Exercises for Chapter 1:

1. In which direction does a Bug always move?

2. Under what circumstances might the Bug be unable to move?

3. When Step is clicked, what happens when the Bug is unable to move?

4. When a Bug moves, what will occupy the cell that it just vacated?

5. How many objects can occupy a Grid cell simultaneously?

6. Can a Rock object be moved? If so, what method would move it?

7. What method sets a new direction for the Bug?

8. Are the Rock objects as displayed, graphically symmetrical in shape?

9. Based on you answer in #7, would it make visual sense for it to be possible to set

a direction for a Rock object? Why?

10. Does a Flower object move with each Step? If so, where? If not, is it possible to

move a Flower object?

11. What does a Bug do when it tries to move and it encounters the edge of the grid
or

a Rock?

12. What happens when a Bug encounters a Flower?

13. What happens when a Rock is moved on top of a Bug?

14. To change the direction of a Bug to west, what parameter would you send to the

setDirection method?

15. What would happen when, instead of turning, an attempt is made to move a
Bug

outside the Grid?

1. The direction it is facing
2. If there is a rock in front of it or it reaches the edge of the grid
3. It turns 45 degrees to the right
4. A flower of the bugs color
5. 1
6. The actor class’ moveTo(Location) method
7. turn() and setDirection(int);
8. no
9. yes, because there are multiple orientations it can possess.
10. No, it can be moved with the actor moveTo(Location) method.
11. It turns 45 degrees to the right.

12. Nothing, it simply moves to the flower’s position replacing it.
13. The bug is removed from the gird.
14. Location.WEST
15. An error is thrown and nothing happens.

Exercises for Chapter 3:

1. What is returned by loc1.equals(loc2)?

2. What is returned by loc1.compareTo(loc2)?

3. What is returned by loc1.compareTo(loc3)?

4. What is returned by loc1.equals(loc4)?

5. What is returned by loc1==loc4?

6. Which of the following are legitimate ways to specify a direction of 45 degrees toward

the upper left? (There may be more than one answer.)

A. 45 B. -45 C. Location.NORTHEAST D. Location.NORTHWEST

E. NORTHWEST F. Location.LEFT + Location.HALF_RIGHT

7. Using Location methods, write code that will store in int i the product of the rows in loc1

and the number of columns in loc4.

8. What is the (row, column) position of loc after the following statement is executed?

Location loc = loc1.getAdjacentLocation(Location.East);

9. What is the (row, column) position of loc after the following statement is executed? Chapter 3-4

Location loc = loc1.getAdjacentLocation(Location.West + Location.RIGHT);

10. How would you determine if the row position of loc1 is even or odd?

11. What is stored in dir by the following?

int dir = loc1.getDirectionToward(loc3);

12. What is stored in dir by the following?

int dir = loc1.getDirectionToward(new Location(13, 4));

1. False
2. 4

3. 2
4. True
5. False
6. B, D, F
7. int i = loc1.getRow() + loc3.getCol();
8. 12, 4
9. 13, 3
10. boolean odd = (loc1.getRow() % 2) == 1;
11. Location.WEST
12. 45

Exercises for Chapter 4:

1. What code will determine if Location (5,13) is on the Grid?

2. Write code that will print the total number of objects in a Grid.

3. What is the difference between the two methods, getOccupiedLocations and

getOccupiedAdjacentLocations?

4. In a Grid of 20 rows by 30 columns, what would be the maximum number of objects in

the ArrayList that getOccupiedLocations could return?

5. How could the number of empty cells in a Grid be determined?

6. Why does Grid not have any code?

7. What classes have full implementation of Grid?

Chapter 4-4

8. What do getRows( ) and getCols( ) return for an UnboundedGrid?

9. The validAdjacentLocations method can have a many as eight elements in its returned

ArrayList object. Using Location constants, list the directions of these eight cells relative

to the specified Location.

10. Several of the methods of the Grid Interface return an ArrayList object. What advantage

does this have over an ordinary array?

11. Suppose an ArrayList object called aryLst is returned by getValidAdjacentLocations( ).

Using an enhanced for-loop, show how to iterate through the Location objects in aryLst and determine if any objects are in the list. Print true if any objects are present; otherwise,

print false.

1. isValid(Locatinon)
2. grid.getNumRows() * grid.getNumCols();
3. The former returns all occupied location on the grid while the latter only returns occupied locations in the 8 adjacent grid locations to the location.
4. 600
5. Subtract the number of the occupied spaces from the total number of spaces.
6. It is an interface to be implemented.
7. UnboundedGrid and BoudnedGrid.
8. -1
9. Location.NORTH, Location.NORTHEAST, Location.EAST, Location.SOUTHEAST, Location.SOUTH, Location.SOUTHWEST, Location.WEST, Location.NORTHWEST
10. It can be of any size and dynamically resize itself.
11. for (Location l : aryLst) System.out.println(true);

Exercises for Chapter 5

1. When an Actor object is instantiated, what color and direction are automatically assigned

by the constructor? Chapter 5-4

2. What would be wrong with using the put method of the Grid class to place an Actor

object in a Grid?

