Design Document

Project Description:

Two chickens face each other on a 10x10 grid. One chicken is controlled by the computer, one by the human. One chicken starts in the upper left corner and the other in the lower right corner. The chicken can move, turn or shoot, but not both. The way the chicken moves is the way the chicken is pointed. It can change its direction, but this is one turn.

The base chicken class can only move two squares in the direction it faces. The chicken can only shoot two squares away in a straight line the way it is facing. The chicken will have variables to store the Hit points. Hit points represent how many times it can be hit before it dies. I.E. if a chicken has 10 hit points and it is hit for 1 damage point it will have 9 hit points left. The amount of damage could change depending on which sub chicken is firing at it.

A turn consists of:

Chicken 1 Move: Pick a direction and update square 2 spaces in that direction. If the chicken is at the end of the grid it can only go that far.

Or

Chicken 1 fire: Fire two squares away in the direction the chicken is facing. Check to see if chicken 2 is there, if so, deduct point(s) from chicken 2's hit points. If chicken 2's hit points are zero game is over. If not, chicken two takes turn.

Or

Chicken 1 will change directions and prepare to move or fire the next turn.

I have provided you with an interface for the chicken class. Your base class will implement this interface and add the required methods. This first assignment will be to create a design document that details the base class as well as the board class and game class necessary to play a game of chicken.

List the Classes and their methods and variables used.

```
public class Game {
    private Scanner scan;
    private bool game_over = false;
    Int turn = 0;
    Chicken chick1, chick2;
```

Design Document

```
        Private Board board;

        Public Game(Scanner scan)
        Private void( check_game_over()
        Private take_turns()
        private void initialize_game()
        private void take_turns()
        private void take_human_turn()
        private void take_computer_turn()
        private void initialize_characters()
        private void initialize_human_character()
        private void initialize_computer_character()
        private void display_info()
        Private void display_player_directions();
        public void play_game()
}


public class Board {
        private char[][] board;
        int rows =10 , columns = 10;
        public Board(int rows, int columns)
        public void initialize_board()
        public void display_board()
}
```

Design Document

Simple flow chart.

Game.play_game() -> intialize_game()

intialize_game() -> intialize_board()

intialize_game() -> initialize_characters()

initialize_characters() -> initialize_human_character()

initialize_characters() -> initialize_computer_character()

intialize_game() -> take_turns()