AP CS A name: Kevin Tran

GridWorld Parts 4 and 5

Follow through the GridWorld Activities described in the GridWorld Case Study Student Manual with Appendixes doc found on Canvas. Pick up on p. 29, the Critter class.

1. What are the 5 basic actions common to all critters when they act?

getactors()

processactors()

getmovelocations()

selectmovelocation()

makeMove()

2. Should subclasses of Critter override the getMoveLocations method? If so, give an example of when you would do so.

Yes, each creature has different getMoveLocations conditions, thus you would override the parent gerMoveLocations method in order to implement those conditions

3. Describe another way that a critter could possibly process an actor, besides the default actions in the Critter class.

The creature can swap places with the actor

4. Why is there no Critter constructor?

Because the constructor is located in actor

read through p. 33

5. Why does act cause ChameleonCritter to act differently from a Critter even though ChameleonCritter does not override act?

ChameleonCritter has overwrites over processactors and makemove, act calls these methods

6. How would you make ChameleonCritter drop flowers in its old location when it moves?

Add following to moveTo

Flower flow = new Flower(getColor());

flow.putSelfinGrid(grid, loc);

7. Which class contains the getGrid method?

actor()

8. How can a ChameleonCritter access its own grid?

Call the getGrid method

read to p. 35

9. If a CrabCritter has location (3,4) and faces east, what are the possible locations for actors that are returned by a call to the getActors method?

(3,5) (2,5) (4,5)

Big jump here…and a bit of foreshadowing: Read up through p. 40.

10. a) Look at the methods from the Grid interface. Are these methods abstract? How do you know they are or are not?

The methods are abstract. There is no implementation of the methods (they have no body)

b) List the methods specified in Grid and after each identify the class in which it is implemented.

|  |  |
| --- | --- |
| Int getNumRows() | Boundedgrid, unboundedGrid |
| Int getNumCols() | BoundedGrid, unboundedGrid |
| Boolean isValid(Location loc) | BoundedGrid, unboundedGrid |
| E put(Location loc, E obj) | BoundedGrid, unboundedGrid |
| E remove(Location loc) | BoundedGrid, unboundedGrid |
| E get(Location loc) | BoundedGrid, unboundedGrid |
| ArrayList<Location> getOccupiedLocation(Location loc) | abstractgrid |
| ArrayList<Location> getValidAdjacentLocations(Location loc) | abstractgrid |
| ArrayList<Location> getEmptyAdjacentLocations(Location loc) | abstractgrid |
| ArrayList<Location> getOccupiedAdjacentLocations(Location loc) | abstractgrid |
| Array<E> getNeighbors(Location loc) abstractedgrid | abstractgrid |

11. a) Is AbstractGrid really an abstract class? How do you know?

Yes, it has public abstract class in the header, along with abstract methods

b) If the header line for AbstractGrid did not identify the class as abstract, would this class have worked as a regular superclass for BoundedGrid and UnboundedGrid? Why or why not?

It would work as a parent class, but would need body implementations for the methods. It would not work as an abstract class.

c) What is the benefit of making the abstract class AbstractGrid? Why not just have BoundedGrid and UnboundedGrid implement Grid and cut out the extra class?

All the methods in Grid only have to be implemented once in AbstractGrid

12. How do BoundedGrid and UnboundedGrid implement the get method differently? Why does UnboundedGrid use a hash map instead of 2-d array? (Even though we haven’t covered these structures yet, I think you could answer that, especially after reading pp. 41 and 42.) (NOTE: HashMaps, TreeMaps, sparse arrays and linked lists are data structures not covered on the AP Exam.)

boundedGrid contains a 2d array as a instance variable

UnboundedGrid contains a hashmap

boundedGrid has finite spaces, while unboundedGrid has infinite spaces to go to