Description:

The game we will design is a local-multiplayer implementation of Yahtzee. Each game will be played with five 6-sided die and adhere to the normal Yahtzee rules. At the beginning of each game the player will get to choose the number of players that will be playing the game and each player can choose a name. On each players turn; the player will first go through the roll phase. In the roll phase they can roll the die 3 times. Between each roll the player may "lock" some of the die and only roll the "unlocked" ones. The player may also unlock previously locked die on the final roll. At the end of each players rolling phase, they will be able to choose which lines to score on and any bonus chips will be calculated and displayed per Yahtzee rules. This will be done either through automatically advance. This consists of one turn.

Each player will rotate turn by turn and the GUI will "reset" at the beginning of each turn. At the beginning of each turn the scorecard for the current player will be displayed with a name to indicate the current players turn. The players will keep rotating through turns until one player's scorecard is full. Once this condition is met, the game will end. Once the game ends the winning players scorecard is displayed on screen with some indication as to who won.

If time, we will add some extra features. These might consist of prompting the user for different die and hand size options at the beginning of the game or allowing more rolls per turn. More features may be thought of later.

a popup window or section on the main window. Once scored the turn will

Functional Requirements:

Part	Title
Priority	High, med, low
Inputs/needs	Describe inputs / needs for function, include sources, valid ranges, special cases
Operators/actors	Which of UML and/or people playing are involved
Output	What values, states or conditions shall be output or set by this requirement

Part	GameControl
Priority	High
Inputs/needs	Inputs: Players and their states, the GUI and their states, should be valid inputs of dice scores, dice rolls, and responses from GUI. No special cases
Operators/actors	Player, ScoreCardWindow, MainMenuWindow, RoundWindow, EndGameWindow

Output	End values will only be Booleans on whether functions and classes managed successfully complete

Part	Player
Priority	High
Inputs/needs	A hand, dice scores, a score card; functions needed include getting a total score from the score card, as well as getting the dice in their hand. Special cases include a preset dice roll, as well as bonus scores included in score card
Operators/actors	ScoreCard, Hand, GameControl
Output	The total score of the player, whether it is the winner, the dice held in it's hand

Part	ScoreCard
Priority	High
Inputs/needs	Dice scores, the possible scores and actual scores, given from scoreCardLine as well as Player, special cases are the lower sections, and valid ranges should only be with integer scores and String sections
Operators/Actors	ScoreCardLine, Player
Output	The total score of a player, as well as the possible scores

Part	ScoreCardLine
Priority	Medium
Inputs/needs	It needs whether its been chosen, the possible score calculated by the hand, and what its actual score is. Valid ranges are all the possible integer scores of a Yahtzee game, and special cases are when a user tries to choose a line that's already been chosen
Operators/Actors	ScoreCard
Output	The possible, and actual score, as well as a toString for easy showing

Part	Hand
Priority	High
Inputs/needs	Needs to know how many dice are in play, the sides of each dice, as well as scores from calculating. It must know individual section

	scores as well. Special cases would include multiple of the same dice, and other calculative instances.
Operators/Actors	Player, Die
Output	The dice that a player holds, as well as the scores of the hand.

Part	Die
Priority	High
Inputs/needs	Number of sides, the side that has been rolled, as well as what type of random to use to roll. Special cases are dictated by the number of sides and which side starts up
Operators/Actors	Hand
Output	The side that's been rolled, as well as a toString for easy showing.

