CUP O' JAVA

Final Report

Hunter Banks

Connor King

James Vuong

Course: CPSC 224 - Software Development

I. Introduction and Project Description

The purpose of this document is to introduce the project, team members and their roles within the project, project requirements, project solutions, and more. The document further discusses the project's needs, along with details such as the priority level of each issue/feature. Following is a full discussion of the approach we used to meet the project's requirements. The next section discusses how we tested our project, how we implemented its features, and our future intentions for new features or improved code. Lastly, our glossary, references, and future work sections.

The goal of this project is to design a complete game of Yahtzee between 1 and 8 players playing against each other on a GUI using Java. The game will be played with normal Yahtzee rules and configurations (6 sided die, 3 rolls per turn, and 5 total die). Before playing starts, player(s) are given the option to change their name from “PlayerX” to whatever they please. The players will not have the option to change this later. The GUI will display only the current player’s scorecard and dice hand.

Throughout each player’s turn, they will be able to view their own scorecard. A player will roll their set of dice a maximum of three times before they are forced to choose a score. Players will have the option to choose which dice they would like to keep and which dice they would like to reroll. Once a player has finished rolling in a single turn, they will be given a score sheet of potential scores to choose from; the player will then have the option to choose the score they would like to keep. Once the first player has chosen their score to keep, the process will be repeated with the remaining players. Players will take turns choosing scores until each line on the scorecard has been filled. Repeats on the scorecard are not allowed (e.g. if Player1 chooses to keep the Full House score, they may not choose that score in the sequential rounds). After all players’ scorecards have been filled, the GUI will display all of the players’ scorecards and the player with the higher total score wins the game of Yahtzee.

A series of unit tests, integration tests, and user tests were used to ensure that Yahtzee was fully functional. We were able to confirm that the code delivered in terms of functionality by testing it in a variety of ways. Furthermore, the user tests aided in the development of a cleaner and easier-to-use user interface.

II. Team Members - Bios and Project Roles

Connor King is a business administration student studying with a concentration in management information systems and a minor in computer science. This is the fourth computer science class that he has taken. Previous computer science courses taken include Computer Science I, Computer Science II, and Introduction to Data Science. Connor’s skills include C++, Java, Python, MATLAB, Pandas, and APIs. For this project, Connor’s responsibilities included writing tests and writing part of the backend.

Hunter Banks is majoring in computer science with a minor in software application development. He is a third year student at Gonzaga University, and looking forward to graduating next year. Prior computer science experience includes programming in C++, Python, and Java. For this project, Hunter's responsibilities include project management, managing the graphical user interface, and optimizing the program to run on the graphical user interface.

James Vuong is a computer science student interested in software development, and data and information science. His previous projects include implementing Yahtzee HWs 1-4, Sleep and Study Correlation for his *Intro to Data Science* class, and other coding assignments from previous classes. James's skills include C++, Python, and Java. For this project, his responsibilities included helping implement the multiplayer feature, assigning different names to players, simple troubleshooting, and helping with basic unit tests.

III. Project Requirements

For this project, we wanted to create a game of Yahtzee that is true to the traditional game rules. This game will allow for multiple players, modification of each player's name, and choice of which scorecard line to use. The entire game will be played through a GUI, meaning no console input is needed.

Requirements:

1. Play an accurate game of Yahtzee, true to the game rules
2. Allows for multiple users to play in a single game.
3. Choice for which scorecard to use hand on
4. Game is played on a graphical user interface (GUI)
5. Player names can be customized

Part	Title
Priority	High
Purpose	Standard play is an accurate game of Yahtzee, true to the game rules
Inputs / Needs	Standard game is a default configuration of 5 dice, 3 rolls, and 6 sides on each die.
Operators / Actors	Involves the "Records" class to ensure the default configuration object is set.

Part	Title
Outputs	Accurate scoring and gameplay per Yahtzee instructions. Classes include, “Die”, “Lower Card”, “Upper Card”, “Main”, “Records”, “ScoreCard”.

Part	Title
Priority	High
Purpose	Allows for multiple users to play in a single game.
Inputs / Needs	Standard game is default 1 player, however up to 8 players can play.
Operators / Actors	Involves the “ScoreCard” class to ensure each player has their own score card.
Outputs	Accurate scoring and gameplay for each player per Yahtzee instructions.

Part	Title
Priority	High
Purpose	Yahtzee game is played on a graphical user interface (GUI)
Inputs / Needs	User must be presented with a GUI representing the game of Yahtzee.
Operators / Actors	This involves the “GUI” class which involves all the graphical elements.
Outputs	No console outputs for the user to see. All output should appear on the GUI.

