```
Location loc1 = new Location(4, 3);
Location loc2 = new Location(3, 4);
```

1. How would you access the row value for loc1?

Answer： int row = loc1.getRow();

2. What is the value of b after the following statement is executed?

```
boolean b = loc1.equals(loc2);
```

Answer: false

3. What is the value of loc3 after the following statement is executed?

```
Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
```

Answer: loc3 ( 4,4 )

4. What is the value of dir after the following statement is executed?

```
int dir = loc1.getDirectionToward(new Location(6, 5));
```

Answer： 135degree

5. How does the getAdjacentLocation method know which adjacent location to return?

Answer：

Gets the adjacent location in any one of the eight compass directions.

@param direction the direction in which to find a neighbor location

@return the adjacent location in the direction that is closest to

```
  public Location getAdjacentLocation(int direction)
{
   // reduce mod 360 and round to closest multiple of 45
   int adjustedDirection = (direction + HALF_RIGHT / 2) % FULL_CIRCLE;
   if (adjustedDirection < 0)
      adjustedDirection += FULL_CIRCLE;

   adjustedDirection = (adjustedDirection / HALF_RIGHT) * HALF_RIGHT;
   int dc = 0;
   int dr = 0;
   if (adjustedDirection == EAST)
      dc = 1;
   else if (adjustedDirection == SOUTHEAST)
   {
      dc = 1;
      dr = 1;
   }
   else if (adjustedDirection == SOUTH)
      dr = 1;
   else if (adjustedDirection == SOUTHWEST)
   {
      dc = -1;
      dr = 1;
   }
   else if (adjustedDirection == WEST)
      dc = -1;
   else if (adjustedDirection == NORTHWEST)
   {
      dc = -1;
      dr = -1;
   }
   else if (adjustedDirection == NORTH)
      dr = -1;
   else if (adjustedDirection == NORTHEAST)
   {
      dc = 1;
      dr = -1;
   }
   return new Location(getRow() + dr, getCol() + dc);
}
```

Set 4

1. How can you obtain a count of the objects in a grid? How can you obtain a count of the empty locations in a bounded grid?

Answer: ArrayList<Location> x = gird.getOccupiedLocations();

int objectsSize = x.size();

int emprtySize = grid.getRow() * grid.getColumn() - x.size();

2. How can you check if location (10,10) is in a grid?

Answer: Location x = new Location(10,10);

grid.isVaild(x)

3. Grid contains method declarations, but no code is supplied in the methods. Why? Where can you find the implementations of these methods?

Answer: it's the interface.

If a class A implements Grid<E>, then A has the implementations of these methods

4. All methods that return multiple objects return them in an ArrayList. Do you think it would be a better design to return the objects in an array? Explain your answer.

Answer: you can trace the count of the object， that is， you can reduce the counts of circle

.

**Set 5**

1. Name three properties of every actor.
Answer: color direction location

2. When an actor is constructed, what is its direction and color?
Answer: NORTH BLUE

3. Why do you think that the Actor class was created as a class instead of an interface?
Answer: Because I can use the written methods of Actor class, however interface is not implementations of every method.

4. Can an actor put itself into a grid twice without first removing itself? Can an actor remove itself from a grid twice? Can an actor be placed into a grid, remove itself, and then put itself back? Try it out. What happens?
Answer: No throw new IllegalStateException( "This actor is already contained in a grid.");
No throw new IllegalStateException( "This actor is not contained in a grid.");
Yes

```
public void putSelfInGrid(Grid<Actor> gr, Location loc)
{
    if (grid != null)
        throw new IllegalStateException(
                "This actor is already contained in a grid.");

    Actor actor = gr.get(loc);
    if (actor != null)
        actor.removeSelfFromGrid();
    gr.put(loc, this);
    grid = gr;
    location = loc;
}
public void removeSelfFromGrid()
{
    if (grid == null)
        throw new IllegalStateException(
                "This actor is not contained in a grid.");
    if (grid.get(location) != this)
        throw new IllegalStateException(
                "The grid contains a different actor at location "
                        + location + ".");

    grid.remove(location);
    grid = null;
    location = null;
}
```

5. How can an actor turn 90 degrees to the right?

Answer: setDirection(getDirection()+90);

**Set 6**

1. Which statement(s) in the canMove method ensures that a bug does not try to move out of its grid?

Answer: Location next = loc.getAdjacentLocation(getDirection());
If (!gr.isValid(next))
return false;

2. Which statement(s) in the canMove method determines that a bug will not walk into a rock?

Answer: return (neighbor == null) || (neighbor instanceof Flower);

3. Which methods of the Grid interface are invoked by the canMove method and why?

Answer: gr.isValid(next)
gr.get(next);

4. Which method of the Location class is invoked by the canMove method and why?

Answer: loc.getAdjacentLocation(getDirection());

5. Which methods inherited from the Actor class are invoked in the canMove method?

Answer: getGrid();
getLocation();

6. What happens in the move method when the location immediately in front of the bug is out of the grid?

Answer: turn()

7. Is the variable loc needed in the move method, or could it be avoided by calling getLocation() multiple times?

Answer: yes, because it's a temporary variable

8. Why do you think the flowers that are dropped by a bug have the same color as the bug?

Answer Flower flower = new Flower(getColor());
flower.putSelfInGrid(gr, loc);

so,the color comes from the bug

9. When a bug removes itself from the grid, will it place a flower into its previous location?

Answer: no, because it don't move, so no flower

10. Which statement(s) in the move method places the flower into the grid at the bug's previous location?

Answer: Location loc = getLocation();
Flower flower = new Flower(getColor());
flower.putSelfInGrid(gr, loc);

11. If a bug needs to turn 180 degrees, how many times should it call the turn method?

Answer: 180 /45 = 4