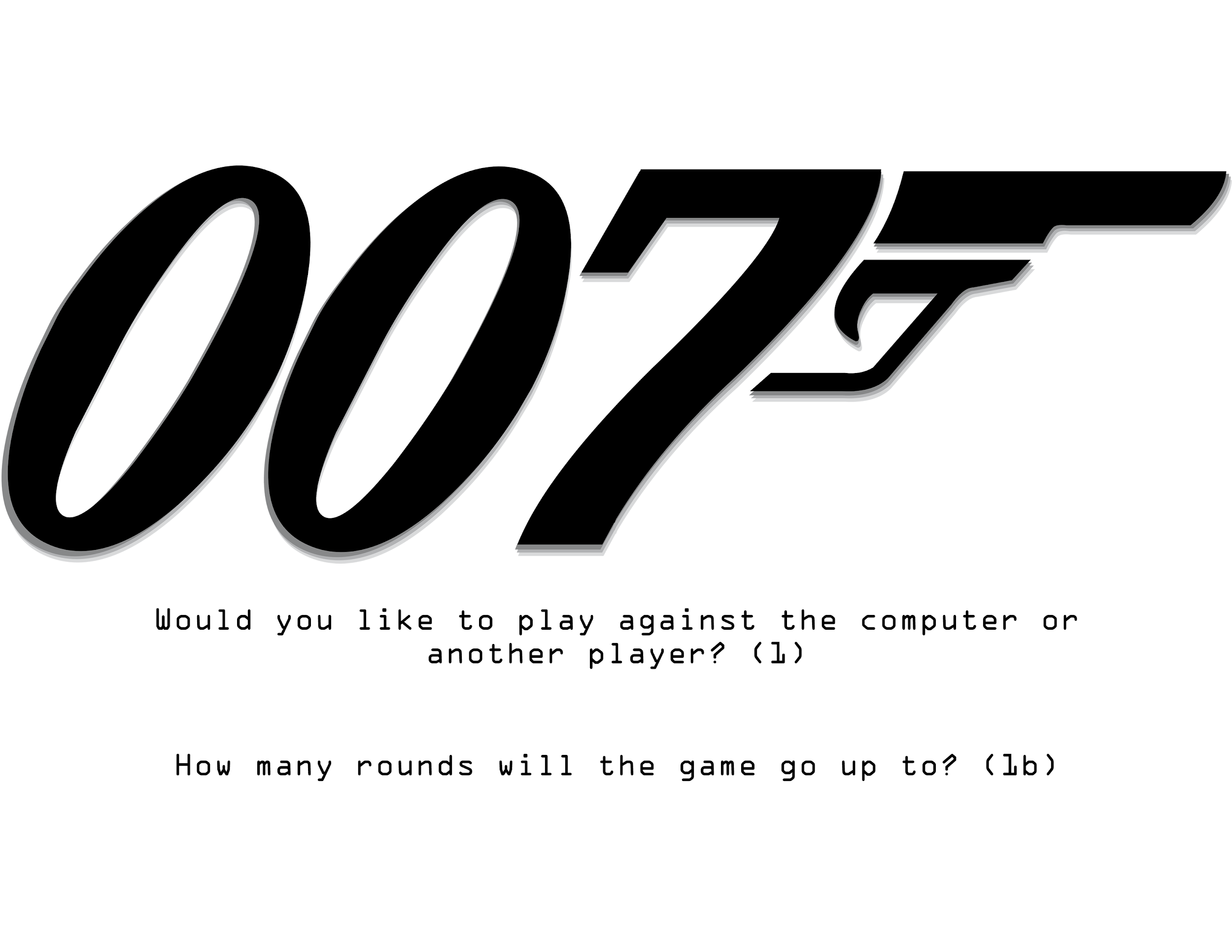
**ICS3U3 Final Project: Design Document**

**Program Name:** 007.java

**Program Overview:** 007.java is a game designed for no more or less than two people. 007 is very similar to rock-paper-scissors but instead involves firearms. 007.java is intended for casual play and is targeted to every audience. The game involves players choosing from three moves: block, shoot, and reload. Players will pick their moves one at a time and cannot be changed once submitted. Similar to rock-paper-scissors, certain moves will counter others like rock vs. scissors. In 007.java, if you don’t block when the other player shoots, you loose. The game can be configured by how many points the player must get before declaring a winner. If the maximum rounds are reached the player with the highest points wins. You must reload your gun before you can shoot. If both players shoot at the same time they both get a point. 007.java will also feature a 1vPC or 1v1 option and a loop asking the player(s) if they want to play again. 007.java will run in a simple console with images being displayed each time a player makes a move.