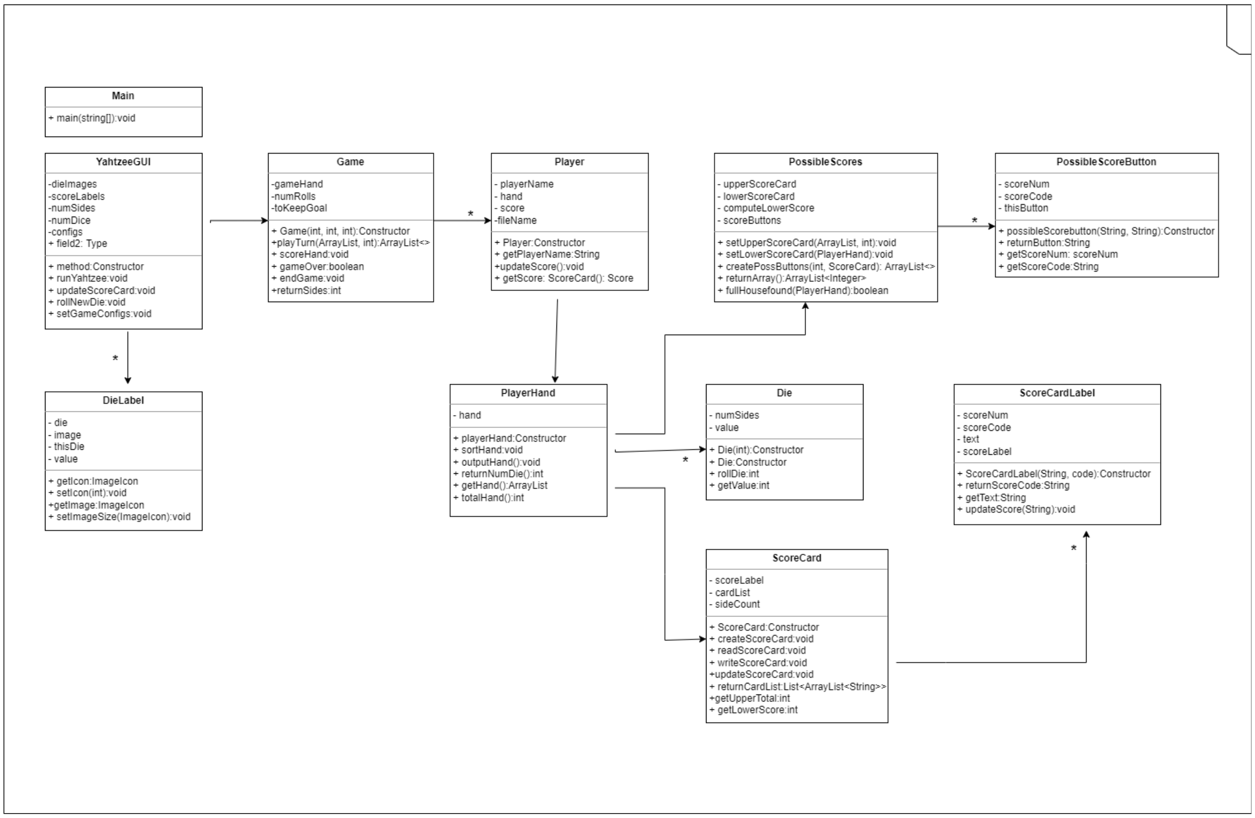
IV. Solution Approach

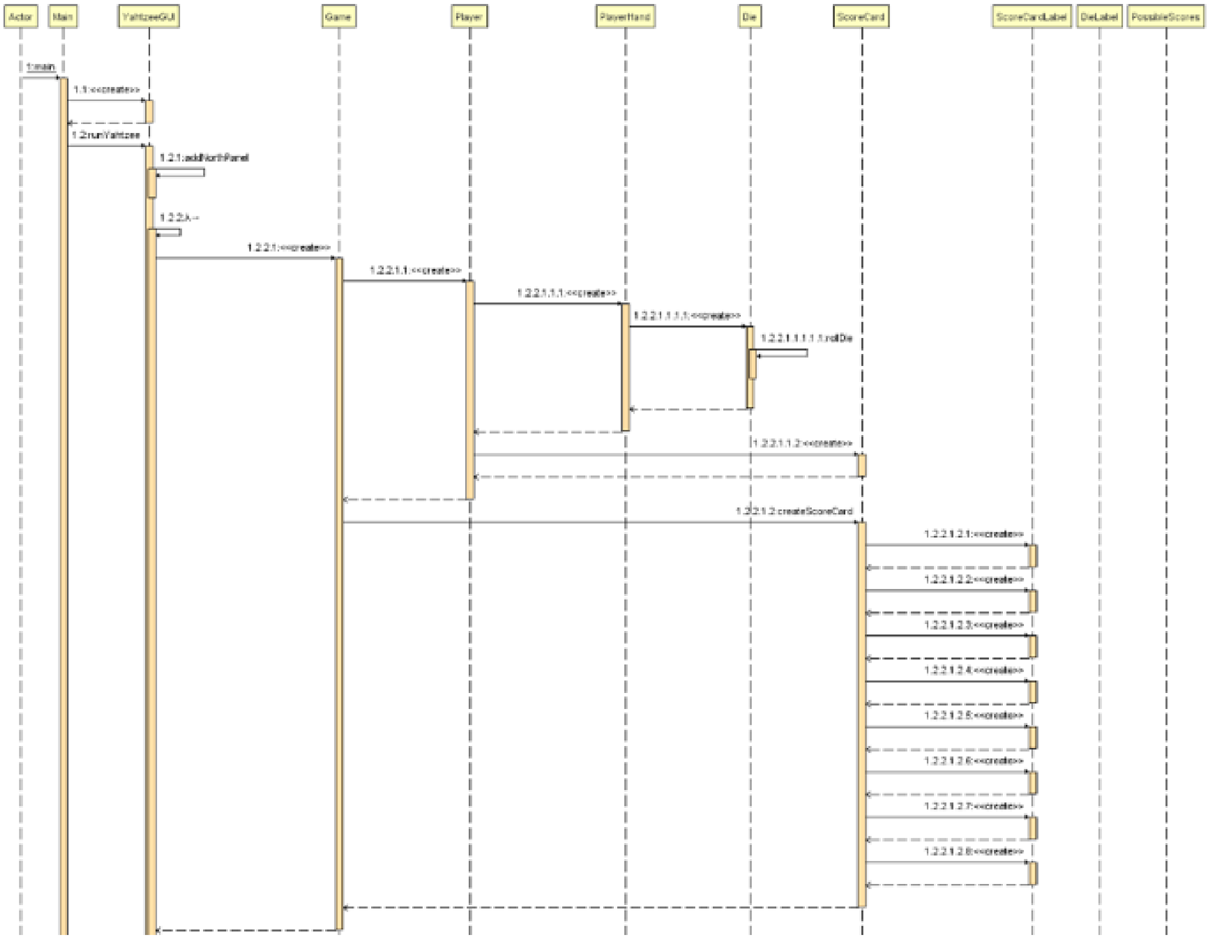
Major Components

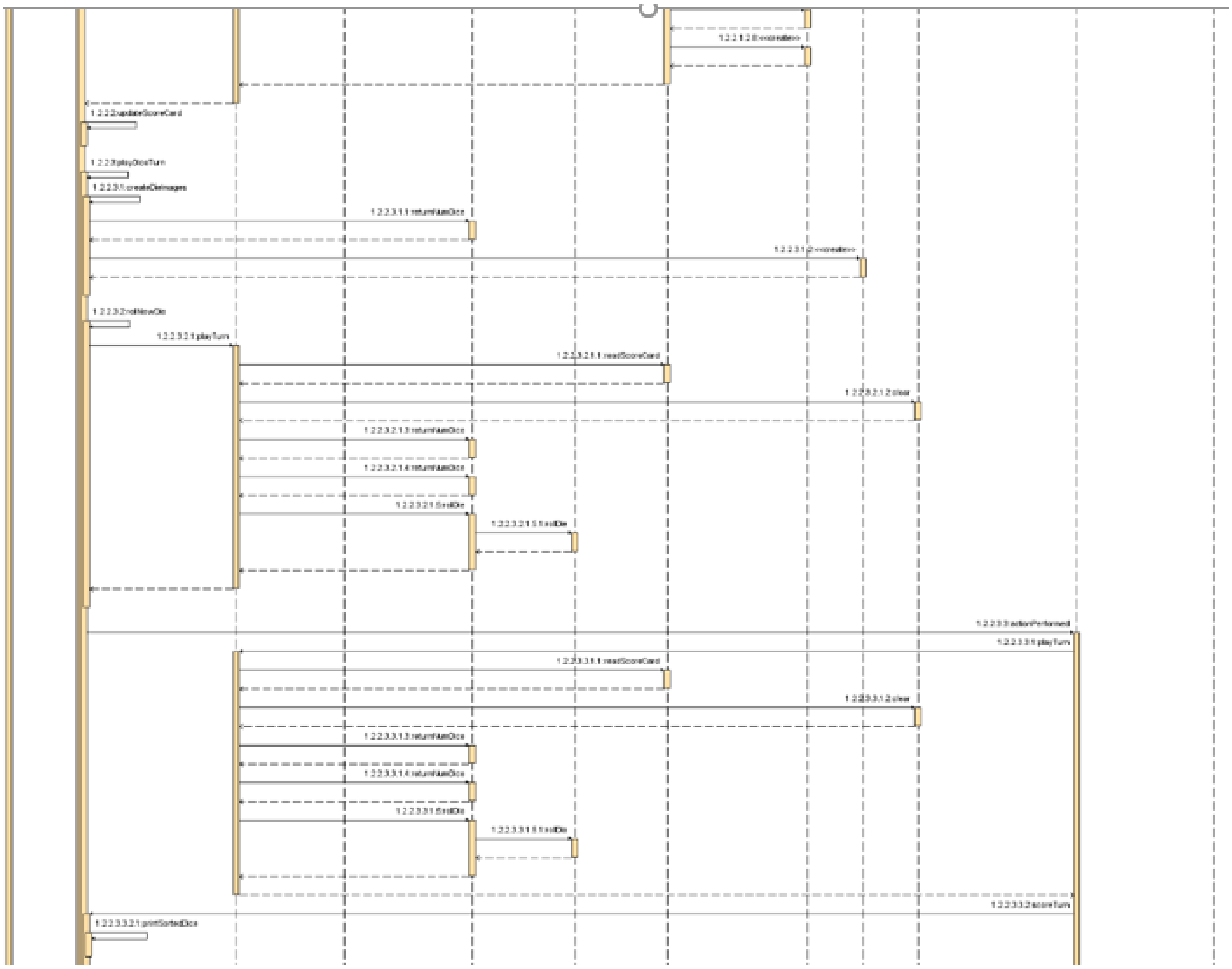
Our team wanted to design our game with a well organized class design when it comes to practicing object-oriented programming. We created a main class that only has the purpose of running the program. Our GUI class contains the components of the GUI and their functionalities. Our game class contains the functionality of the game of Yahtzee. This class adheres to the project requirements that we created at the beginning of this project. Our final major class is our player class, which contains the data for each player (names, scorecards, score lines used, etc.).

