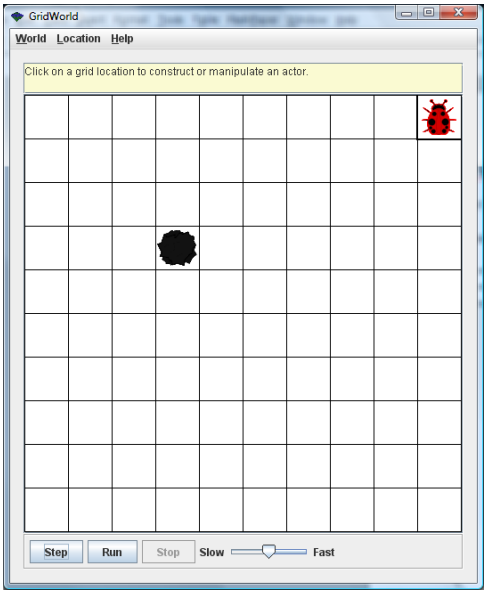
Unit 8. GridWorld Questions.



Part 1.

1. What is the location of the bug?

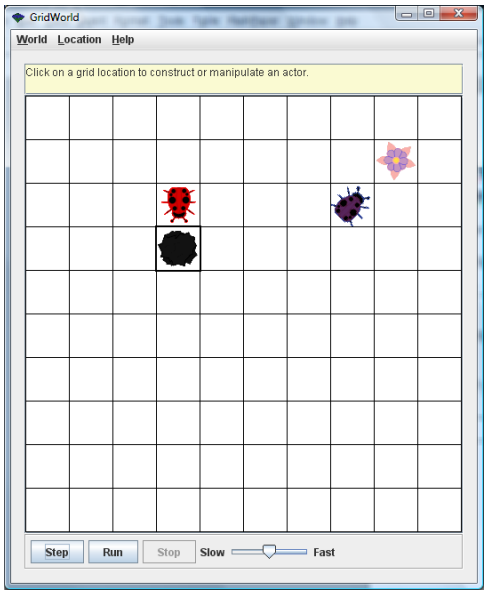
row _____, column _____

2. What is the direction of the bug? _____

3. If you click on the Step button once, what happens?

_____

_____



4. What is the location of the bug on the left?

row _____, column _____

5. What is the direction of the bug on the left? _____

6. What is the direction of the bug on the right? _____

7. If you click on the Step button once, what does the Bug on the left do? _____

_____

What does the Bug on the right do (be specific)?

_____

8. What does a Rock do when you call its act method? _____

9. What does a Flower do when you call its act method? _____

Part 2. After running BoxBugRunner and examining the code, answer the following questions.

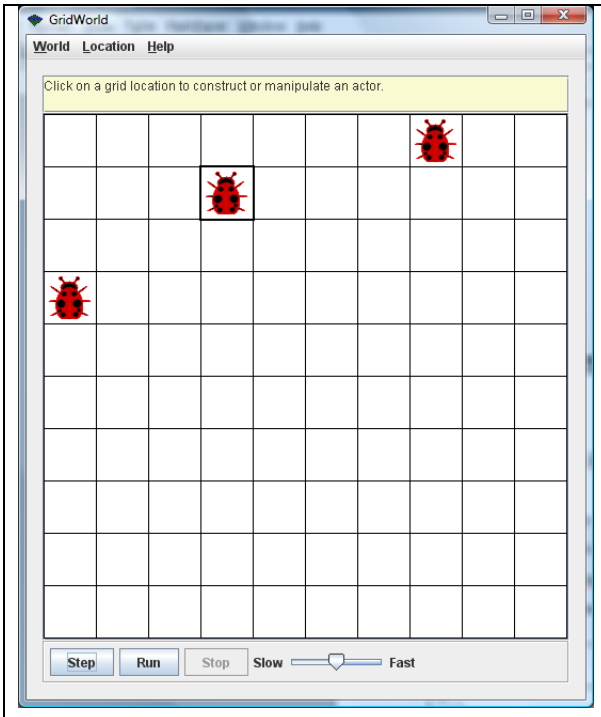
10. When a BoxBug is constructed, what direction is it facing? _____

11. When a BoxBug is constructed, what color is it? _____

12. Modify the BoxBug class so that when an object is constructed, it will be green by default.

13. Under what circumstances does a BoxBug <u>not</u> trace out a square box? _____

_____



To the left are three SomeBugs.

