# Final Project P2 (Text-Based)

# Project Design Doc

**Tej Prattipati - tprat1s**

**USES TEXT-BASED SCORECARD****

**DID NOT USE SCORECARD INTERFACE - Based on Email Conversation: Determined it wasn't necessary, nor useful.**

**More in-Depth Descriptions May be Found inside programs as method or Class Headers as they were done after implementation***

**Note**: A lot of this document was completed/redone after the program was completed to ensure consistency. However, the project was planned prior to beginning coding, just not as thorough as this document would suggest (All methods and attributes planned)

## Overview

This doc shows the attributes, methods, and Explanations of each Class as you would see in a Class header/method header. It also explains how each class interacts with others directly.

## Class Descriptions

### Die

Represents a single die used in the game. Responsible for rolling the die and keeping track of its current value.

**Attributes:**

- dieVal (int): Current value of the die, which ranges from 1 to 6.

**Constructor:**

- Die(): Creates a new die and rolls it right away to give it an initial value.

**Methods:**

- int roll(): Rolls the die to generate a random value between 1 and 6, updates dieVal, and returns the new value.
- int getValue(): Returns the current value of the die without rolling it again.

**Relationships:**

- **YahtzeeHand**: Uses Die objects to represent the dice in the player's hand.
- **YahtzeeScore**: Scores for different categories depend on the values of the Die objects in a given hand (YahtzeeHand).

## YahtzeeHand

Manages a collection of five dice. It provides methods to roll all the dice, roll specific ones, and show their current values.

**Attributes:**

- dice (Die[]): Array of 5 Die objects representing the dice in the player’s hand.

**Constructor:**

- YahtzeeHand(): Sets up the hand with 5 dice and rolls them all initially.

**Methods:**

- void rollAll(): Rolls all five dice at once.
- void roll(int dieNumber): Rolls a specific die, identified by dieNumber.
- void changeHand(): Allows player to choose which dice to re-roll.
- String showDice(): Provides a string showing the current values of all dice.

**Relationships:**

- **Die**: Uses Die objects to manage and roll the dice.
- **Game**: The Game class interacts with YahtzeeHand to handle dice during the game.

## ScoreCard

### USES TEXT-BASED SCORECARD***, not GUI

Keeps track of the scores for each of the 13 Yahtzee categories. It also shows which categories have been used and calculates the final score. Uses arrays to track this.

**Attributes:**

- categoryScores (int[]): An array storing the scores for each category.
- used (boolean[]): An array indicating if a category has been used.
- names (String[]): An array of category names.
- int totalScore: tracks the current total Score of all categories

**Constructor:**

- ScoreCard(): Initializes the arrays for category scores and usage status.

**Methods:**

- void displayScoreCard(): Shows the scorecard with category names, scores, and usage status.
- void updateScore(int category, int score): Updates the score for a specific category and marks it as used.
- boolean isCategoryUsed(int category): Checks if a category has been used.

- int getCategoryScore(int category): Gets the score for a specific category.
- int getFinalScore(): Calculates and returns the total score from all categories.
- int getScoredCategories(): Counts how many categories have been scored.

**Relationships:**

- **Game**: The Game class uses ScoreCard to keep track of and display scores throughout the game.

## YahtzeeScore

Calculates scores for different categories based on the dice values. It's used to determine the scores for the upper and lower sections of the game.

**Attributes:**

- dice (Die[]): An array of dice whose values will be used to calculate scores.

**Constructor:**

- YahtzeeScore(Die[] dice): Initializes with an array of dice to be scored.

**Methods:**

- int getUpperScore(int category): Calculates the score for a specific upper section category.
- int scoreThreeOfAKind(): Calculates the score for three of a kind.
- int scoreFourOfAKind(): Calculates the score for four of a kind.
- int scoreSmallStraight(): Calculates the score for a small straight.
- int scoreLargeStraight(): Calculates the score for a large straight.

- int scoreFullHouse(): Calculates the score for a full house.
- int scoreYahtzee(): Calculates the score for a Yahtzee.
- int scoreBonusYahtzee(): Calculates the score for a bonus Yahtzee.
- int scoreChance(): Calculates the score for a chance.

**Relationships:**

- **Die**: Uses Die objects to determine scores based on their values.
- **Game**: The Game class utilizes YahtzeeScore to calculate and assign scores during the game.

## Game

Runs a full Yahtzee game. It handles turns, dice rolling, scoring, and displays the scorecard. Runs 13 turns, and reports the final score of the game. Interacts with the user along the way

**Attributes:**

- finalScore (int): Keeps track of the final score of the game.
- scoredCategories (int): Records how many categories have been scored.
- turnNumber (int): Tracks the current turn number.
- card (ScoreCard): An instance of the ScoreCard class to manage scoring.

**Constructor:**

- Game(): Sets up the game and initializes the scorecard.

**Methods:**

- void displayScoreCard(): Shows the current state of the scorecard.
- void runGame(): Manages the overall flow of the game.
- void turn(YahtzeeHand hand): Handles a single turn of the game.
- void rerollingSequence(YahtzeeHand hand): Manages the rerolling of dice.

- void scoringSequence(YahtzeeHand hand): Handles scoring for the dice in a turn.
- int getFinalScore(): Returns the game's final score.

**Relationships:**

- **YahtzeeHand**: Uses YahtzeeHand to manage dice during gameplay.
- **YahtzeeScore**: Uses YahtzeeScore to compute scores based on dice values.
- **ScoreCard**: Uses ScoreCard to track and display scores.

## Match

Handles a series of Yahtzee games, allows multiple games to be played in sequence until the user decides to end the match.

**Attributes:**

- endMatch (boolean): Indicates if the match should end.
- trueOrFalse (String): User input to decide whether to play another game.
- gameNumber (int): Tracks the current game number within the match.
- matchScore (int): Keeps track of the cumulative score across all games in the match.

**Methods:**

- public static void main(String[] args): Manages the loop for running multiple games and handling user input to continue or end the match.

**Relationships:**

- **Game**: Uses Game to run and manage individual games within the match.

**Possible Addition:** Add upper section bonus. Not in the requirements, but it might be a nice incentive as an extra 35 points to be careful when doing upper scoring.