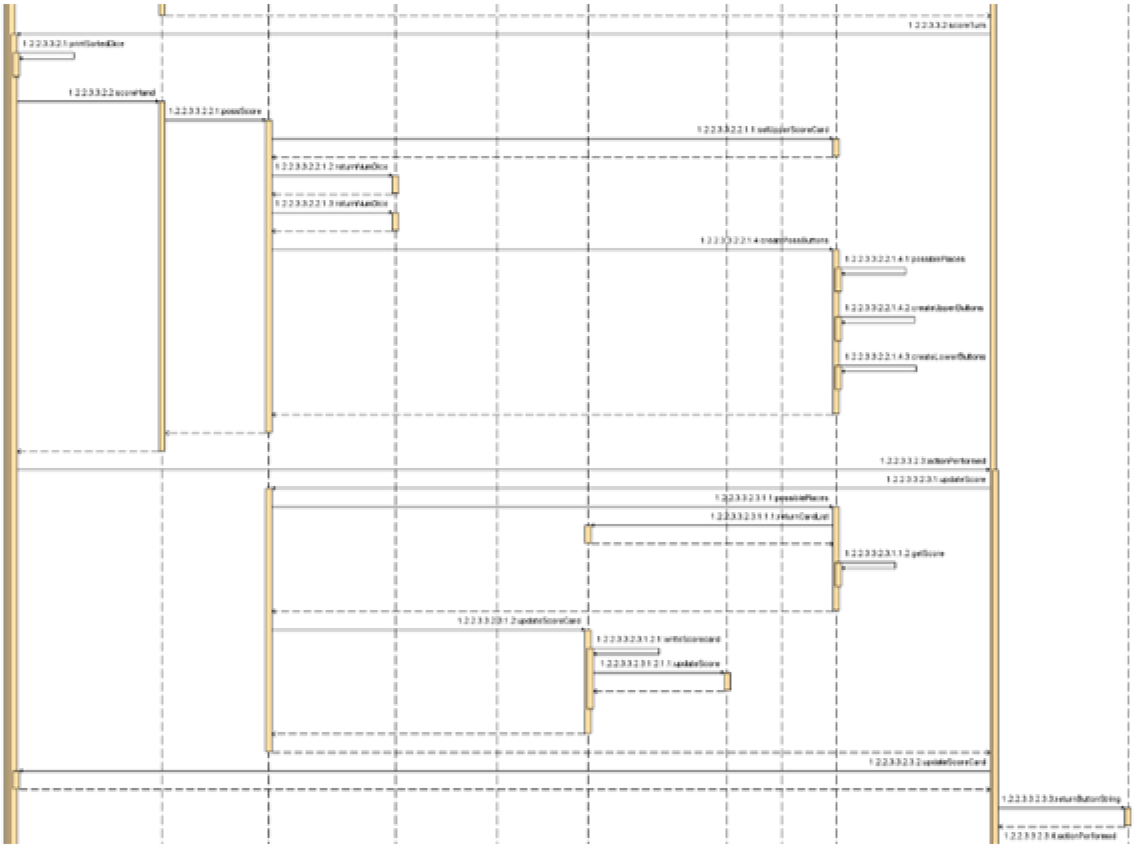
Game & GUI Focuses

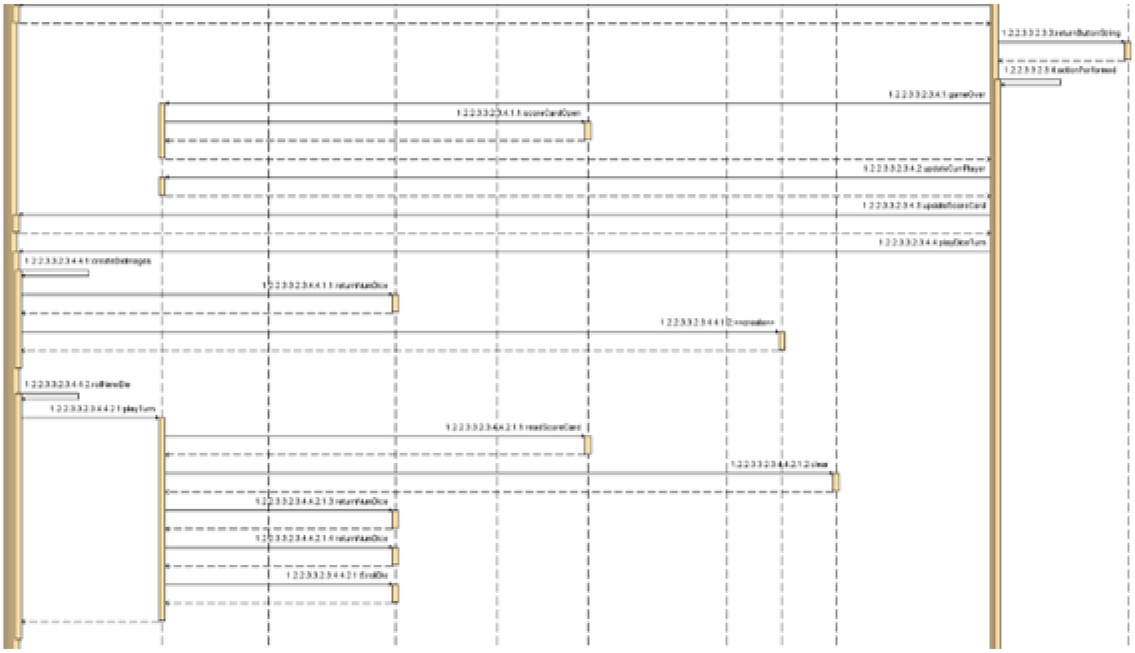
For our approach to game and GUI functionality, we used our project requirements as guidance. We focused on making our game function smoothly while players make choices throughout the game. Specifically, we wanted our game to function well with both single player and multiplayer games. This included displaying the correct information such as player names, player hands, and which player's turn is currently in process. Our focuses on the GUI were to have it be functional while making it look good. We wanted to make sure that the GUI was not too simple, but not overstimulating for the user.

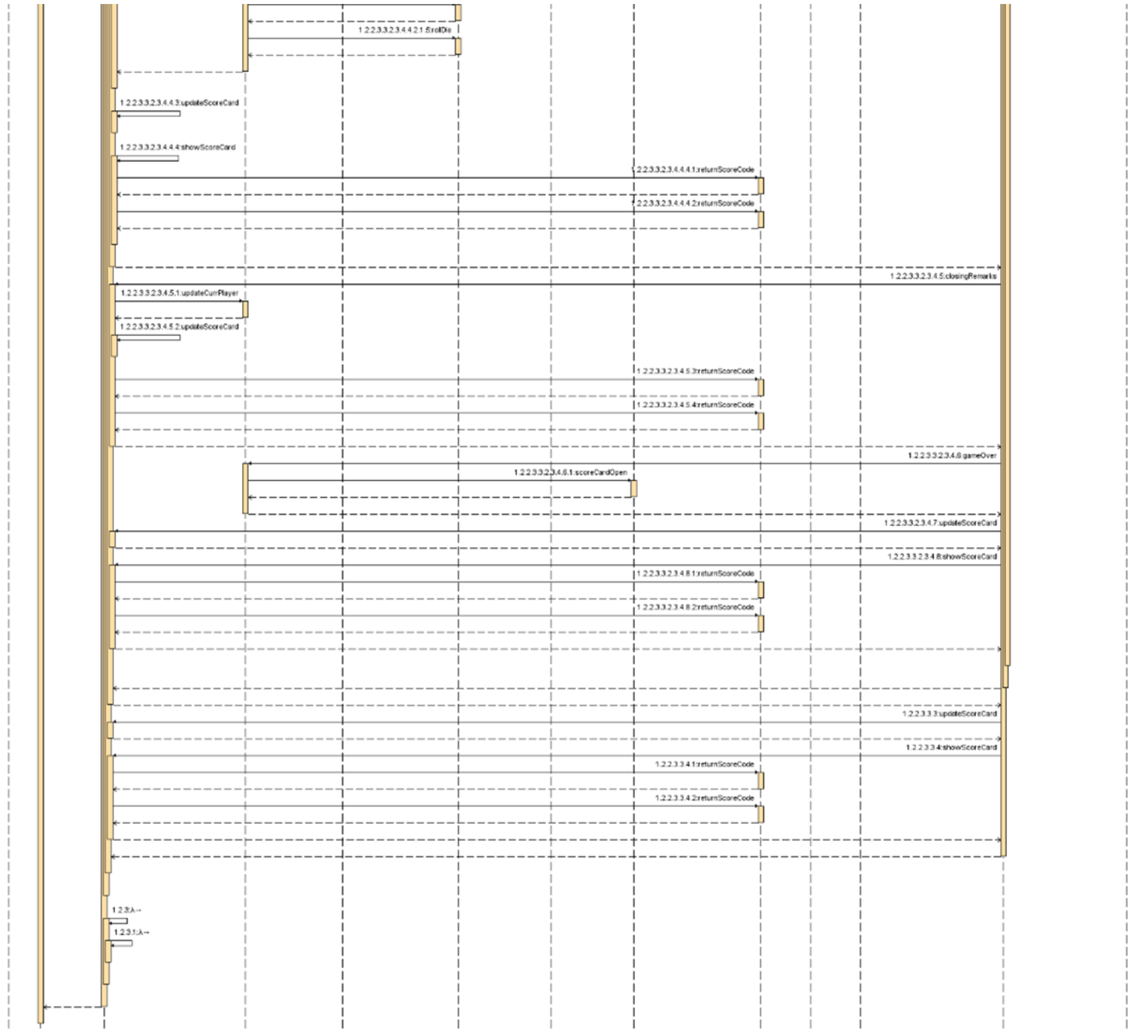

Main
+ main(string[]):void
YahtzeeGUI
-dieImages
-scoreLabels
-numSides
-numDice
-configs
+ field2: Type
+ method:Constructor
+ runYahtzee:void
+ updateScoreCard:void
+ rollNewDie:void
+ setGameConfigs:void
*
DieLabel
- die
- image
- thisDie
- value
+ getIcon:ImageIcon
+ setIcon(int):void
+getImage:ImageIcon
+ setImageSize(ImageIcon):void
Game
-gameHand
-numRolls
-toKeepGoal
+ Game(int, int, int):Constructor
+playTurn(ArrayList, int):ArrayList<>
+ scoreHand:void
+ gameOver:boolean
+ endGame:void
+returnSides:int