Part	GameWindow
Priority	High
Inputs/needs	Needs to know the width and length of the desired frame. Is used to be a basic format for all other Views to implement for easier construction and handling
Operators/Actors	EndGameWindow, RoundWindow, MainMenuWindow, ScoreCardWindow, GameControl
Output	Nothing except any specific methods added to all children classes

Part	EndGameWindow
Priority	Medium
Inputs/needs	Needs to know the final score of the game, as well as who won the game. This is for displaying the correct winner and their score.
Operators/Actors	GameWindow, GameControl
Output	Nothing, after executing this window, the game should shut down

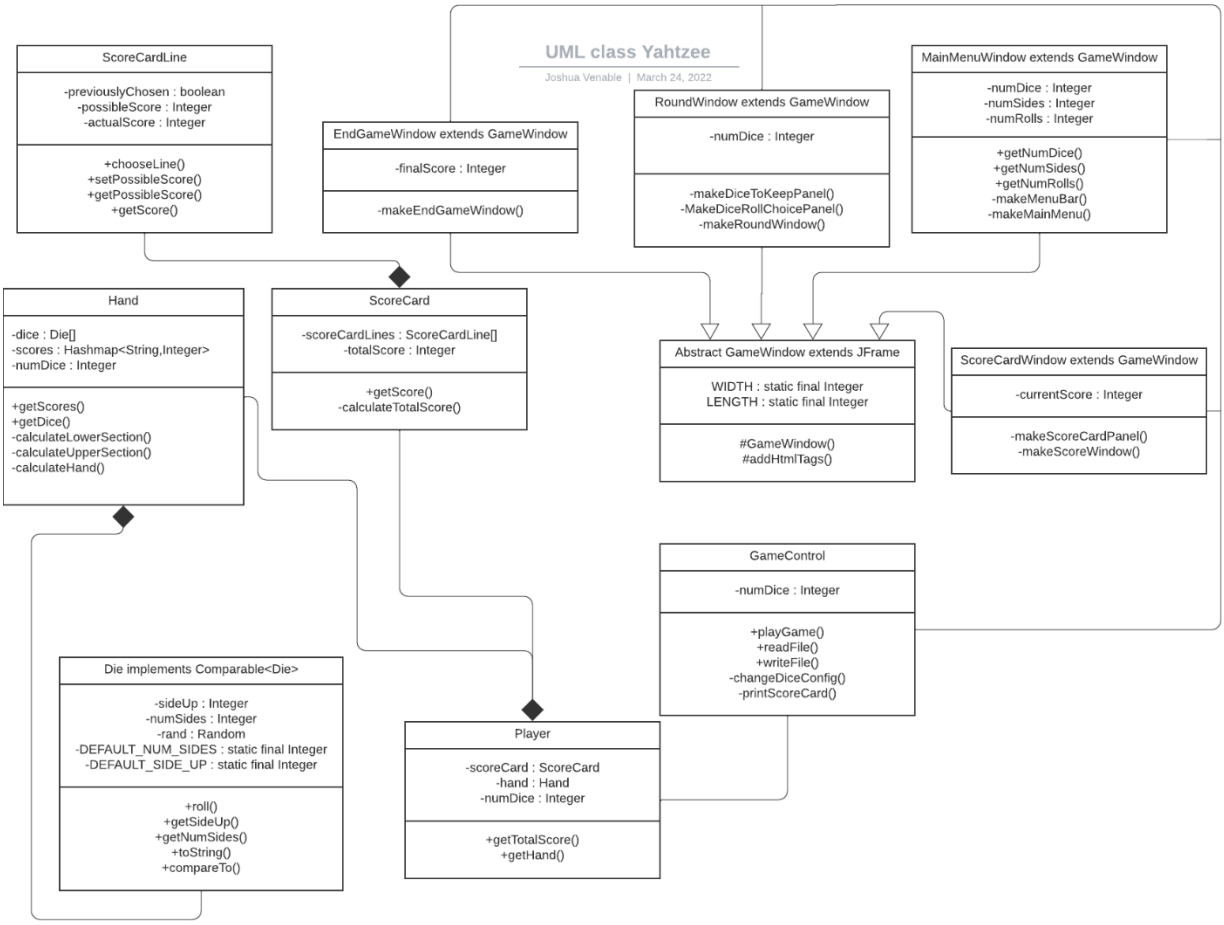
Part	RoundWindow
Priority	Medium
Inputs/needs	Needs to know the round its on, how much there is left to score, how many players, the dice amount, the dice sides, as well as the number of rolls each player gets. It needs to be able to roll the dice independently and do various methods controlling the dice
Operators/Actors	GameWindow, GameControl
Output	The dice and/or the score after each player is done rolling

