1. Name: Yancheng Yuan

X500.: yuanx322

1. My Strategy:

Create a list for conquered planets. Create two lists for conquered and unconquered planets. Create a map to keep track of current population of my conquered planets. Create two maps to link planet ID and IVisiblePlanet for conquered planets and unconquered planets (and take advantage of the containsKey() method as well). Move through all visible planet at the beginning of each turn to put planets to corresponding data structure. Use the list of conquered planets to do the depth first traversal (with a separate method) and get a traversed queue. Use the traversed queue to find planets that is next to neutral or opponent (with a separate method). This is a queue called frontierPlanets. Use a while loop to send population to the frontier planets. Let the frontier planets attack planets next to them when the population is greater than the edge number, start from neutral to opponent (send one person to neutral planet and send most people to opponent planet). Then handle a special case of some planets with only one edge. Send all population except one remaining to the other end of the edge. Finally, if the frontier planet doesn’t satisfy the condition above, skip this planet.

1. Data structures I used:

Queue: keep track of a list of planets, easy to loop through it with poll() method;

Map: relate different attributes of planets together, make the code easier to write;

List: to find the traversal order;

1. Strategy I wanted to be graded: Strategy.java
2. No external libraries
3. Known Issue: planets at the corner might send population between each other, not sending people to the center of the graph