*
Player
- playerName
- hand
- score
-fileName
+ Player:Constructor
+ getPlayerName:String
+updateScore():void
+ getScore: ScoreCard(): Score
PlayerHand
- hand
+ playerHand:Constructor
+ sortHand:void
+ outputHand():void
+ returnNumDie():int
+ getHand():ArrayList
+ totalHand():int
PossibleScores
- upperScoreCard
- lowerScoreCard
- computeLowerScore
- scoreButtons
+ setUpperScoreCard(ArrayList, int):void
+ setLowerScoreCard(PlayerHand):void
+ createPossButtons(int, ScoreCard): ArrayList<>
+ returnArray():ArrayList<Integer>
+ fullHousefound(PlayerHand):boolean
*
PossibleScoreButton
- scoreNum
- scoreCode
- thisButton
+ possibleScorebutton(String, String):Constructor
+ returnButton:String
+ getScoreNum: scoreNum
+ getScoreCode:String
Die
- numSides
- value
+ Die(int):Constructor
+ Die:Constructor
+ rollDie:int
+ getValue:int
*
ScoreCardLabel
- scoreNum
- scoreCode
- text
- scoreLabel
+ ScoreCardLabel(String, code):Constructor
+ returnScoreCode:String
+ getText:String
+ updateScore(String):void
*
ScoreCard
- scoreLabel
- cardList
- sideCount
+ ScoreCard:Constructor
+ createScoreCard:void
+ readScoreCard:void
+ writeScoreCard:void
+updateScoreCard:void
+ returnCardList:List<ArrayList<String>>
+getUpperTotal:int
+ getLowerScore:int


