Rules for same game:

You start off with a board of circles, each colored with one of three random colors (green, blue or red).
When you select a circle, all circles in the same color region are selected. Two circles are in the same color region if there is a single color path between (and including) them using horizontal and/or vertical lines.
Each time you select a same-color region, you gain a points depending on the number of circles in the region (n) and their color:
single circle regions (i.e. n=1) - earn -10 points.
Blue or Green regions: earn ⌊ n ln n ⌋ points.
Red regions: earn - ⌊ n ln n ⌋ points.
The game ends when there are no circles, or all same-color regions have only one circle.
Your final score (which you are attempting to maximize) is the total number of points you accumulate.
After a color region is selected, all its circles disappear. Circles fall down vertically, until they are compacted (i.e. no empty spaces between them), and leftward horizontally (i.e. no empty columns).

Homework:

Import the project repository from bit bucket, modify the brain package and add a myBrain.java and a greedyBrain.java extending Brain.java. The greedyBrain.java should always pick the largest same-color region and the myBrain.java should get higher score than simpleBrain.java and lazyBrain.java.

My solution:

greedyBrain.java:
greedyBrain.java is similar with the lazyBrain.java. So, I just modify the loop part in lazyBrain.java.

myBrain.java:
Priority for different regions:
1, red regions.
2, blue or green regions.
3, single red region.
4, single blue or green region.
Every time my brain does a choice, it will choose the largest region with highest priority.