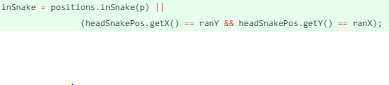
Changes to make after Ben’s Food-Spawn Update

1. Intro new vars for speed and size of AI snake, if you want a difference
   1. Line 30 & 31 for reading from file
2. 28
   1. Make AISnake object
   2. Tuple positionDepart for bad snake
3. 46
   1. foodEaten for bad snake
4. 43,44
   1. headsnakePos for bad snake
5. 80 boolean eatingFood
   1. Make snake specific, (calling for head pos in scope of boolean)
   2. 81, 82 update individually
   3. else→ compare bools for draw
6. 110 valid tuple
   1. Check sets of both snakes in while statement
7. 119
   1. headSnakePos.getX() == ranY
   2. 

//in the Tuple class

Public static Tuple toLeft(Tuple spot){

Return new Tuple(spot.x-1 %19, spot.y);

}

Public static Tuple toRight(Tuple spot){

Return Tuple right = new Tuple(spot.x+1 %19, spot.y);

}

Public static Tuple above(Tuple spot){

Return new Tuple(spot.getX(), spot.getY() + 1 % 19);

}

Public static Tuple below(Tuple spot){

Return new Tuple(spot.getX(), spot.getY() - 1 % 19);

}

Public static int availability(Tuple spot){

Tuple left = spot.toLeft();

Tuple right = spot.toRight();

Tuple up= spot.above();

Tuple down= spot.below();

If( goodPositions.contains(spot) || badPositions.contains(spot)){

Return 0;

} else{

Return (availability(left) + availability(right) + availability(up) +availability(down ));

}

//In moveInternal function for survivalist AI

Int[] safetyOfMoves = [];

Int safest = 0;

safety.add ( availability (badSnakePos.toLeft()) );

safety.add ( availability (badSnakePos.toRight()) );

safety.add ( availability (badSnakePos.Above()) );

safety.add ( availability (badSnakePos.toLeft()) );

Safest = Numbers.Utils.max(safetyOfMoves);

Int direction = safetyOfMoves.indexOf (safest); //use in External movement function