THE GAME OF SET

1] Basic Rules:

Set is a card game in which players find three cards with four different features.
Features include: color, symbol, number, and symbol. Each feature has three values. There are a total of 81 cards (3^4). On the table, 12 cards are dealt in a 3 by 4 layout. A group of three cards is a set if the values are either all the same or all different with the three cards. If the card meets the phrase “Two are X and one is not,” then it is not a set. If the three cards the player selects form a set, then a player is rewarded a point. Bonuses are awarded for consecutive correctness of sets. If it is not a set, the player is penalized a point. The game is over when there are no more cards left in the deck to deal, or there are no more sets remaining in play. The player who gets the most points is declared the winner. If there is a tie, then all players with the highest score are declared as the winner.

Each card consists of 4 attributes:

- Color (red, green, or purple)
- Shape (diamond, squiggle, or heart)
- Shading (solid, empty, or striped)
- Number (one, two, or three)

2] Requirements

Language Used: Ruby (2.7.1)
OS: Linux (Ubuntu 20.04)

1. If you are downloading the tag, and are in the main repository directory, open the project 1 directory using ‘cd project_1’
2. Run ‘sudo apt install libsdl2-dev libsdl2-image-dev libsdl2-mixer-dev libsdl2-ttf-dev’
3. Run ‘gem install colorize’
4. Run the command ‘bundle install’ to install the ruby2D gem
5. Now, to play the game run ‘ruby main.rb’ruby

Ruby2D- Utilized to print the cards with its easy ASCII image styling.

3] Guidelines of the game

- Once the game is opened, you are prompted to enter “Start” or “menu”
- Menu- Lists other options for text input: “stats” or “scores”
- Scores- Prints a list of high scores from previously played games.
- Stats- Lists various stats from games previously played
    - Includes: Number of games, total time spent playing in minutes, average game time length, number of deals, number of extra deals (deals going over 12 cards), the odds of a set not appearing (in decimal probability form), number of sets

correctly guessed, number of sets, incorrectly guessed,and the average accuracy (decimal/probability)

- Start- Begins the game, along with a spy guessing song over the game.
- The game prompts the user to enter player names, and then enter an empty line to begin the game
- You cannot start with a blank line and you cannot repeat names, you will be prompted to enter again

- ‘cpu’ - If a player enters ‘cpu’, the game will include a computer player
    - They are then prompted for a difficulty level, correlating with how often the computer player guesses and how often the computer guesses randomly vs. correctly
        - 0 - Guesses every 22 seconds, and has a 33% chance of being random
        - 1 - Guesses every 18 seconds, and has a 20% chance of being random
        - 2 - Guesses every 10 seconds, and has a 14% chance of being random
- The cards are then printed on screen with their symbol, shading, and number of symbols on the card. If you would rather read the attributes, they are listed under the card along with a number correlating to the input for guessing a set.
- To make a guess, you type in your name, so that in the case of multiplayer, the game knows who is making the guess. You then have 10 seconds to type the three card numbers with spaces between (# # #) or the game will guess the first three cards for you.
- If the user attempts to type in ‘cpu’, the game will reject it and ask them again.
- If the user enters anything other than a listed name, they will be prompted for a new name.
- After a few seconds, a prompt will come up for the user to ask for a hint
- ‘hint’ - If entered by the user, the player is given one card that is contained by a set\
    - Every successive hint prints another number, so after 3 hints, the user has a full set to guess.
- If a computer player is present, when they make their guess, the screen will print whether “beep boop:” and their guess.
- When a user or computer guesses, the validity of that guess (correct/incorrect) will print and the new updated scores will print.
- Scoring system - A correct guess is 1 point, incorrect is -1 point
    - Every successive guess increases the points earned by 1 point. So the first correct guess is 1 point, the next is 2, then 3, etc. If the user guesses incorrect the next correct guess will be 1 point again.
    - An incorrect guess results in your score going down by 1.
- When a correct set the cards are removed from the active 12 cards and 3 more are pulled from the deck
- The game ends when there are no longer sets available to choose (no combinations or < 3 cards)
- The game prints a congratulations message for the winner and a neat victory jingle plays.

- The game asks if you would like to add the score to the all-time high score list, which you can accept/decline with Y/N
- The game then ends, returning to the command line

4] How our implementation works

- Card.rb- Contains the card class, which holds the card attributes, number, symbol, color, and shading. Its functions are used to print the cards to the string in an ASCII design and in print, and to check if three cards are a set.

- Deck.rb-This file consists of the deck class which is used to make a well shuffled deck of 81 cards, each different in at least one attribute of the card. This class also contains functions to draw the next card in the deck, return a colored card and check if a set occurs.

- Game.rb - The game class has an array of cards that represents the active 12 cards on the table, a deck with the rest of the available cards, an array of players, and a boolean correlating to if the game is still playable. The class handles adding players, dealing cards, finding a set from the list of active cards, checking if a set exists in the deck + active cards (if the game can still be played), removing cards from the active cards, printing hints for the player, and checking the players guess for a correct set

- Player.rb- Contains a class for the player, and a subclass for a computer player. The class held the players name, points, and consecutive correct guesses and handled score functionality. The computer player subclass had the same attributes, but also held a difficulty, wait time, and their next guess of cards. The computer player has functions to make a choice of cards based on difficulty (impacted time and rate of incorrect guesses), and print that guess for the user.

- Scores.rb- Contains the scores class which tracks the list of all time scores for the game. The class contains functions to add a new player to the list of scores, and display the list of all-time top scores. This was done with a text file that the scores class would read//write to when they needed to see/update the scores.

- Stats.rb- Contains functionality to calculate the various statistics listed in part 3, and then writes those over the old stats in the stats text file.

- timer.rb-This file consists of a timer, which will tell the player(s) how long he/she (they) took to finish the game.

- main.rb-This file contains the functional game. It adds a function for checking valid set input and displaying the cards to the terminal. Being what runs to play the game, it contains the text prompts for starting and playing the game, the music that plays, and utilizes a game object to contain all the various attributes of the cards, the player(s), and their respective scores.