Set 3

Assume the following statements when answering the following questions.
Location loc1 = new Location(4, 3);
Location loc2 = new Location(3, 4);
1. How would you access the row value for loc1?
int row = loc1.getRow();
2. What is the value of b after the following statement is executed?
boolean b = loc1.equals(loc2);
false
3. What is the value of loc3 after the following statement is executed?
Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
Location(4, 4)
4. What is the value of dir after the following statement is executed?
int dir = loc1.getDirectionToward(new Location(6, 5));
dir == 135 Location.SOUTHEAST.
5. How does the getAdjacentLocation method know which adjacent location to return?
According to the parameter(int direction). Then return the Adjacent Location in that direction.

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?
int count = grid.getOccupiedLocations().size();
int empty = grid.getNumRows() * grid.getNumCols() - grid.getOccupiedLocations().size();
2. How can you check if location (10,10) is in a grid?
boolean flag = grid.isValid(new Location(10, 10));
if flag == true, location(10, 10) is in the grid.
3. Grid contains method declarations, but no code is supplied in the methods. Why? Where can you find the implementations of these methods?
Because Grid is just an Interface, which only defines the Function of the grid.
We can find the implementations in the abstract class "AbstractGrid" which implements the Interface.
And "BoundedGrid", "UnboundedGrid" which derives the AbstraceGrid.
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.
No. We can't know how many objects will be returned. So we can't use array to store these objects because array's size is fixed. However, ArrayList is a container which can enlarge its size when there are many objects. So it's suitable for storing all the objects returned.

Set 5

1. Name three properties of every actor.
Color Direction Location
2. When an actor is constructed, what is its direction and color?
North Blue
3. Why do you think that the Actor class was created as a class instead of an interface?
Because most of the functions in actor class is common for other classes. There is no need in creating a new class to implements these methods seperately.
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?

(1)Can't pustself twice, java.lang.IllegalStateException: This actor is already contained in a grid.
(2)Can't remove self from a grid twice. java.lang.IllegalStateException: This actor is not contained in a grid.
(3) Yes, it can.
5. How can an actor turn 90 degrees to the right?
Call turn() method twice when facing to the NORTH.

Set 6

1. Which statement(s) in the canMove method ensures that a bug does not try to move out of its grid?
if(!gr.isValid(next)) {return false;}
2. Which statement(s) in the canMove method determines that a bug will not walk into a rock?
return neighbor == null || neighbor instanceof Flower;
3. Which methods of the Grid interface are invoked by the canMove method and why?
boolean isValid(Location var1); To justify if location is valid.
E get(Location var1); To get the actor at the next location along bug's path.
4. Which method of the Location class is invoked by the canMove method and why?
public Location getAdjacentLocation(int direction); To get location in front of the bug.

5. Which methods inherited from the Actor class are invoked in the canMove method?
getGrid()
getLocation()
6. What happens in the move method when the location immediately in front of the bug is out of the grid?
this.removeSelfFromGrid();
7. Is the variable loc needed in the move method, or could it be avoided by calling getLocation() multiple times?
Yes,the loc is needed. It can't be avoided by calling getLocation() multiple times.
Because loc stores the old position of the bug. It's needed for placing flowers:
flower.putSelfInGrid(gr, loc);
8. Why do you think the flowers that are dropped by a bug have the same color as the bug?
Flower flower = new Flower(this.getColor());
this.getColor() is the color of the bug.
9. When a bug removes itself from the grid, will it place a flower into its previous location?
No, it will leave nothing.
10. Which statement(s) in the move method places the flower into the grid at the bug's previous location?
Flower flower = new Flower(this.getColor());
flower.putSelfInGrid(gr, loc);
11. If a bug needs to turn 180 degrees, how many times should it call the turn method?
4 times.