Take-Home Notes

**CONTAINMENT & INHERITANCE**
- Always try to factor code out code if certain classes/methods are getting too crowded
  - SideBar class factored out from PaneOrganizer for updating Labels
- Polymorphism
  - What do Dots, Energizers, and Ghosts have in common?
    - They can all collide with Pacman!
- GhostPen class
  - Keeps track of the unreleased Ghosts(or Ghosts that have been eaten)
  - Releases Ghosts in the order in which they got eaten

**BOARD**
- In section, we talked about how to construct the board logically:
  - Recommended using a 2D-array (23 x 23) of SmartSquares
    - Each SmartSquare contain a list of Collidable objects (Energizers, Dots, Ghosts, etc.)
    - You are not limited to this option. Feel free to explore other designs that you have in mind!
  - Use the enum support map in the stencil code to determine what a SmartSquare is/contains at the beginning of the game
    - Enum is a class that represents very simple labels and logic
    - Walls or other object-holding locations

**BFS**
- An algorithm used when Ghosts have more than one valid move options and it would:
  - Lead the Ghosts to the SmartSquare that is closest to the target
    - Different modes, different targets
  - Return the direction that the Ghosts should move in so they can get to the target as fast as possible
- Handout gives a high-level pseudocode and here are some things you want to pay attention to:
  - Usage of Queue data structure(implemented using Java’s built-in LinkedList)
    - First In, First Out(FIFO)
  - Which neighbor SmartSquare is valid?
    - Not the one that the Ghost just came from
    - Not a Wall
    - Not one that has already been visited
      - Visited means that this SmartSquare’s location has already been filled with a Direction
- WRAPPING-
  - Some visually disjoint locations are valid neighbors!
  - What is the shortest distance?
    - Pythagorean distance = \( \sqrt{\text{rowDiff}^2 + \text{colDiff}^2} \)
      - In Java, doing exponents is \( \text{Math.pow(double base, double, exp)} \)!

- TA TIPS FOR COMPLETING THE PROJECT
  - Start **EARLY**
  - Hand simulate for BFS
    - Break the pseudocode into parts and simulate line by line!
    - Keep track of what should change/be modified at each step of the algorithm!
  - Coding **INCREMENTALLY**!
    - Run your program each time you have added a major logic! Do not move on to the next step if you have not completed the step before.