Actor
Main
YahtzeeGUI
Game
Player
PlayerHand
Die
ScoreCard
ScoreCardLabel
DieLabel
PossibleScores
1:main
1.1:<<create>>
1.2:runYahtzee
1.2.1:addNorthPanel
1.2.2:λ→
1.2.2.1:<<create>>
1.2.2.1.1:<<create>>
1.2.2.1.1.1:<<create>>
1.2.2.1.1.1.1:<<create>>
1.2.2.1.1.1.1.1:rollDie
1.2.2.1.1.2:<<create>>
1.2.2.1.2:createScoreCard
1.2.2.1.2.1:<<create>>
1.2.2.1.2.2:<<create>>
1.2.2.1.2.3:<<create>>
1.2.2.1.2.4:<<create>>
1.2.2.1.2.5:<<create>>
1.2.2.1.2.6:<<create>>
1.2.2.1.2.7:<<create>>
1.2.2.1.2.8:<<create>>


1.2.2.1.2.8:<<create>>
1.2.2.2:updateScoreCard
1.2.2.3:playDiceTurn
1.2.2.3.1:createDieImages
1.2.2.3.1.1:returnNumDice
1.2.2.3.1.2:<<create>>
1.2.2.3.2:rollNewDie
1.2.2.3.2.1:playTurn
1.2.2.3.2.1.1:readScoreCard
1.2.2.3.2.1.2:clear
1.2.2.3.2.1.3:returnNumDice
1.2.2.3.2.1.4:returnNumDice
1.2.2.3.2.1.5:rollDie
1.2.2.3.2.1.5.1:rollDie
1.2.2.3.3:actionPerformed
1.2.2.3.3.1:playTurn
1.2.2.3.3.1.1:readScoreCard
1.2.2.3.3.1.2:clear
1.2.2.3.3.1.3:returnNumDice
1.2.2.3.3.1.4:returnNumDice
1.2.2.3.3.1.5:rollDie
1.2.2.3.3.1.5.1:rollDie
1.2.2.3.3.2:scoreTurn
1.2.2.3.3.2.1:printSortedDice


1.2.2.3.3.2 scoreTurn
1.2.2.3.3.2.1 printSortedDice
1.2.2.3.3.2.2 scoreHand
1.2.2.3.3.2.2.1 possScore
1.2.2.3.3.2.2.1.1 setUpperScoreCard
1.2.2.3.3.2.2.1.2 returnNumDice
1.2.2.3.3.2.2.1.3 returnNumDice
1.2.2.3.3.2.2.1.4 createPossButtons
1.2.2.3.3.2.2.1.4.1 possiblePlaces
1.2.2.3.3.2.2.1.4.2 createUpperButtons
1.2.2.3.3.2.2.1.4.3 createLowerButtons
1.2.2.3.3.2.3 actionPerformed
1.2.2.3.3.2.3.1 updateScore
1.2.2.3.3.2.3.1.1 possiblePlaces
1.2.2.3.3.2.3.1.1.1 returnCardList
1.2.2.3.3.2.3.1.1.2 getScore
1.2.2.3.3.2.3.1.2 updateScoreCard
1.2.2.3.3.2.3.1.2.1 writeScorecard
1.2.2.3.3.2.3.1.2.1.1 updateScore
1.2.2.3.3.2.3.2 updateScoreCard
1.2.2.3.3.2.3.3 returnButtonString
1.2.2.3.3.2.3.4 actionPerformed


1.2.2.3.3.2.3.3 returnButtonString
1.2.2.3.3.2.3.4 actionPerformed
1.2.2.3.3.2.3.4.1 gameOver
1.2.2.3.3.2.3.4.1.1 scoreCardOpen
1.2.2.3.3.2.3.4.2 updateCurrPlayer
1.2.2.3.3.2.3.4.3 updateScoreCard
1.2.2.3.3.2.3.4.4 playDiceTurn
1.2.2.3.3.2.3.4.4.1 createDieImages
1.2.2.3.3.2.3.4.4.1.1 returnNumDice
1.2.2.3.3.2.3.4.4.1.2 «create»
1.2.2.3.3.2.3.4.4.2 rollNewDie
1.2.2.3.3.2.3.4.4.2.1 playTurn
1.2.2.3.3.2.3.4.4.2.1.1 readScoreCard
1.2.2.3.3.2.3.4.4.2.1.2 clear
1.2.2.3.3.2.3.4.4.2.1.3 returnNumDice
1.2.2.3.3.2.3.4.4.2.1.4 returnNumDice
1.2.2.3.3.2.3.4.4.2.1.5 rollDie


1.2.2.3.3.2.3.4.4.2.1.5:rollDie
1.2.2.3.3.2.3.4.4.3:updateScoreCard
1.2.2.3.3.2.3.4.4.4:showScoreCard
1.2.2.3.3.2.3.4.4.4.1:returnScoreCode
1.2.2.3.3.2.3.4.4.4.2:returnScoreCode
1.2.2.3.3.2.3.4.5:closingRemarks
1.2.2.3.3.2.3.4.5.1:updateCurrPlayer
1.2.2.3.3.2.3.4.5.2:updateScoreCard
1.2.2.3.3.2.3.4.5.3:returnScoreCode
1.2.2.3.3.2.3.4.5.4:returnScoreCode
1.2.2.3.3.2.3.4.6:gameOver
1.2.2.3.3.2.3.4.6.1:scoreCardOpen
1.2.2.3.3.2.3.4.7:updateScoreCard
1.2.2.3.3.2.3.4.8:showScoreCard
1.2.2.3.3.2.3.4.8.1:returnScoreCode
1.2.2.3.3.2.3.4.8.2:returnScoreCode
1.2.2.3.3.3:updateScoreCard
1.2.2.3.3.4:showScoreCard
1.2.2.3.3.4.1:returnScoreCode
1.2.2.3.3.4.2:returnScoreCode
1.2.3:λ→
1.2.3.1:λ→

V. Test Plan

Unit Testing

To perform unit testing, we will be testing the reading from the configuration file. We will also be testing the correctness of the scores. Connor will write these unit tests and all group members will review these tests before they are considered “complete”.

