**GridWorld Case Study - Chapter 4 Quiz**

1. Assume that Grid<Actor> grd has been initialized as a 10x10 bounded grid.

Assume that Actor a1 has put itself in grd at location (2,2) and Actor a2 has put itself in grd at location (3,3). Assume that Actor a3 has been created but has not been placed in grd.

What will be the result of the call

a3.putSelfInGrid(grd, new Location(2,2)) ?

a. a3 will be added to grd at location (2,2) so that a call to grd.get(new Location(2,2)) will return the set of actors {a1, a3}

b. a3 willl be placed in grd at location (2,2) after a1 has been removed from that location in grd.

c. grd will not change because when an actor tries to place itself into a location already occupied by another actor, no change is made.

d. The call will not compile; it should be a3.putSelfInGrid(grd, 2, 2)

e. An exception will be thrown because a3 is trying toput itself into an already occupied location in grd.

2. Make the same assumptions about grid grd and actors a1 and a2 as for the previous question.

What will be the result of the call

a2.putSelfInGrid(grd, a1.getLocation()) ?

a. The call will not compile; it should be a2.putSelfInGrid(grd, a1.getRow(), a1.getCol())

b. An exception will be thrown because an actor that has already been added to a grid with a call to putSelfInGrid cannot be added to a grid again (unless it has been removed).

c. Actor a1 will be removed from grd and actor a2 will removed from location (3,3) in grd and placed in location (2,2) in grd, changing its own location to (2,2)

d. Actor a2 will be removed from location (3,3) in grd and added to location (2,2) in grd with its own location changed to (2,2). A subsequent call to grd.get(new Location(2,2)) would return the set of actors {a1, a2}

e. No change will be made to grd, or actors a1 and a2.

3. Assume that Grid<Actor> grd has been initialized as a 10x10 bounded grid. Assume the following statements have been executed.

Actor a1 = new Actor();

grd.put(new Location(5,5), a1);

a1.setDirection(Location.EAST);

Which of the following are the correct values for the attributes of a1 given by calls to a1.getColor(), a1.getLocation(), a1.getDirection() (in that order) ?

a. Color.black, (5,5), 0

b. Color.black, (5,5), 90

c. Color.blue, (5,5), 0

d. Color.blue, (5,5), 90

e. Color.blue, null, 90

4. Assume that Grid<Actor> grd has been initialized as an unbounded grid and that Actor a1 has put itself into grd at location (3,5). Assume that instances of Rock have been placed in the grid locations (1,1), (1,3), and (4,6). Consider the following code segment.

Location nextLoc = a1.getLocation().getAdjacentLocation(Location.NORTHWEST);

while(grd.isValid(nextLoc))

{

a1.moveTo(nextLoc);

nextLoc = a1.getLocation().getAdjacentLocation(Location.NORTHWEST);

}

Which of the following best describes what happens when this code segment is executed?

a. a1 never changes location because the loop immediately exits due to the rock at location (4,6)

b. a1 moves once to location (2,4). the loop then exits due to the rock at location (1,3)

c. a1 moves once to location (2,4), then during execution of the second iteration of the loop the call to a1.moveTo(nextLoc) throws an exception

d. a1 move three times to location (0,2), then the loop exits.

e. a1 continues to move northwest; the loop never terminates.

5. Consider the following code segment.

Actor a1 = new Actor();

a1.setDirection(a1.getDirection() + 135);

Which of the following best describes the result of compiling and executing this code segment?

a. The code will not compile. the only way for a1 to change its direction is by a call to the turn method

b. A NullPointerException is thrown when the second statement is executes because a1.getDirection() returns null, because a1 has never called putSelfInGrid

c. a1.getDirection == Location.NORTHWEST is true

d. a1.getDirection == Location.SOUTHEAST is true

e. The direction of a1 is the same as it would be if the second statement were replaced by "a1.turn(Location.RIGHT + Location.HALF\_RIGHT)".

6. Which of the following statements insures that the Bug canMove method returns false if a Rock is immediately in front of the bug?

Choose one answer. a. if(gr.get(next) instanceof Rock) return false;

b. if(!gr.isValid(next)) return false;

c. return !(neighbor instanceof Rock);

d. return (neighbor == null) || (neighbor instanceof Flower);

e. There is no such statement.

7. Assume that DumbBug extends Bug and overrides the act() method with a method with the single statement "move();". Assume that an instance of DumbBug is placed in an unbounded grid that also contains several instances of Rock and Flower at random locations. Consider the following descriptions of what might happen when the GridWorld run button is clicked (the dumb bug's act method is repeatedly called).

I. The dumb bug moves forward in a straight line, dropping flowers behind it.

II. The dumb bug moves forward dropping flowers behind it until it reaches a location where there is a rock immediately in front of it. At this point the dumb bug no longer moves, and does not take any other action.

III. The dumb bug moves forward dropping flowers behind it until it reaches a location where there is a rock immediately in front of it. At this point an exception is thrown the next time the dumb bug tries to move.

Which of these scenarios could occur when the GridWorld simulation is run?

a. I only

b. II only

c. III only

d. I or II only

e. I or III only

8. The Bug move method uses a local variable loc for clarity. Suppose the Bug move method is rewritten as follows (the changed lines are indicated by \*).

public void move()

{

Grid<Actor> gr = getGrid();

if(gr == null)

return;

\* Location next = getLocation().getAdjacentLocation(getDirection());

// replaces two lines Location loc = getLocation();

// Location next = loc.getAdjacentLocation(getDirection());

if(gr.isValid(next))

moveTo(next);

else

removeSelfFromGrid();

Flower flower = new Flower(getColor());

\* flower.putSelfInGrid(gr, getLocation());

// replaces flower.putSelfInGrid(loc);

}

What will be the result of this change?

a. The revised method will work exactly the same way as the original.

b. The revised method will fail to compile.

c. The revised method will always throw an exception

d. The revised method will make the Bug move exactly as it did before, but no flowers will be left behind it.

e. The first time the revised method is called, the bug will disappear from the grid and a flower will be placed in the location in front of the bug's original position.