1) Annotating Player.h and Maze.h:

a) Draw a square around the constructors for the Player, Board, and Maze objects.

b) Draw a circle around the fields (class attributes) for the Player, Board, and Maze objects.

c) Underline any methods that you think should not be public. Explain why you think that they should not be public.

2) Critiquing the design of the "maze" game:

a) Methods: should do 1 thing and do it well. They should avoid long parameter lists and lots of boolean flags. Which, if any, methods does your group think are not designed well? Is there a method that you think is a good example of being well-designed? which?

b) Fields: should be part of the inherent internal state of the object. Their values should be meaningful throughout the object's life, and their state should persist longer than any one method. Which, if any, fields does your group think should not be fields? Why not? What is an example of a field that definitely should be a field? why?

c) Fill in the following table. Use a check mark (✓) to indicate when you believe that a class does fulfill the trait. If you don't think that a class fulfills the trait, explain why not.

Trait	Player	Board	Maze
cohesive (one single abstraction)			
complete (provides a complete interface)			
clear (the interface makes sense)			
convenient (makes things simpler in the long run)			
consistent (names, parameters, ordering, behavior should be consistent)			

1) Re-design the "maze" game. **Do not write code**. Indicate what classes you would have, and what **methods, fields, and interactions** they would have. Interactions here indicate which methods and fields call-on and depend-on methods in other classes and other classes themselves. Once you've done this, write a short paragraph summarizing the differences between your re-designed "maze" game and the original one we did for homework 1.