**Diagrams:**



1. User will enter “computer” or “another player” or “player” to start the game.

1b) The user will enter an integer value of how many rounds they wish to go up to.



1. Player 1 will always be the main player if versing a computer. Players will type their move: “Shoot”, “reload”, or “block”. If player is versing a computer, a random number will generate which will represent their move and the bottom line won’t appear.



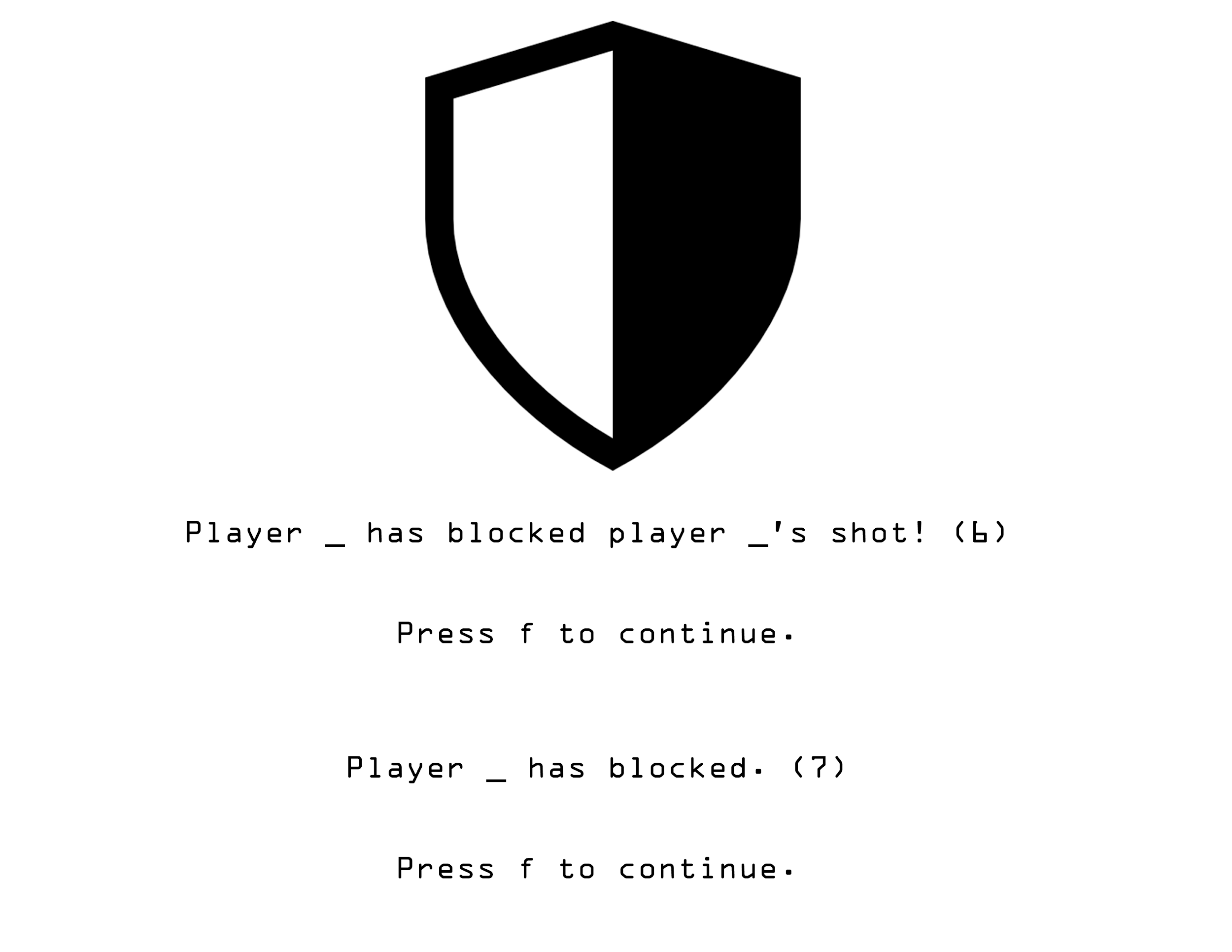
3a) If player blocks the shot this line will appear and the user will have to press f to continue.

