Inheritance and Gridworld Quiz

Key

Multiple Choice
1. (A) - number of doors is a number, presence of air conditioning is a boolean and average miles per gallon is a number. Why wrap such simple properties into classes when we have primitive types we can use? Choices B-E can immediately be discarded on that basis. Further analysis would reveal that choices C-E use inheritance improperly. The point of inheritance is to share behavior between related types, establishing "is-a" relationships. For example, Sedan would be an appropriate subclass of Car because a sedan is a type of car. A door is clearly not a car, so it wouldn't make sense to have a Door class subclass a Car class.

2. (B) - an interface establishes a contract -- all classes which implement a particular interface must provide implementations of the abstract methods specified in the interface definition. The Student interface includes exactly two abstract methods: getGPA() and getSemesterUnits(). Those are the only ones FullTimeStudent is bound to implement. It could, of course, implement other methods, but it doesn't need to in order to compile.

3. (B) - Bug is implemented such that if an instance of its class can't move in the direction it is facing, it will turn 45 degrees. Actors move one at a time in the simulation so one of the Bugs will move into (3,6) first because no obstacles exist there. When it comes time for the second Bug to move, the first Bug is an obstacle so it turns.

4. (B) - Bug is implemented such that all Actors represent obstacles except for Flowers. If a location is occupied by a Flower, a Bug instance will move there. Any other actor will make the bug move.

5. (D) - The important thing to notice with this question was that it was asking for LEGAL constructors, not useful ones. II is immediately out because it accesses private fields. Recall that private fields are inaccessible to a class' subclasses. III is good because it calls the superclass' constructor to set common fields and then sets the name field separately. I is fine too since it just sets the name. Granted, it's not very useful, but that's not what the question was asking!

Inheritance Mystery
Fire
Fire 1
Fire 2

Rain
Snow 1 Fire 2
Fire 2

Rain
Fire 1 Rain 1
Fire 2

Fire
Ice 1
Fire 2

Gridworld
As I mentioned, I graded this one pretty loosely. For destroy, I was looking for code to find the appropriate column and iterate over all of its rows BELOW the Twister finding occupied locations and removing those Actors from the grid. For move, I was looking for generating a random number, having code to make the Twister go either east or west and code to remove itself.

Of course, this is only one possible solution. Take a look at some of the things that allow the code to be short -- for example, in move, we set next to east no matter what and then reset it to west based on a random number.

```
public void destroy() {
    Grid<Actor> gr = getGrid();
    ArrayList<Location> locs = gr.getOccupiedLocations();

    for (Location loc : locs) {
      Actor a = gr.get(loc);
      Location aLoc = a.getLocation();
      if(aLoc.getCol() == getLocation().getCol() &&
     aLoc.getRow() > getLocation().getRow()) {
            a.removeSelfFromGrid();
        }
    }
}

public void move() {
    Location next = getLocation().getAdjacentLocation(Location.EAST);
    if(Math.random() < 0.5) {
        next = getLocation().getAdjacentLocation(Location.WEST);
    }
    if(getGrid().isValid(next)) {
        moveTo(next);
    } else {
        removeSelfFromGrid();
    }
}
```