3. What properties does an Actor object have?

4. What is returned if getGrid is called and the Actor object is contained in multiple grids?

(This is a trick question.)

5. What action is taken in the act method of the Actor class?

6. What are three classes that inherit the Actor class?

7. Show code that will move Actor actr to Location (3, 15) only if that Location does not

contain a Rock object. Assume that actr is already in a Grid.

8. After an Actor is removed from a Grid with removeSelfFromGrid, what will getLocation

return?

9. Is it possible to place an Actor at a Grid location already containing another Actor? If not,

will this produce an exception? If possible, what would be the result?

10. Suppose we want to remove an Actor from a Grid. Knowing that a precondition for doing

so is for the Actor to already be in a Grid, demonstrate a safe way to remove Rock rk.

1. java.awt.Color.BLUE and Location.NORTH
2. It bypasses the Actor’s putSelfInGrid method which checks for errors and doesn’t set the acto’s grid field.
3. A grid, location, color, and direction.
4. It can’t contain multiple grids and will throw an IllegalStateException.
5. It turns 180 degrees.
6. Bug, Rock, Flower
7. If(!actr.getGrid().get(new Location(3, 15)) instanceof Rock) actr.moveTo(new Location(3, 15));
8. null
9. Yes it is possible. It removes the former Actor from the grid first unless it is invalid in which case it throws an IllegalStaeException and does nothing.
10. If(rk.getGrid() != null) rk.removeSelfFromGrid();

Exercises for Chapter 6:

1. What is the largest number of objects possible in the ArrayList object returned by

getActors?

2. What are the five other methods called within the act method of the Critter class?

3. What type of objects are contained in the ArrayList object returned by

getMoveLocations?

4. What type objects are not eaten by the processActors method?

5. When creating your own class and overriding the processActors method, must you

necessarily make it “eat” something?

6. Critter has no constructor. What happens in the absence of a constructor when a new

Critter object is instantiated?

7. Is it possible for getEmptyAdjacentLocations to return an invalid Location in its list?

8. Why is the Location selected by selectMoveLocation not predictable?

9. What is returned by selectMoveLocation if there are no Locations in the ArrayList that it

receives?

10. Suppose the ArrayList of possible move Locations (from which an eventual Location is

selected) is obtained in the Critter class using the getMoveLocations. Is the following

code in the makeMove class really necessary? Why?

if(loc = = null)

removeSelfFromGrid( );

11. In the selectMoveLocation method, why is it necessary to cast (Math.random( ) * n) as an

int type?

12. Suppose you create a new class by extending the Critter class. Some of the methods that

you might override in this new class make use of ArrayList. If you override one of those

methods and make use of ArrayList, will it be necessary to do an import? If so, what

should you import?

1. 8
2. getGrid, processActors, getMoveLocations, selectMoveLocation, makeMove.
3. Location
4. Rock and Critter objects.
5. No, you don't *have* to.
6. The Actor constructor is called.
7. No, one of it's checks is the isValid method.

8. It uses Math.random() to select a random location from the list of move locations.
9. The Critter's current location.
10. No, the getMoveLocations method only returns valid locations.
11. To store the result the double returned by Math.random() has to be cast to a double.
12. Yes, java.util.ArrayList;

Exercises for Chapter 7a:

1. How would the code in makeMove be modified so as to make the new direction exactly the opposite from that presently set in this method of ChameleonCritter?

2. Why isn't the getActors method overridden in ChameleonCritter?

3. In the processActors method, the following line of code is found:

int r = (int) (Math.random( ) * n);

Would it be permissible to leave off the parenthesis surrounding all of Math.random( ) * n? Why?

4. Even though the act method of the Critter superclass is not overridden in ChameleonCritter, it does behave differently. Why?

5. Show code for making a ChameleonCritter object access its own Location and store the result in a variable called myLoc.

6. What modifications are needed to make ChameleonCritter drop a flower in the position it vacates after making a move? Be sure to guard against replacing the ChameleonCritter object with a Flower in case no move is made.

7. Suppose instead of calling super.makeMove(loc) in the makeMove method of ChameleonCritter, makeMove was called. What would be the result?

8. The method getGrid can be legally called within any overriding method of a subclass of Critter even though the subclass has no getGrid method. Why is

this legal?

9. Create a method called darken that is to be within ChameleonCritter and that darkens the color of the displayed object to 90% of its original intensity.

1.