Integration Testing

To perform Integration Testing, our group will be performing tests on things like generating the upper scorecard based on the user’s configuration. This integration test will test multiple components: making sure that the upper scorecard has generated the right amount of die scores, and validating that the scores for each side of the die are correctly computed. Our overall integration tests will cover testing the upper scorecard, lower scorecard, creating a die hand, and more.

System Testing

To perform System Testing, our group will come up with a list of use cases that could possibly happen for a user while they're playing Yahtzee on the GUI. Essentially, our developers will test the program of GUI Yahtzee but from the user-end perspective, searching for potential bugs and errors. Our team will then run through each of the possible scenarios and make sure that those scenarios are functional and follow the project requirements/guidelines. All exposed bugs will be reported to the developing team and fixed as soon as possible.

Functional Testing:

To perform functional testing, we will be testing the GUI to see if it works the way we intend it to. We will do this by having all members of the group run the program and discuss if the GUI looks and functions the way we want it. The requirements of how the GUI should look like and function will be taken from the functional requirements that we have agreed upon. This test will also be finalized by our final test, which is user acceptance testing. This will allow us to see how a user outside of the group uses and interacts with the GUI.

Performance Testing:

To perform performance testing, we will be pushing the limits of the program. One test we will perform is when a user inputs a large number for each configuration. This will test how the program handles excessively large numbers of dice, turns, and sides on the dice. We will also be testing if the user enters too many or too little y's or n's when asked which dice to keep.

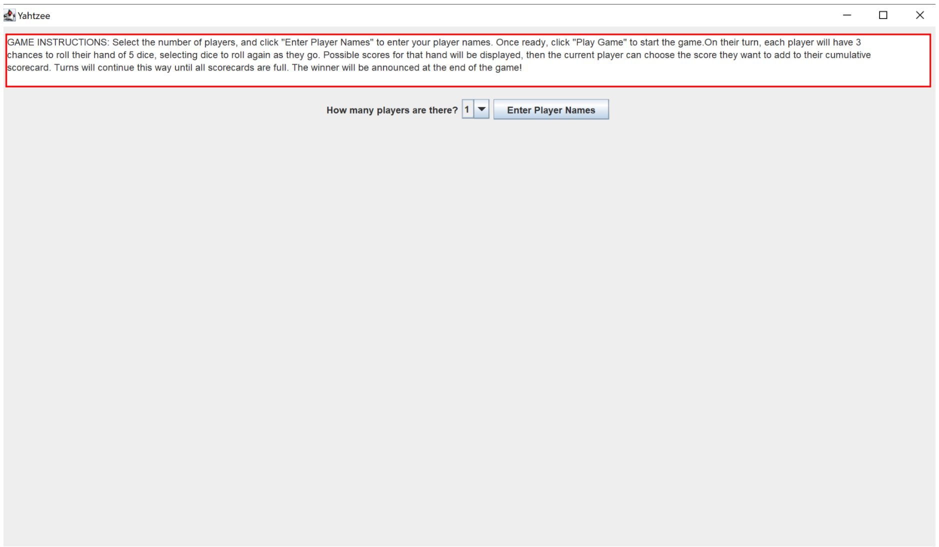
User Acceptance Testing:

To perform User Acceptance Testing, we will have randomly selected 1-3 people, who are not members of our team, play a full game of Yahtzee on the GUI that is generated from our code. The testers will run through the game checking for any display errors or any functional errors that they may come across. All exposed bugs will be reported to the developing team and fixed as soon as possible.

VI. Project Implementation Description

Our group implemented everything that was dealt with the program and its requirements, including things like creating a die hand, rolling/rerolling die, choosing from a score line, entering names, deciding how many players there will be, etc. Screenshots of our functioning GUI are shown below to demonstrate what our game of Yahtzee would look like and the functionality of the game was implemented.

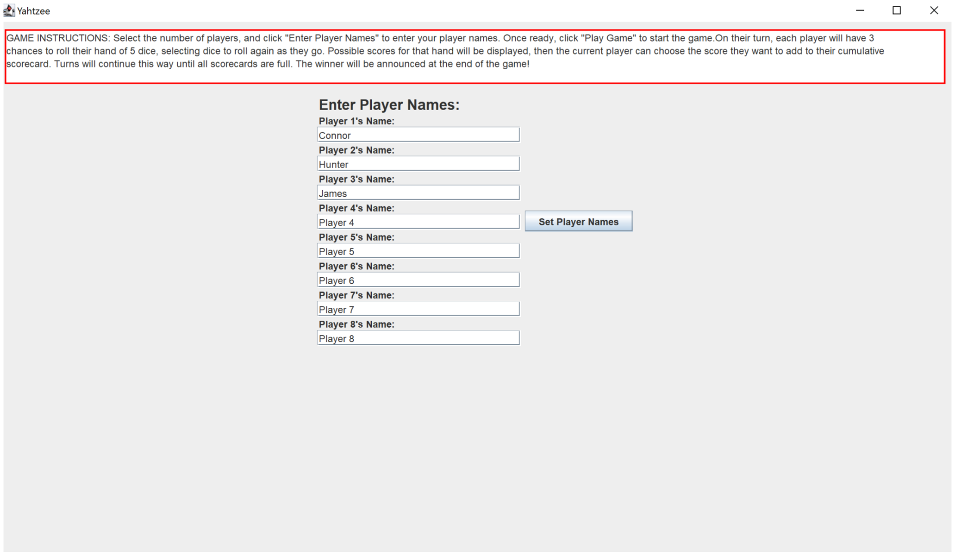
This is the successful implementation of Yahtzee. The interface first allows the users to choose how many players are going to be playing in the game.



GAME INSTRUCTIONS: Select the number of players, and click "Enter Player Names" to enter your player names. Once ready, click "Play Game" to start the game.On their turn, each player will have 3 chances to roll their hand of 5 dice, selecting dice to roll again as they go. Possible scores for that hand will be displayed, then the current player can choose the score they want to add to their cumulative scorecard. Turns will continue this way until all scorecards are full. The winner will be announced at the end of the game!

How many players are there? 1

The players are then able to change their names to whatever they please or leave them the same as the defaults.



GAME INSTRUCTIONS: Select the number of players, and click "Enter Player Names" to enter your player names. Once ready, click "Play Game" to start the game.On their turn, each player will have 3 chances to roll their hand of 5 dice, selecting dice to roll again as they go. Possible scores for that hand will be displayed, then the current player can choose the score they want to add to their cumulative scorecard. Turns will continue this way until all scorecards are full. The winner will be announced at the end of the game!