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;

public class SomeBug extends Bug{
        public void act() {
                super.act();
                super.act();
        }
}
```

14. After the Step button is clicked, what does the leftmost SomeBug do? _____

_____

What does the middle SomeBug do? _____

_____

What does the rightmost SomeBug do? _____

_____

Part 3.

15. What is the value of n12? _____
16. What is the value of n21? _____
	Location loc1 = new Location (4, 5);
	Location loc2 = new Location (4, 4);
	int n12 = loc1.getDirectionToward(loc2);
	int n21 = loc2.getDirectionToward(loc1);

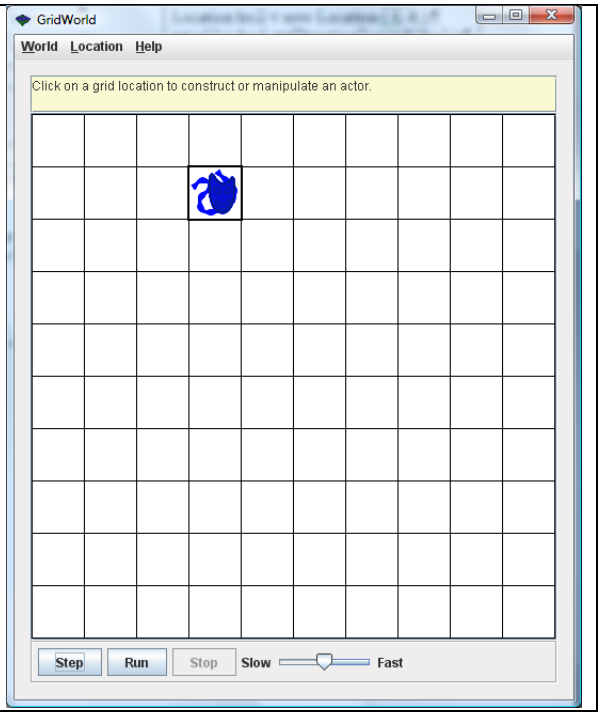
17. What is displayed?	Location loc3 = new Location (800, -3); Location loc4 = loc3.getAdjacentLocation(Location.EAST); System.out.println(loc4.getRow() + ", " + loc4.getCol());

18. Where is the setDirection method defined?

19. If an ABug is added to the grid, what (if anything) does it do when the Step button is clicked?

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;

public class ABug extends Bug{

        public ABug() {
                setDirection( 270 );
        }
}
```

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;

public class Actor_A extends Actor{

    public void act() {
        Grid<Actor> gr = getGrid();
        if (gr == null)
            return;

        int row = getLocation().getRow();
        moveTo( new Location( row, 0 ) );

        int n = gr.getNumCols();
        Location loc = new Location( row, n-1 );
        Rock r = new Rock(getColor());
        r.putSelfInGrid(gr, loc);
    }
}
```



Check the information in the appendix before answering problems 20 and 21.

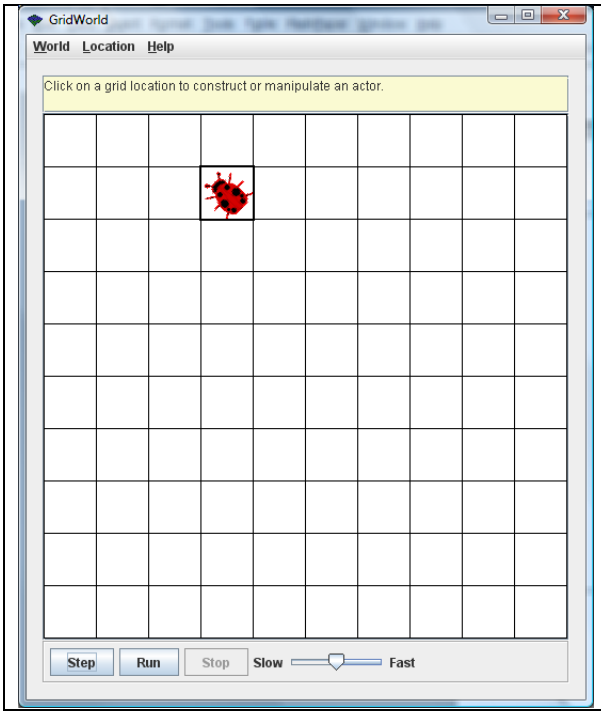
20. The grid above has an Actor_A in it. What happens when the Step button is clicked?

_____

21. What happens if the Step button is clicked a second time? _____

_____

_____




