Proposal

3 Lucky Duckies - Ziying Jian, Nora Miller, Gloria Lee

- Our trio's proposal is to create a Chain Reaction game in terminal - a grid based game in which each box of the grid contains a maximum of 2, 3, or 4 dots. Corner boxes contain at most 2 dots; edge boxes at most 3; middle boxes at most 4. When this maximum is reached, the box 'explodes' - each dot contained in the box is sent to one neighboring box.

- Since this is a 2 (or more)-player game, the objective is to eliminate all of the other player's dots. Each player's dots have a color unique to them (red or green etc.). All boxes can contain the dots of only one player, and when one of your boxes explodes into a box containing dots of the other player, the other player's dots become yours. Once the board only has dots of your color, you've won. Since explosions in one box can often lead to chain reactions of explosions in other boxes, hence the name Chain Reaction.

- Essentially, we're trying to create a version of the game that can be run in the terminal. https://play.google.com/store/apps/details?id=com.BuddyMattEnt.ChainReaction&hl=en_US&gl=US

PLAN:

1. Basic apparatus
    - **a. Make the grid work**
        - **i. Implement all of class DotCluster**
            1. **Decide on a set of Strings to represent different numbers of dots for different players**
        - **ii. Make both constructors for Grid, and its toString method**
        - **iii. Create a way to display warnings for dot clusters about to explode**
    - **b. Allow the user to add dots**
        - **i. By typing in coordinates and having Java recognize that**
    - c. Once a dot is added,
        - i. check if explosion is necessary
        - ii. execute chain explosions if necessary
2. Driver
    - a. Set up a turn system for 2 players, with some exit command

Proposal

3 Lucky Duckies - Ziying Jian, Nora Miller, Gloria Lee

   b. Set up check system to check after each turn if game is won
3. Possible ways to extend code:
   a. Set up an AI replacing the second player

UML for the game apparatus (We will outline the AI player part later)

public class Grid	public class DotCluster	public class Driver
- int height - int width DotCluster[height][width] contents Contains the contents of the Grid	- int numDots - String color “red” or “green” if numDots > 0; “” if numDots = 0 ~~String rep~~ ~~The visual representation of the DotCluster to the user. Should use color and 1 or 4 characters~~ (see toString) - final int maxDots This will be set when the Grid is initialized, and should not be able to be changed (there are no circumstances in which this max will change)	Scanner redPlayer Used to take in input from the red player Scanner greenPlayer Used to take in input from the green player boolean win true if a player has won the game and the game should end
Grid() Initializes a Grid with size 0x0 Grid(int[] size) Initializes a grid with specified size, using the constructor from above and initializing contents and size. For each DotCluster in contents, maxDots will be set based on the DotCluster’s position.	DotCluster() toString() returns a character (or 4) to represent the DotCluster, based on numDots and color. Necessary for toString in Grid to work. Preferable to having an instance variable for the implementati	runGame() First creates an instance of Grid. Then, prints out instructions to the player, as needed, then the Grid itself. Invokes checkWin(). If true, game ends, and winning player is congratulated. If false, invokes turn(). turn()

Proposal

3 Lucky Duckies - Ziying Jian, Nora Miller, Gloria Lee

toString() prints out grid with representation of DotClusters in each grid box explode(int x, int y) Helper method for method below. Takes dots from one box and distributes them to neighboring boxes. Uses DotCluster.numDots. Sets the color of neighboring clusters to that of of the cluster that just exploded. ~~explodeAll(int x, int y)~~ ~~Uses the above method to perform all of the explosions initiated when one box explodes~~ addDot(String color, int x, int y) This adds a Dot by increasing DotCluster.numDots of the relevant DotCluster by 1. It then checks whether this reaches maxDots for that cluster, and if so, it invokes explode. (Note that using this function and explode() together obviates explodeAll.)		Prints out a statement telling the (two) user(s for now) whose turn it is. Receives user input, alternating between using redPlayer and greenPlayer, on where they want to add a dot. Checks whether this location is permissible - if so, invokes addDot; if not, allows them to try again until they give a permissible location. boolean checkWin() Checks if a player has won by checking if DotCluster.color has the same value for every DotCluster in Grid.