Part	MainMenuWindow
Priority	Medium
Inputs/needs	It needs to know the number of dice, the sides on each die, and the number of rolls. It should also know how many players are playing. This is necessary for setting up the RoundWindow GUI and backend
Operators/Actors	GameWindow, GameControl
Output	The number of dice, sides, and rolls

Part	ScoreCardWindow
Priority	Medium
Inputs/needs	It needs to know the dice from the round, as well as the possible scores of all the sections. This is for handling the showing of the scorecard to the player as well as possible scores for choosing one
Operators/Actors	GameWindow, GameControl
Output	Nothing except a continuation of the game, an asynchronous setting of the possible score the user chose to the actual score will occur though.

UML:

Class Diagram:



GameControl: Controls the flow of the Yahtzee game, coordinating the GUI and the other classes, so that the individual classes don't have to talk to each other, reducing complexity

Player: A player class dedicated to holding information specific to each player that gets instantiated.

Hand: The 'hand' of dice that each player will have one of. Used to hold the dice and perform actions on it.

Die: The most basic component of the Yahtzee Program, dedicated to being able to set the number of sides and roll to get a specific 'side up'

ScoreCard: Another aspect that the player class will contain, dedicated to holding the player's scoring card and its lines, able to directly access the scores of the lines.

ScoreCardLine: A class dedicated to holding a specific 'line' of the scoring sheet, for example '1' for the dice with a roll of 1, or 'full house' for the scores and possible scores of that section.

GameWindow: An abstract class for other subsequent GUI classes to inherit and create a standard window/frame. It extends JFrame, so that all child classes are also JFrame and get all functionality that comes with it.

ScoreCardWindow: The GUI class specifically for holding the score card, able to showcase the score card of a specific player as it is at any moment in time.

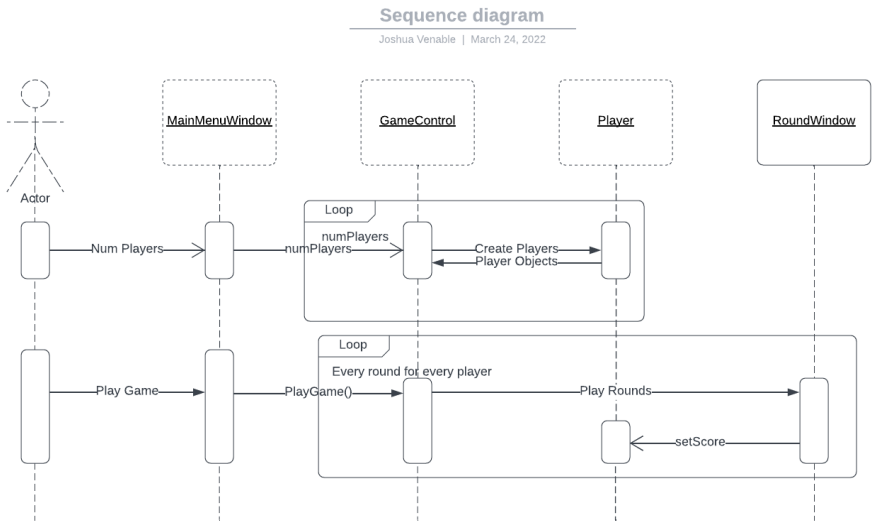
MainMenuWindow: The GUI class dedicated to showing the beginning of the game, enabling the player to change any possible configurations, and start the game.

RoundWindow: The GUI class built for showing the actual playing of a round of Yahtzee, able to keep track of the dice to keep from rolling, as well as the possible scores after all rolls are used up. Should be able to differentiate between players and print the possible scores of unchosen sections specific to each player.