```
import info.gridworld.actor.*;
import info.gridworld.grid.*;

public class Buggy extends Bug{
        public Buggy(){
                setDirection( -45 );
        }

        public void turn() {
                setDirection(getDirection() +
                        Location.LEFT);
        }
}
```

The grid contains one Buggy.

22. After you click the Step button once, Buggy will be at row _____, column _____, and have a direction of _____ (give a number).

23. After you click the Step button a second time, Buggy will be at row _____, column _____, and have a direction of _____ (give a number).

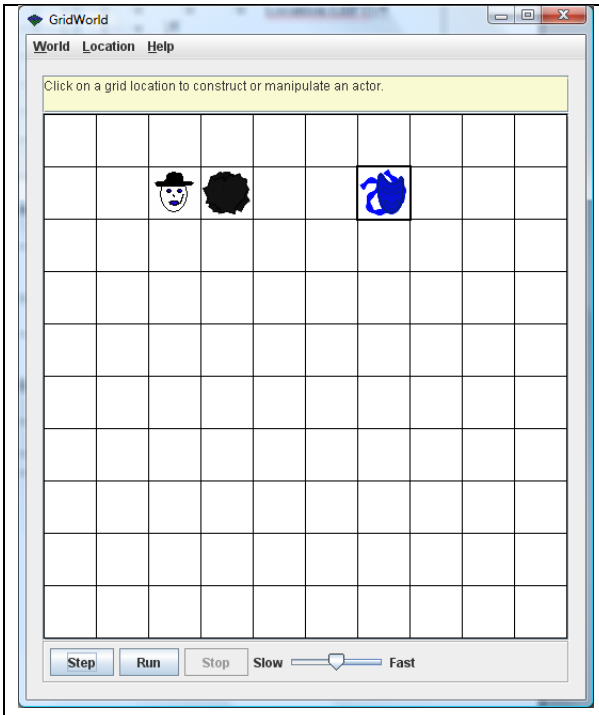
24. If Buggy moves, does it leave a flower behind? Explain. _____

_____

25. This class compiles and may run for awhile but eventually you get a runtime error. Why?

26. What method of the Grid interface could be used to prevent this error from happening?

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;
import java.util.*;

public class Bad_Actor extends Actor{
    public void act() {
        Grid<Actor> gr = getGrid();
        if (gr == null)
            return;

        Location loc1 = getLocation();
        int dir = getDirection();
        Location loc2 = loc1.getAdjacentLocation( dir );
        moveTo( loc2 );
    }
}
```



The above grid contains (from left to right) a Mean_Actor, a Rock, and a (regular) Actor.

```
import info.gridworld.actor.*;
import info.gridworld.grid.*;
import java.util.*;

public class Mean_Actor extends Actor{
  public void act() {
      Grid<Actor> gr = getGrid();
      if (gr == null)
          return;

      ArrayList<Location> a =
gr.getOccupiedAdjacentLocations(getLocation());

if ( a.size() > 0 ){
          int n = (int)( a.size() * Math.random() );
          Location loc = a.get(n);
          moveTo( loc );
      }
  }
}
```

27. After you click the Step button once, the Mean_Actor will be at

    row _____ and column _____.

28. After you click the Step button a second time, the Mean_Actor will be at

    row _____ and column _____.

29. If a Mean_Actor moves, does it leave a flower behind? Explain. _____

_____

Part 4 (Critters)

30. Suppose you were going to write a FrontCritter that only eats something directly in front of it (and they cannot be a Critter or a Rock). It moves just like a regular Critter. What method would be the best method to override?

    a) getActors
    b) processActors
    c) getMoveLocations
    d) selectMoveLocation
    e) makeMove

40. After a Critter moves from Location(2, 4) to Location(3, 5), what will its direction be? ___

After the Step button is clicked once ...

