3. This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendixes. In part (a) you will write a method to return an array list of all empty locations in a given grid. In part (b) you will write the class for a new type of `Critter`.

   (a) The `GridWorldUtilities` class contains `static` methods. A partial declaration of the `GridWorldUtilities` class is shown below.

```
public class GridWorldUtilities
{
  /** Gets all the locations in grid that do not contain objects.
   *  @param grid a reference to a BoundedGrid object
   *  @return an array list (possibly empty) of empty locations in grid.
   *          The size of the returned list is 0 if there are no empty locations in grid.
   *          Each empty location in grid should appear exactly once in the returned list.
   */
  public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
  {  /* to be implemented in part (a) */  }

  // There may be instance variables that are not shown.
}
```

   Write the `GridWorldUtilities` method `getEmptyLocations`. If there are no empty locations in `grid`, the method returns an empty array list. Otherwise, it returns an array list of all empty locations in `grid`. Each empty location should appear exactly once in the array list.

WRITE YOUR SOLUTION ON THE NEXT PAGE.

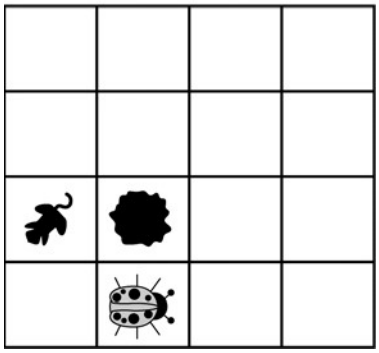
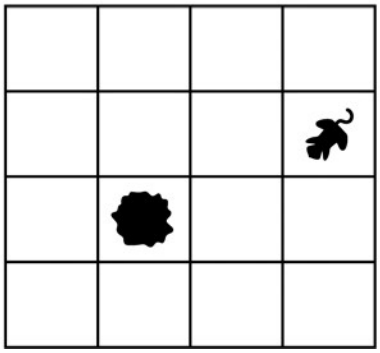
Complete method `getEmptyLocations` below.

```
/** Gets all the locations in grid that do not contain objects.
 *  @param grid a reference to a BoundedGrid object
 *  @return an array list (possibly empty) of empty locations in grid.
 *          The size of the returned list is 0 if there are no empty locations in grid.
 *          Each empty location in grid should appear exactly once in the returned list.
 */
public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
```

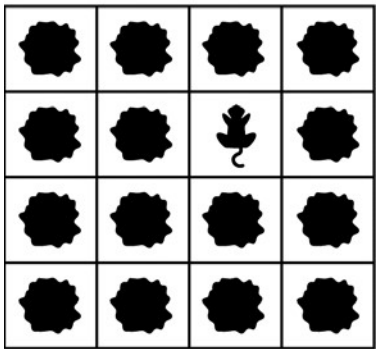
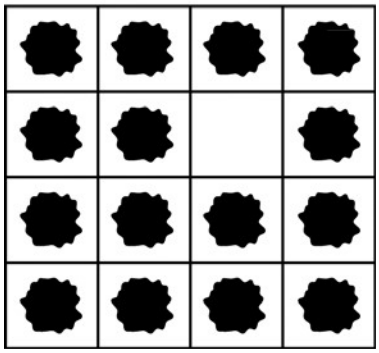
Part (b) begins on page 14.

(b) A `JumpingCritter` acts like a `Critter`, except that it moves by jumping to a randomly selected empty location in its grid. If there are no empty locations, the `JumpingCritter` removes itself from the grid.

The following diagram shows an example of a jumping critter that is able to move to an empty location. Example World #1 is shown below on the left. After the jumping critter at location (2, 0) acts, the world shown below on the right is one possible result.

EXAMPLE WORLD #1	POSSIBLE WORLD AFTER ACT
	
A jumping critter is in location (2, 0).	The jumping critter has eaten the bug that was in location (3, 1) and has moved to location (1, 3).

Example World #2 is shown below on the left. After the jumping critter at location (1, 2) acts, the world shown below on the right is the result.

EXAMPLE WORLD #2	WORLD AFTER ACT
	
A jumping critter is in location (1, 2).	The jumping critter removed itself from the grid because no empty locations were available.

Class information repeated from the beginning of the question

```
public class GridWorldUtilities

public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
```

Assume that the `GridWorldUtilities getEmptyLocations` method works as specified, regardless of what you wrote in part (a). Solutions that reimplement the functionality of this method will not receive full credit.

Write the complete `JumpingCritter` class. Do NOT override the `act` method. Remember that your design must not violate the postconditions of the methods of the `Critter` class.