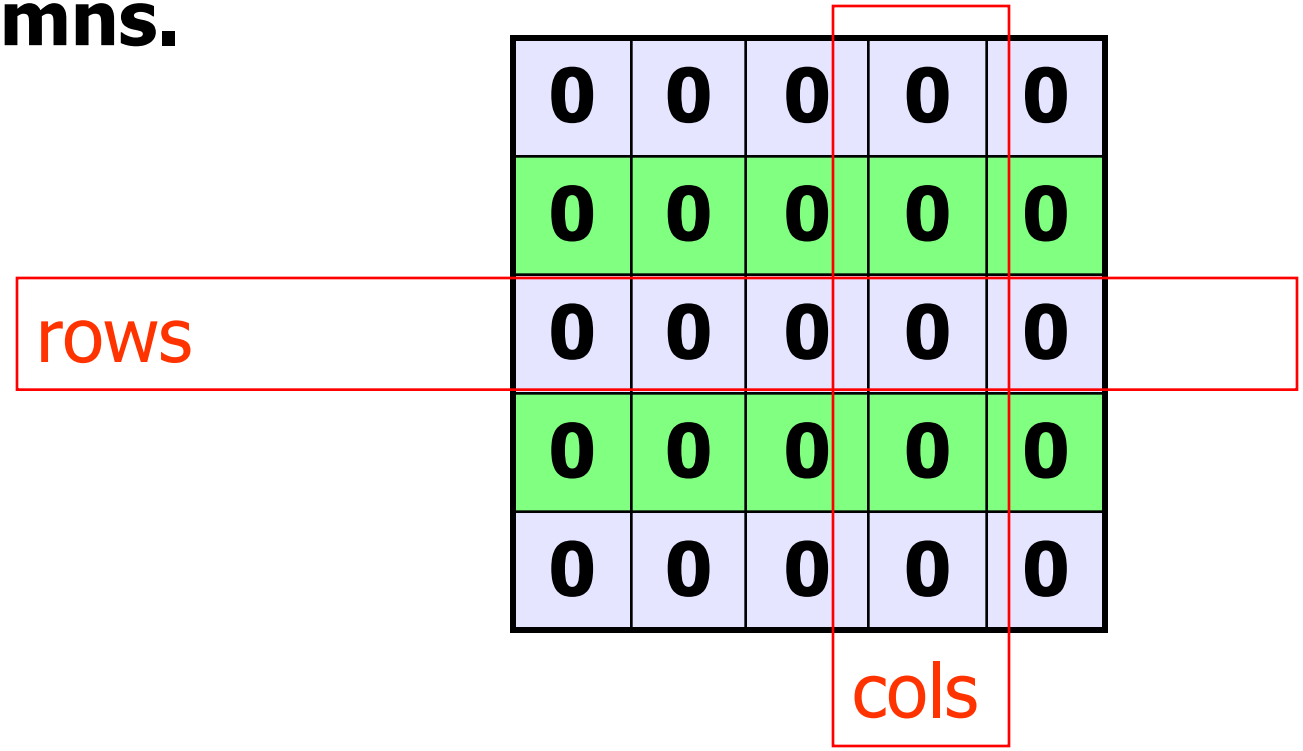
**What happens when you hit the run button?**



# open actorfour.java

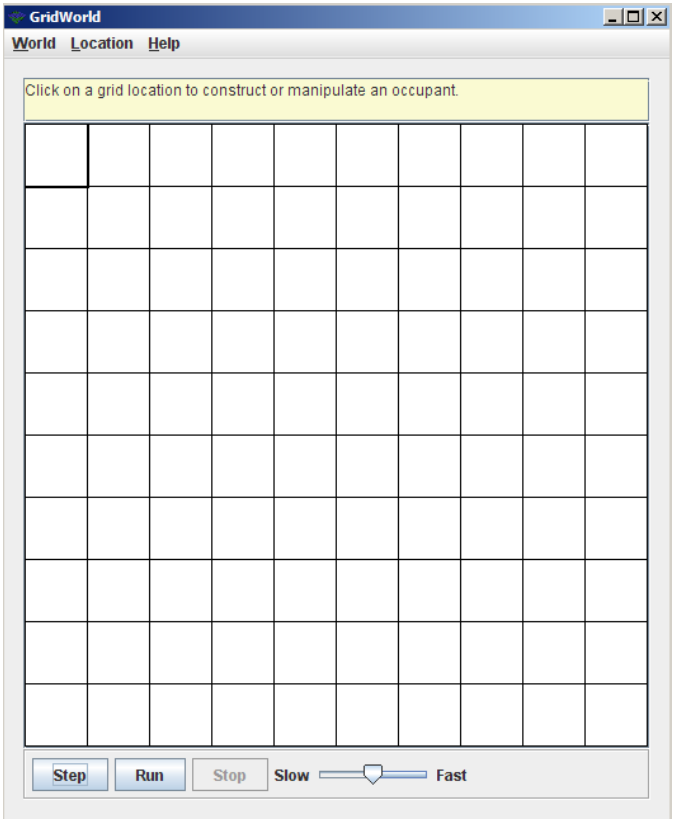
# Grid

**All of the actors are stored in a grid. A grid has rows and columns.**



# Grid

**World is used to show the grid graphically.**



# Grid
## frequently used methods

| Name | Use |
|---|---|
| get(loc) | returns the object at location loc |
| getNumCols() | gets the # of cols for this grid |
| getNumRows() | gets the # of rows for this grid |
| 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;
```

# getGrid

**The getGrid method returns the grid housing this actor.**

```
Grid<Actor> grid = chucky.getGrid();
```

# open actorfive.java

# Continue work on Actor Labs