41. Where will the top Critter be? _____

_____

_____

42. Where will the bottom Critter be? _____

_____

_____

43. What happens to the 7 Rocks? _____

_____

_____

44. What happens to the 3 Flowers? _____

_____

_____

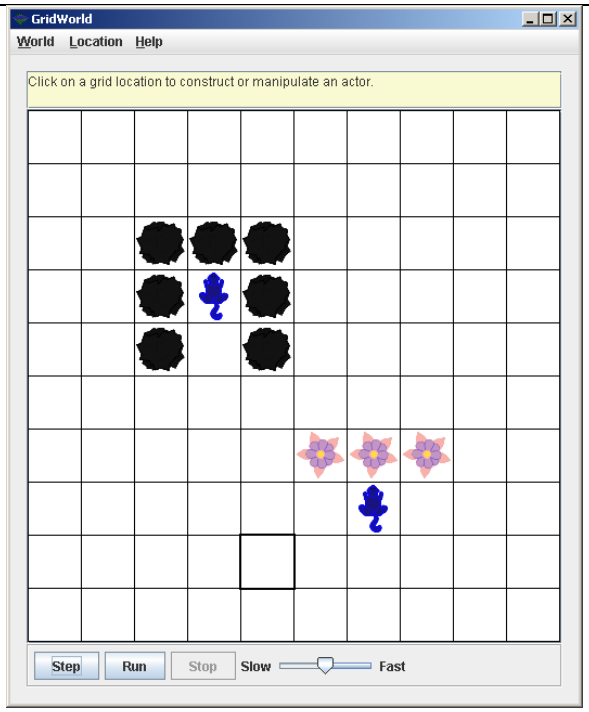


Figure 1. There are 7 black rocks, 2 blue Critters, and 3 pink Flowers.

45. Will the MadCritter class run without errors? If Yes, how does a MadCritter act differently than a regular Critter? If no, what is the problem?

_____

_____

_____

_____

```
public class MadCritter extends Critter {
    public ArrayList<Actor> getActors() {
        return null;
    }
}
```

46.How does a Bad Critter move?

_____

_____

47. Will a BadCritter cause a program to eventually crash?

_____

```
// this compiles

public class BadCritter extends Critter {
public ArrayList<Location> getMoveLocations(){
    ArrayList<Location> list = new ArrayList<Location>();
    Location loc = getLocation();
    list.add( new Location( loc.getRow(), loc.getCol() + 1 )
);
    return list;
    }
}
```

48. Suppose you were going to write a RandomCritter that behaves like a regular critter except that it can move to any random empty location in the grid. What method would be the best method to override?

a) getMoveLocations
b) selectMoveLocation
c) makeMove
d) You would need to override more than one of these methods.

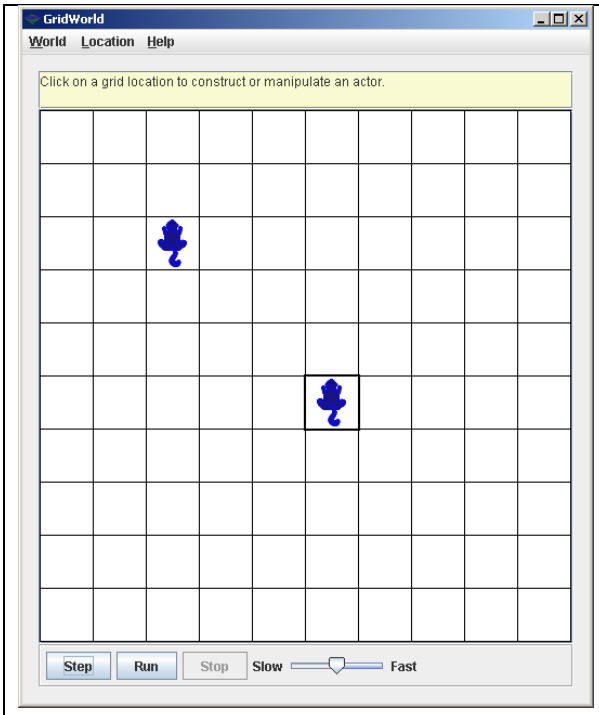


Figure 2.