3b) If the player does not block this line will appear and the score will be updated. If both players shoot both of them will die and points will be added to both their scores.

4) If the points reach the threshold, this line will appear telling the user to input “yes” or “no”.



5) Player will be updated on their ammo count when they reload. They will have to press f to continue to the next screen.



6) If a player fires and the other player blocks, this line will appear and the user will have to press f to continue.

7) If the player chooses to block but no other player has shot their weapon, this line will appear and the user will have to press f to continue.

**Top-Down Design:**

* One main method which will display if player 1 or player 2 wins and uses a do while loop to check if the player wants to play again.
  + **public static void main;**
* Two sub-main methods where one will contain the game for 2 players and one for one player.
  + **public static boolean playerVsPlayer(win);**
    - This method will hold all the next methods which are part of the game where the player versus another player. It will take the boolean value of win and return a false boolean value if player 2 wins and will return true if player 1 wins. The method will loop until a player wins.
  + **public static boolean playerVsAI(win);**
    - This method will hold all the next methods which are part of the game where the player versus the computer. It will take the boolean value of win and return a false boolean value if the computer wins and will return true if player 1 wins. The method will loop until a player wins.
* 1 for each combination (block + shoot, block + reload, block + block, shoot + reload, shoot + shoot, reload + reload)
  + **public static void shootBlock();**
    - This method will contain the commands which displays the shoot and the block screen for each player.
  + **public static int shootShoot(score);**
    - This method will contain the commands which displays the shoot and the shoot screen for each player and when player 1 and player 2 gets a point. It will also return an int value for the score to be added into an array in either the playerVsPlayer method or playerVsAI method.
  + **public static void shootReload();**
    - This method will contain the commands which displays the shoot and reload screen for each player and the ammo count for the player who reloaded.
  + **public static void reloadReload();**
    - This method will contain the commands which display the reload screen for both players and the ammo count.
  + **public static void reloadBlock();**
    - This method will contain the commands which displays the shoot and the block screen for each player.
  + **public static void blockBlock();**
    - This method will display the block screens for both players.
  + **public static int random(input);**
    - This method will generate a random integer from 1-3 for when the player is versing a computer. This random number will determine the move the computer makes.

**Pseudocode:**

public static void main(String[] args) {

boolean playAgain;

Do {

//input

Print(“Would you like to play against the computer or another player?”;

String answer = sc.next();

Print(“How many rounds do you want to go up to?”;

Int rounds = sc.nextInt();

//variables

boolean win = true;

if(answer.equalsIgnoreCase((“the computer”) || (“computer”))

**playerVsAI(win);**

else(answer.equalsIgnoreCase((“another player”) || (“player”))

**playerVsPlayer(win);**

if(win == true)

Print(“Player 1 wins!”);

Else if(win == false)

Print(“Player 2 wins!”);

Print(“Would you like to play again?”);

playAgain = sc.next();

If (playAgain.equalsIgnoreCase((“yes”))

Boolean playAgain = true;

Else

Boolean playAgain = false;

Break;

} (while playAgain == true);

}

**playerVsAI(win);**

* All the other methods are nested in this method and will be called inside this method instead of the main method. Other than **random(input);,** both methods will nest the same methods.

**playerVsPlayer(win);**

* This method also nests all the other methods and will be called inside this method instead of the main method.

**Changes**:

* added 1 method: **f();**
  + Method prints f to continue and waits until the user enters f to continue.