Enter Player Names:

Player 1's Name: Connor

Player 2's Name: Hunter

Player 3's Name: James

Player 4's Name: Player 4

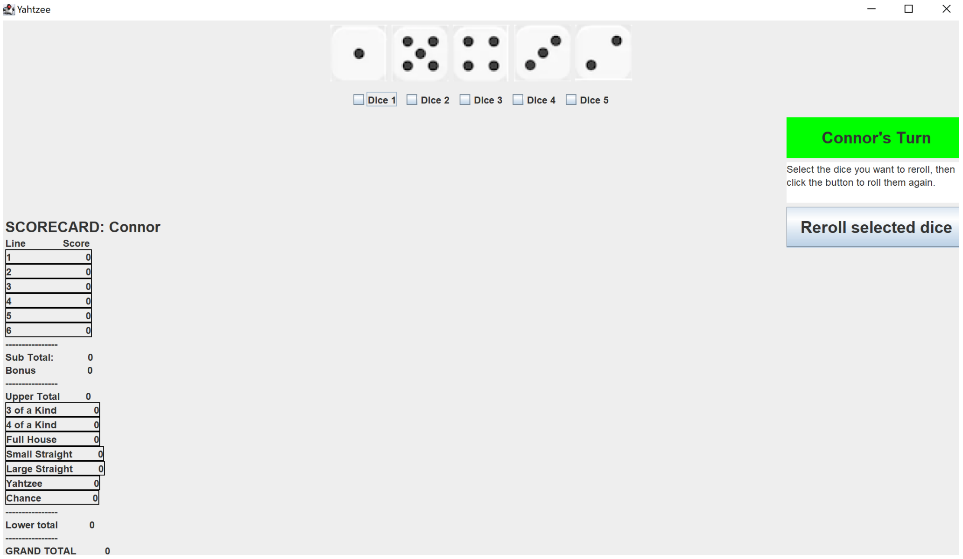
Player 5's Name: Player 5

Player 6's Name: Player 6

Player 7's Name: Player 7

Player 8's Name: Player 8

Players are given a set of die that have already been rolled. The players can choose which dice they would like to reroll by clicking on that respective die number.



Select the dice you want to reroll, then click the button to roll them again.

SCORECARD: Connor

Line	Score
1	0
2	0
3	0
4	0
5	0
6	0

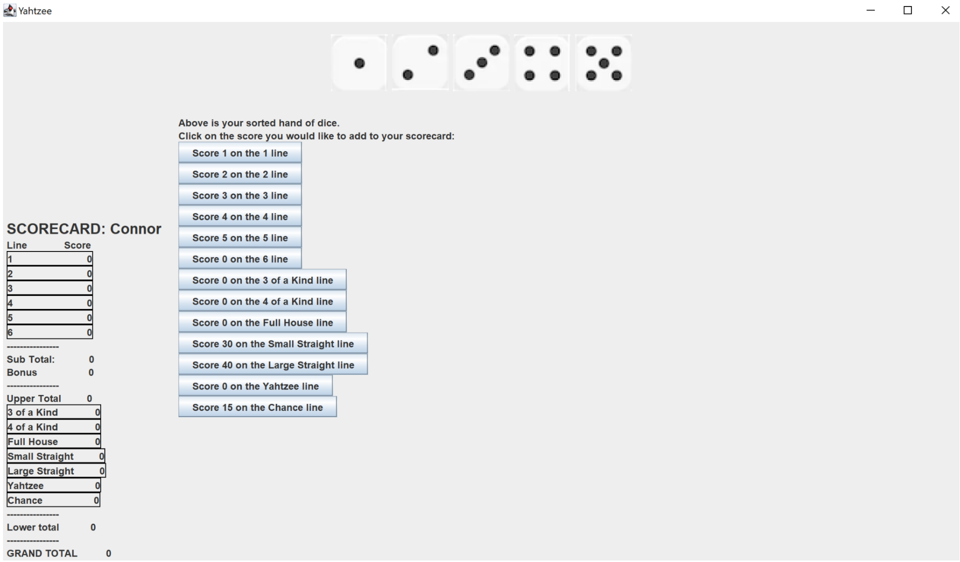
Sub Total:	0
Bonus	0

Upper Total	0
3 of a Kind	0
4 of a Kind	0
Full House	0
Small Straight	0
Large Straight	0
Yahtzee	0
Chance	0

Lower total	0

GRAND TOTAL	0

After all three rolls have been used, the player can choose which score line to use by clicking on one of the score line buttons.


Yahtzee
Score 1 on the 1 line
Score 2 on the 2 line
Score 3 on the 3 line
Score 4 on the 4 line
Score 5 on the 5 line
Score 0 on the 6 line
Score 0 on the 3 of a Kind line
Score 0 on the 4 of a Kind line
Score 0 on the Full House line
Score 30 on the Small Straight line
Score 40 on the Large Straight line
Score 0 on the Yahtzee line
Score 15 on the Chance line

Above is your sorted hand of dice.
Click on the score you would like to add to your scorecard:

SCORECARD: Connor

Line	Score
1	0
2	0
3	0
4	0
5	0
6	0

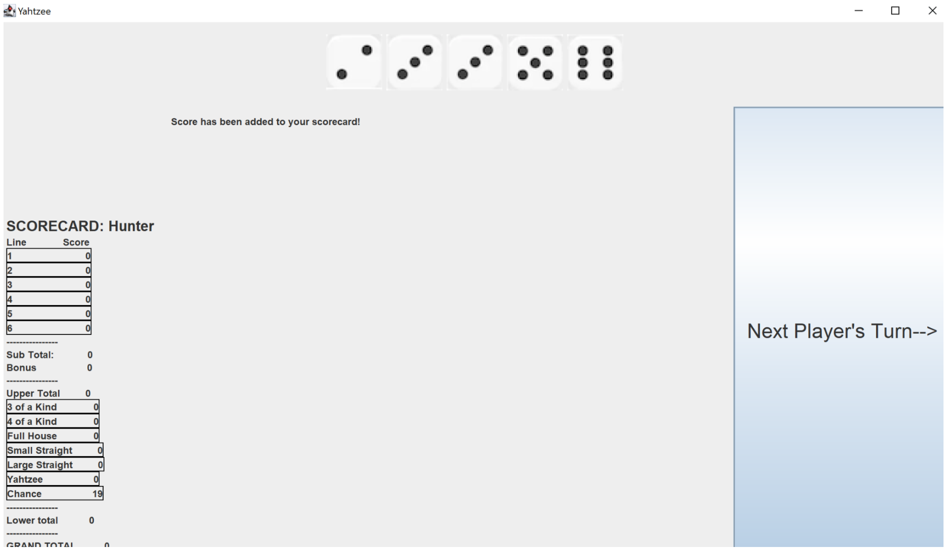
Sub Total:	0
Bonus	0

Upper Total	0
3 of a Kind	0
4 of a Kind	0
Full House	0
Small Straight	0
Large Straight	0
Yahtzee	0
Chance	0

Lower total	0

GRAND TOTAL	0

Once a score line has been selected, the player’s updated scorecard is displayed. The game can be continued by clicking on the “Next Player’s Turn” button on the right.



