```java
package club.westcs.GridWorldBeckerbauer;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Random;

import info.gridworld.actor.Actor;
import info.gridworld.actor.ActorWorld;
import info.gridworld.actor.Bug;
import info.gridworld.actor.Critter;
import info.gridworld.actor.Flower;
import info.gridworld.actor.Rock;
import info.gridworld.grid.Grid;
import info.gridworld.grid.Location;

public class ThanosCritterLevel2 extends Critter{

    private Random rand;
    private ArrayList<Actor> halfActs;
    private ArrayList<Location> allLocs1;
    private boolean hasInfinityStones, firstRun, ender;
    private Rock spaceStone, realityStone, mindStone, powerStone, timeStone, soulStone;


    public ThanosCritterLevel2() {
        ender = false;
        firstRun = true;
        rand = new Random();
        hasInfinityStones = false;
        halfActs = new ArrayList<>();
        allLocs1 = new ArrayList<>();
        spaceStone = new Rock();
        spaceStone.setColor(Color.BLUE);
        mindStone = new Rock();
        mindStone.setColor(Color.YELLOW);
        realityStone = new Rock();
        realityStone.setColor(Color.RED);
        powerStone = new Rock();

        powerStone.setColor(new Color(128, 0, 128));
        timeStone = new Rock();
        timeStone.setColor(Color.GREEN);
        soulStone = new Rock();
        soulStone.setColor(Color.ORANGE);
    }



    public void addStones() {
        spaceStone.putSelfInGrid(getGrid(),randomLocation() );
        realityStone.putSelfInGrid(getGrid(), randomLocation() );
        mindStone.putSelfInGrid(getGrid(), randomLocation() );
        powerStone.putSelfInGrid(getGrid(), randomLocation() );
        timeStone.putSelfInGrid(getGrid(), randomLocation() );
        soulStone.putSelfInGrid(getGrid(), randomLocation() );
    }

    public Location randomLocation() {
        Location loc = new Location(0,0);
        do {

            loc = new Location(rand.nextInt(getGrid().getNumRows()), rand.nextInt(getGrid().getNumCols()));
        }
        while(getGrid().get(loc) != null);
        System.out.println(loc);
        return loc;
    }

    public void act() {
        if(getGrid() == null || ender) {
            return;
        }
        if(firstRun) {
            firstRun = false;
            addStones();
        }
        ArrayList<Actor> infinityStones = getInfinityStones();
```

```
77          if(infinityStones.isEmpty()) {
78              doomed();
79              System.out.println("Mr.Stark, I don't feel so good....");
80              System.out.println("Doomed " + halfActs);
81              remove();
82              ender = true;
83          }
84          else {
85              moveToInfinityStones(infinityStones);
86          }
87      }
88
89      public ArrayList<Actor> doomed() {
90          halfActs.clear();
91          ArrayList<Location> allLocs1 = getGrid().getOccupiedLocations();
92          for(Location b: allLocs1) {
93              if(!(getGrid().get(b) .equals(this))) {
94                  halfActs.add(getGrid().get(b));
95                  if(halfActs.size() >= allLocs1.size()/2) {
96                      break;
97                  }
98              }
99          }
100         return halfActs;
101     }
102
103     public void remove() {
104         for(Actor a: halfActs) {
105             a.removeSelfFromGrid();
106         }
107     }
108
109     private void moveToInfinityStones(ArrayList<Actor> infinityStones) {
110         Actor infinityStone = pickAInfinityStone(infinityStones);
111         for(int i = 0; i < 1; i++) {
112             int dir = getLocation().getDirectionToward(infinityStone.getLocation());
113             Location next = getLocation().getAdjacentLocation(dir);
114             if(getGrid().get(next) == null) {
```

```
115                 makeMove(next);
116             }
117             addInfinityStones();
118 
119         }
120     }
121 
122     private ArrayList<Actor> addInfinityStones() {
123         ArrayList<Location> locs = getGrid().getOccupiedLocations();
124         ArrayList<Actor> myInfinityStones = new ArrayList<>();
125         for(Location loc : locs) {
126             Actor temp = getGrid().get(loc);
127             if(temp instanceof Rock ) {
128                 myInfinityStones.add(temp);
129             }
130         }
131         return myInfinityStones;
132     }
133 
134     private Actor pickAInfinityStone(ArrayList<Actor> infinityStones) {
135         double dist = 1000000;
136         Actor choice = new Actor();
137         for(Actor a: infinityStones) {
138             if(isCloser(a, dist)) {
139                 choice = a;
140                 dist = saveDist(a);
141             }
142         }
143         return choice;
144     }
145 
146     private double saveDist(Actor a) {
147         System.out.println(a.getLocation().getCol());
148         return Math.sqrt(Math.pow(a.getLocation().getCol() - this.getLocation().getCol() , 2) +
149                 Math.pow(a.getLocation().getRow() - this.getLocation().getRow() , 2));
150     }
151 
152     private boolean isCloser(Actor a, double dist) {
```

```
153         return saveDist(a) <= dist;
154     }
155 
156     private ArrayList<Actor> getInfinityStones() {
157         processActors(getActors());
158         ArrayList<Location> locs = getGrid().getOccupiedLocations();
159         ArrayList<Actor> InfinityStones = new ArrayList<>();
160         for(Location loc : locs) {
161             Actor temp = getGrid().get(loc);
162             if(temp instanceof Rock) {
163                 InfinityStones.add(temp);
164                 hasInfinityStones = true;
165             }
166         }
167         return InfinityStones;
168     }
169 
170     @Override
171 
172     public void processActors(ArrayList<Actor> actors)
173     {
174         for (Actor a : actors)
175         {
176             if (a instanceof Rock && !(a instanceof Critter))
177                 a.removeSelfFromGrid();
178         }
179     }
180 
181 
182 }
183 
```