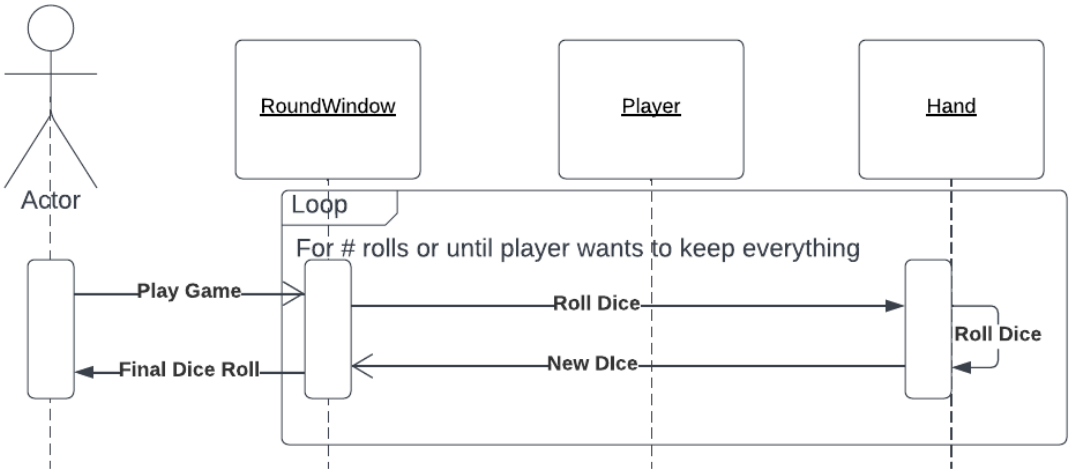
EndGameWindow: The final GUI class for when the game has been quit or force ended when everyone's score card is full. Showcases the final scores of all players, as well as the winner of the game.

UML Sequence Diagrams:

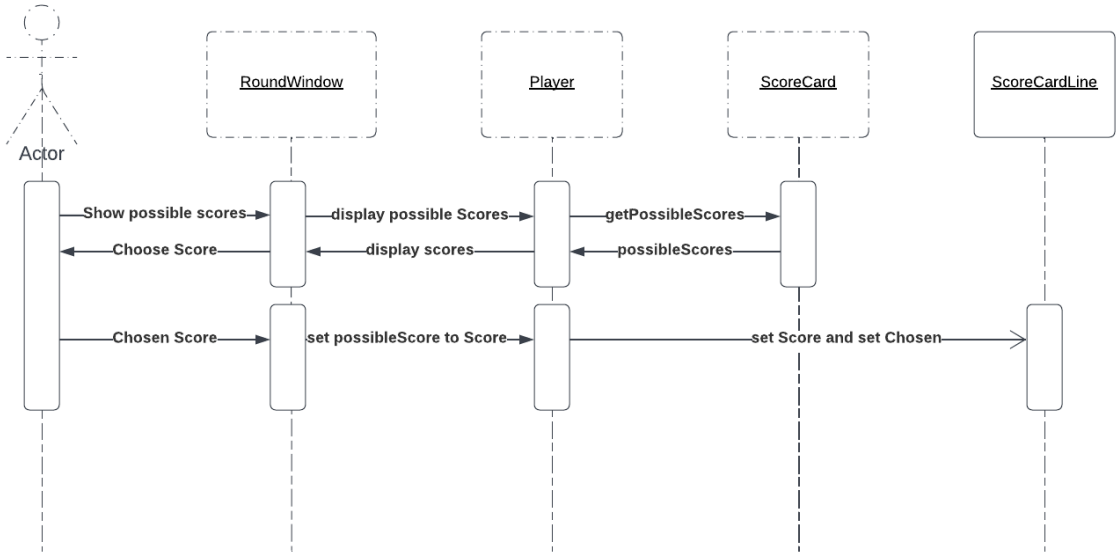
Main Menu → playing rounds of Yahtzee:



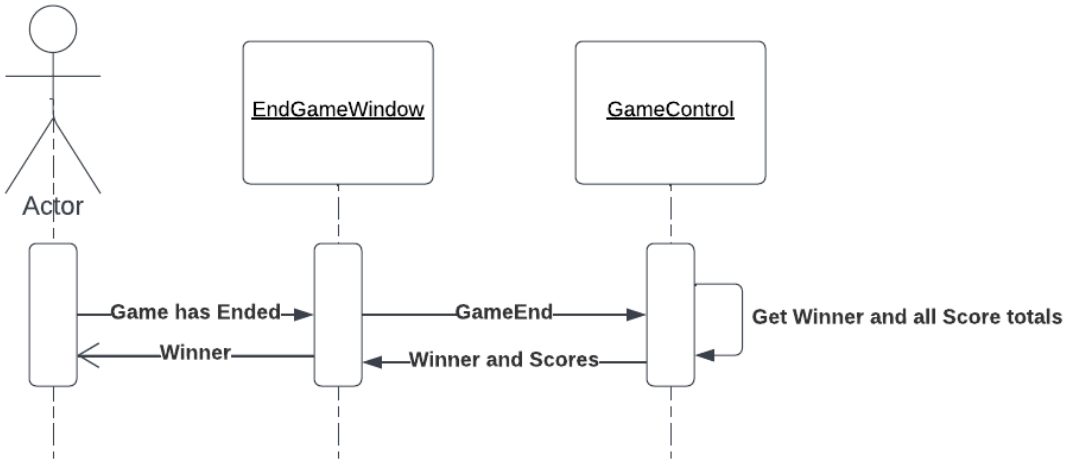
Playing round of Yahtzee:



Choosing Possible Scores to hold:



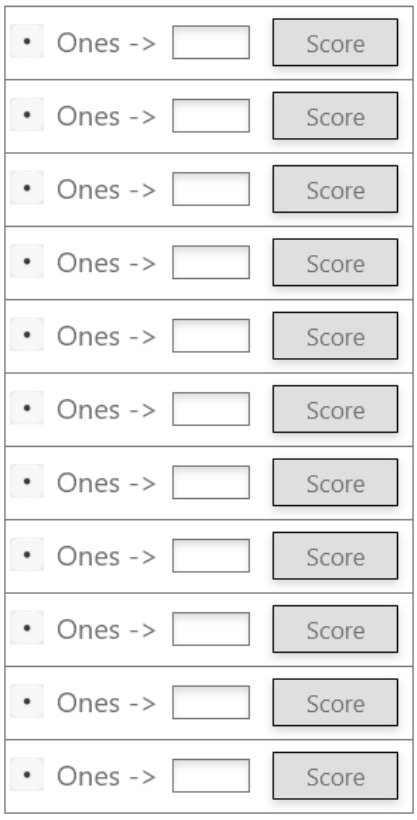
Game End:



UI Wireframe:

Main Window:

Scoring Popup Window:



Plan/Schedule:

Tasks to be completed: create Player object, create ScoreCard object, create Hand object, create GameControl object, Create GameWindow object and its extensions, create ScoreCardLine object, create multiple players, implement turns to cycle through multiple players, determining the winner, testing the objects and their functions.

- Create Player Object: Assigned: Maalik. Estimated hours: 3. Due date: 4/2/22
- Create ScoredCard object: Assigned: Joshua. Estimated hours: 3. Due date 4/2/22
- Create Hand object: Assigned: Tyler. Estimated hours: 3. Due date: 4/2/22
- Create GameControl object: Assigned: Maalik. Estimated hours: 3. Due date: 4/2/22
- Create GameWindow object and its extensions: Assigned Joushua. Estimated hours: 4. Due date: 4/2/22
- Create ScoreCardLine object: Assigned: Tyler. Estimated hours: 3. Due date: 4/2/22
- Create multiple players: Assigned: Maalik. Estimated hours: 3. Due date: 4/2/22
- Implement turns to cycle through multiple turns: Assigned: Joshua. Estimated hours: 6. Due date: 4/8/22
- Determining the winner: Assigned: Tyler. Estimated hours: 4. Due date: 4/8/22
- Testing the objects and their functions: Assigned Everyone tests the objects and functions they made. Estimated hours: 5. Due date: 4/17/22