Score has been added to your scorecard!

SCORECARD: Hunter

Line	Score
1	0
2	0
3	0
4	0
5	0
6	0

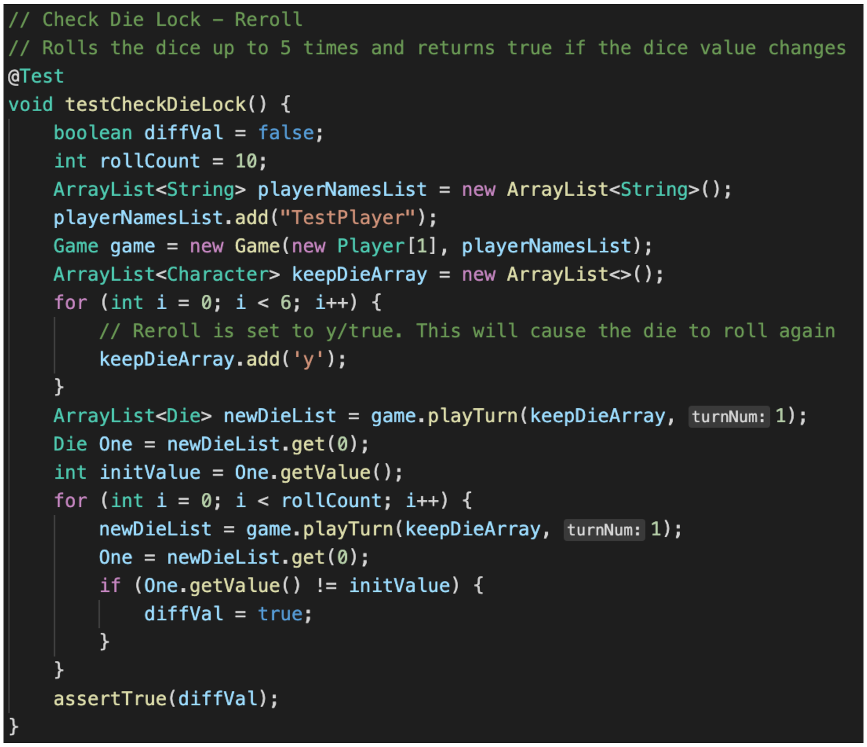
Sub Total:	0
Bonus	0

Upper Total	0
3 of a Kind	0
4 of a Kind	0
Full House	0
Small Straight	0
Large Straight	0
Yahtzee	0
Chance	19

Lower total	0

GRAND TOTAL	0

The below test checks to make sure that the dice rolls and changes its initial value. This value is stored into a temporary variable, and compared with the first die after it is rolled. The dice will be rolled 10 times, and if the value was never changed, the test will fail. This test is structured the same way the GUI flows to do a reroll.



```java
// Check Die Lock - Reroll
// Rolls the dice up to 5 times and returns true if the dice value changes
@Test
void testCheckDieLock() {
    boolean diffVal = false;
    int rollCount = 10;
    ArrayList<String> playerNamesList = new ArrayList<String>();
    playerNamesList.add("TestPlayer");
    Game game = new Game(new Player[1], playerNamesList);
    ArrayList<Character> keepDieArray = new ArrayList<>();
    for (int i = 0; i < 6; i++) {
        // Reroll is set to y/true. This will cause the die to roll again
        keepDieArray.add('y');
    }
    ArrayList<Die> newDieList = game.playTurn(keepDieArray, turnNum: 1);
    Die One = newDieList.get(0);
    int initValue = One.getValue();
    for (int i = 0; i < rollCount; i++) {
        newDieList = game.playTurn(keepDieArray, turnNum: 1);
        One = newDieList.get(0);
        if (One.getValue() != initValue) {
            diffVal = true;
        }
    }
    assertTrue(diffVal);
}
```

Github Repository:

https://github.com/GU-2021-Fall-CPSC224/final-yahtzee-cup-o-java

VII. Future Work

For future development of this project, one of the first things we would implement would be the Lizard Spock rules from HW2. We would make it so that the player has the option to choose, how many sides are on a die, how many dice they would like to play with, and how many rolls per turn each player would get. This would mean adding features to the configuration page at the very beginning of the game. In addition to being able to change player names, the GUI would have dropdown menus of choosing the number of dice to play with, and so on. In addition to the HW2 requirements, we would further this project by displaying a small leaderboard that displays every player's current score for the entire duration of the game. This would mean having to adjust the GUI layout during player turns and having to make room for the leaderboard. By adding a feature like this, players are able to think more strategically as they can see their opponents score, as well as increase the competitive drive in the game. Lastly, our group could make the game more user-friendly/fun by adding an animation that shows the dice being re-rolled, or a big celebratory pop-up that is prompted when a user rolls a Yahtzee! This makes the game more fun and exciting by rewarding players for their excellent rolls.

VIII. Glossary

GUI - Graphical User Interface

Java - Programming language which the project will be developed in

JUnit - The testing framework which will be used to run the tests

Java Swing - The interface used to develop the graphical user interface

UML - Unified Modeling Language

IX. References

Wiki. “Yahtzee.” Wikipedia. Wikimedia Foundation, April 21, 2022. https://en.wikipedia.org/wiki/Yahtzee#:~:text=Yahtzee%20is%20a%20dice%20game,Edwin%20S.%20Lowe%20in%201956.

Sefan, Petr, Vojtech Horky, Lubomir Bulej, and Petr Tuma. “Unit Testing Performance in Java Projects: Proceedings of the 8th ACM/Spec on International Conference on Performance Engineering.” ACM Conferences, April 1, 2017. https://dl.acm.org/doi/abs/10.1145/3030207.3030226.

BoardGames, Ultra. “The History of Yahtzee.” UltraBoardGames. Ultra BoardGames, 2020. https://www.ultraboardgames.com/yahtzee/history.php.

Crandall, Aaron. “Unit Testing II, Slide 8.” Blackboard, January 2022. https://learn.gonzaga.edu/ultra/courses/_966960_1/cl/outline.

Indeed Editorial Team. “What Is System Testing? (plus How to Perform a System Test).” Indeed Career Guide, July 22, 2021. https://www.indeed.com/career-advice/career-development/system-testing.

“Unit Testing Swing Applications with Marathon/Javadriver.” Test Automation Tools for Java and Web. Accessed April 5, 2022. https://marathontesting.com/blog-posts/unit-testing-swing-apps/.