Who can Build the Game Bulldog Better AI or Human?

Sarah Lawrence, University of Southern Maine 02/11/2025

Bulldog Rules

Each turn consists of a player repeatedly rolling a six-sided die until either a 6 is rolled or the player decides to end their turn. If the player rolls a 6, their turn is over, and they score nothing for the turn. Control then passes to the next player. If a number other than six is rolled, the

number is added to the player's turn score. If the player rolls any number other than a 6, the number is added to their score for that turn. If the player wants to continue, they roll the die again, going through the same processes as above. If the player chooses to end their turn, they add their accumulated turn score to their overall score and control passes to the next player.

AI Code Description

The AI code was fully completed in 10 minutes, was almost entirely independent, the strategy was that if there was a problem, it was used to ask for a solution. The AI code only needed help from a human once for a file error. The style of AI code is simple but effective. For example, we will evaluate the code below.

AI code PlayerFifteen

```
public class FifteenPlayer extends Player {

   public FifteenPlayer(String name) {
      super(name);
   }

   @Override
   public int play() {
      int turnScore = 0;
      while (true) {
         int roll = (int) (Math.random() * 6) + 1;
         System.out.println(getName() + " rolled: " + roll);

         if (roll == 6) {
```

```
            System.out.println("Rolled a 6! Turn over with 0 points.");
            return 0; // turn ends with no score
        }

        turnScore += roll;
        System.out.println("Current turn score: " + turnScore);

        if (turnScore >= 15) {
            System.out.println(getName() + " ends turn with score: " +
                turnScore);
            return turnScore;
        }
    }
  }
}
```

This code has some notable qualities: the variable names are concise, the while loop is straight forward, and each if statement has a simple print line and return. The style of commenting doesn't follow what was required (Kettering Style) but no prompts were made to fix it. The output was just as simple as seen below.

AI FifteenPlayer and WimpPlayer Output

```
Enter the number of players: 2
Enter name for player 1: Player1
Choose player type (HumanPlayer, RandomPlayer, FifteenPlayer, UniquePlayer, Wimp
Enter name for player 2: Player2
Choose player type (HumanPlayer, RandomPlayer, FifteenPlayer, UniquePlayer, Wimp

Player1's turn:
Player1 rolled: 2
Current turn score: 2
Player1 rolled: 6
Rolled a 6! Turn over with 0 points.
Player1's total score: 0

Player2's turn:
   Player Player2 rolled 5 and chose not to continue, scoring 5 for the turn.
Player2's total score: 5
```

The options at the beginning "(HumanPlayer, RandomPlayer, FifteenPlayer, UniquePlayer, WimpPlayer)" make it easier for the player to know what to pick. The game output was simple, making it easier for players to know their current score and total score. It has great spacing, and each

line is short but still helpful for the player. However, I do wish there was something for new players to know what each player type does. Overall, the AI did a good job making the output player friendly.

Human Code Description

The Human code was fully completed in 5 hours, entirely independent, and the strategy was to code a small workable part and then debug. The style of the Human code was more complex than the AI code. For example, we will evaluate the code below.

Human code PlayerFifteen

```
/* Sarah Lawrence */
/* COS 420, spring 2025 */
/* Programming Assignment 1 */
/* Fifteen Player class: The Game Bulldog */

public class FifteenPlayer extends Player {
  // Default FifteenPlayer.
  public FifteenPlayer () {
    this("Fifteen");
  }
  // Constructor new FifteenPlayer object.
  public FifteenPlayer (String name) {
    super(name);
  }

  public int play() {
    boolean continue_g = true;
    int total_score = getScore();
    int save_roll = 0;

      // Choose to continue the turn until a score of at least 15 is
         reached.
    while (continue_g) {
      int roll = (int) (Math.random()*6 + 1);
      total_score = total_score + roll;
      System.out.print(" Player " + getName() + " rolled " + roll );

      // If score = 104 end game.
      if (total_score >= 104){
        return total_score;
      }
      // If 6 score for round = 0.
      else if (roll == 6) {
```

```
          roll = 0;
          System.out.println(" and scored 0 for the turn.");
          return getScore();
        }
        // Keep going until the roll is 15 or more.
        else {
          save_roll = save_roll + roll;
          if (save_roll =< 15) {
            System.out.println(" and chose to continue, the roll "
              + roll + " will be add makeing their total " + total_score +
                " for the turn.");
          }
          else {
            total_score = total_score + roll;
            System.out.println(" and chose not to continue, scoring "
              + total_score + " for the turn.");
            return total_score;
          }
        }
      }
      return 0;
    }

}
```

Like the AI code, this code has some notable qualities: the variable names being complex, the while loop is long and complicated, there are too many if else statement, and there are unnecessary lines.
For example:

```
if (total_score >= 104){
          return total_score;
        }
```

Some lines could have been simplified as well.
For example:

```
total_score = total_score + roll;
// could have been
total_score += roll;
```

The style of commenting doesn't follow what was required (Kettering Style) but there was more commenting than the AI code. The output was just as complex as seen below.

Human FifteenPlayer and HumanPlayer Output

```
Welcome to Bulldog!
```

```
How many players will be playing?
2
Player 1 What player would you like? (Type (Player_Info) to see the options)
Human
Player 1 has picked to be a human player.
Player 2 What player would you like? (Type (Player_Info) to see the options)
Fifteen
Player 2 has picked to be a fifteen player.
Let's Play!
   Player Player 1 rolled 5 Woud you like to continue? (Yes or No)yes
 and chose to continue, the roll 5 will be add makeing their total 5 for the turi
   Player Player 1 rolled 3 Woud you like to continue? (Yes or No)no
 and chose not to continue, scoring 8 for the turn.
   Player Player 2 rolled 3 and chose to continue, the roll 3 will be add makein
```

In the beginning, options were okay. Personally, the option to view the Player-Info is helpful for new players that don't know the game. But not having a list for users to copy/paste like the AI code had is bothersome. The game output is too long. Taking advantage of proper spacing and shorter statements would have benefited this. There were some little errors, like misspellings and improper capitalization. Overall, the Human code did a good job on some parts and could have done better on others to make the code more player friendly.

Conclusion

Overall the AI took less time to build, was mostly independent, and had great code and output quality. The only issue was that the AI did not have any commenting style. The human code took longer to build, was independent, had a less than ideal code and output quality. The Human code did have a good player menu feature for new players. If I had to do this assignment again, I would work on three things. First is working on implementing better spacing for the output. Second is making the code cleaner. Third having better commenting style. In the end, I learned how to better my code for future projects.