Part 3
Set 3
1.loc1.getRow();

2.b = false;

3.loc3 = (4, 4);

4.dir = 135 (degrees)

5.getAdjacentLocation method will find the locations that on the direction which has declared and choose the closet one.

Set 4
Grid<E>
1.E.getNeighbors().size();
E.getEmptyAdjacentLocations().size();

2.E.isVaild(new Location(10, 10));
When it return true the location(10, 10) is in the grid, and when it return false the location(10, 10) isn't in the grid.

3.Grid is a place that show what codes produce.In AbstractGrid and the BoundedGrid and UnboundedGrid classes, we can find the implementations of these methods.

4.No, because the ArrayList don't need declare the size before it use, but the array have to. And we don't konw how many multiple objects we will use.

Set 5
1.Location, direction and color.

2.Its initial direction is North, and its initial color is blue.

3.An interface can't declare actor's variables and implement its methods.

4.An actor can't put itself into a grid twice without first removing itself.An actor can't remove itself from a grid twice.An actor can't be placed into a grid, remove itself, and then put itself back.
The same location only can be put an actor.And when actor has be removed, its location and grid are both null, it can be put back again, but we can put another new actor on it.

5.setDirection(getDirection() + 90);

Set 6
1.if (!gr.isValid(next)) return false;

2.Actor neighbor = gr.get(next);
   return (neighbor == null) || (neighbor instanceof Flower);
   // ok to move into empty location or onto flower
   // not ok to move onto any other actor

3.get() and isVaild() are invoked by the canMove method, because canMove() need isVaild() to judge next location whether can move and need get() to get next actor's grid.

4.getAdjacentLocation() is invoked by the canMove method, because it need to find a closet location that can move on actor's direction.

5.getLocation(), getGrid(), and getDirection() inherited from the Actor class are invoked in the canMove method.

6.It will remove itself from the grid.

7.loc is needed, because when the bug move a step, a flower shuold be put at the last location, so it need to be recorded, and calling getLocation() multiple times is useless.

8.Because their initial colors are the same color.

9.No.If you call the removeSelfFromGrid(), it only remove the bug and don't place a flower into its previous location.

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

11.Four times. Once turn() is 45 degrees.