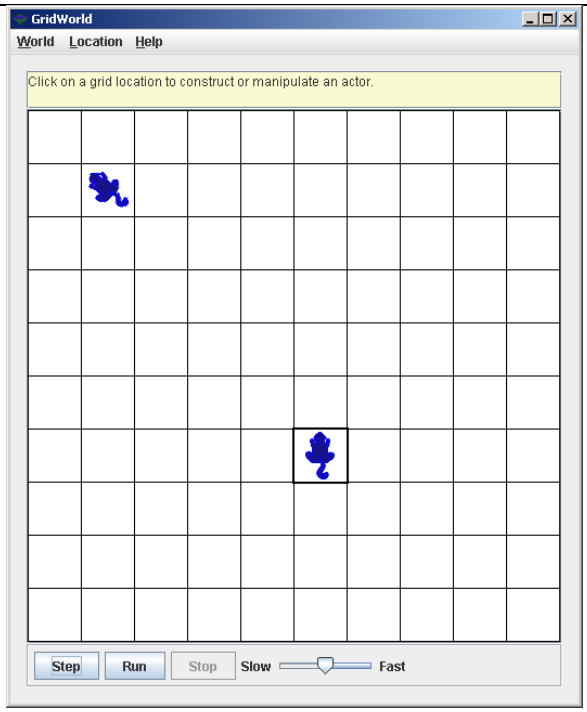


Figure 3

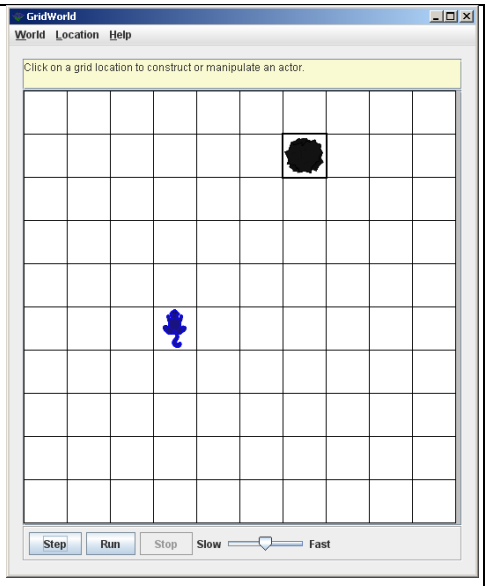
49. Figure 2 shows two critters: one is regular critter and one is a chameleon critter. I changed the icons so they look the same. Figure 3 shows the same grid after one step. Select the TRUE statement.

a) The top critter must be a chameleon critter.
b) The bottom critter must be a chameleon critter.
c) Either one could be a chameleon critter.
d) There's been an awful mistake, neither one could be a chameleon critter.

50. Select the TRUE statement(s). An OddCritter ...
a) eats only 1 neighboring actor at a time (provided that there are neighbors).
b) can eat rocks, critters, any actor.
c) moves just like a critter.
d) eventually throws a run-time error.
	// This compiles
	public class OddCritter extends Critter {
	public void processActors(ArrayList<Actor> actors){
	if (actors.size() == 0) return;
	int n = (int) (actors.size() * Math.random());
	actors.get(n).removeSelfFromGrid();
	}
	}

51. A BitterCritter ...		// This compiles
a) turns every time it moves.	T F	public class BitterCritter extends Critter{
b) acts <u>exactly</u> like a critter.	T F	public void turn(){
c) eventually throws a run-time error.	T F	int dir = getDirection();
d) turns only if it eats an actor.	T F	setDirection(dir + 90);
		}
		}

```
public class SitterCritter extends Critter  {
private Location x;

public ArrayList<Actor> getActors() {

    ArrayList<Actor> a = super.getActors();

    Grid<Actor> gr = getGrid();

    x = null;
    ArrayList<Location> b = gr.getOccupiedLocations();
    b.remove( getLocation() );
    if ( b.size() > 0 ){
        int n = (int)( b.size()*Math.random() );
        x = b.get( n );
    }

    return a;
}

public void makeMove( Location loc ) {
    super.makeMove( x );
}
}
```



There is a SitterCritter and a Rock. This compiles and runs (at least for awhile).

52. When the Step button is clicked, what happens to the SitterCritter in the above grid? If it moves, where does it move? If the program crashes, explain.

_____

_____

53. When the Step button is clicked a second time, what happens to the SitterCritter in the above grid? If it moves, where does it move? If the program crashes, explain.

_____

_____

54. What design principle(s) does the SitterCritter violate? _____

_____

_____