1. Modify the ChameleonCritter so that when it goes to process other actors, if it doesn't find any to process, it turns its color back to blue. Use the ChameleonClient I included to test it out.
2. Create a class called ChameleonKid that extends ChameleonCritter as modified from the previous exercise. ChameleonKid should change its color to one of its neighbors like its parent does, but it should only select from neighbors that are Rocks and Flowers. If there is no neighbor that is either a Rock or Flower, then the ChameleonKid should turn back to blue like the modified ChameleonCritter. Decide before you start: is it better to override getActors() or processActors()? Should you override any other methods? Only override methods where necessary. Use the included ChameleonKidClient to test it out.
3. Create a class called GoingHomeCritter that extends Critter. When a GoingHomeCritter goes to make its move, it should select from the lowest location that a Critter could move to. Remember that Location objects have a concept of ordering because they implement the Comparable interface, and thus have a .compareTo() method. Decide before you start: is it best to override getMoveLocations(), selectMoveLocation(), or makeMove()? Only override methods where necessary. Be sure to consider the case when there are no locations available that a Critter could move to, i.e. see what Critter does in this case. Try it out with the client I supplied.