That One Team

Adam Kowalchyk, Daniel Hoberman, Maya Fleming, Trevor Diuco

Project Overview

- Multiplayer Yahtzee with a Graphical User Interface
  - We implemented the capability to play a full game of traditional Yahtzee with 1 or more players
- Features
  - Full Yahtzee gameplay capable of supporting 1-6 players
  - Full automated dice rolling and score tracking with one click of a button
  - Interactive GUI designed using the Java Swing Toolkit
  - Ability to name each player in the game
- Limitations
  - Only supports traditional Yahtzee rules with 5 dice and 3 rolls per turn

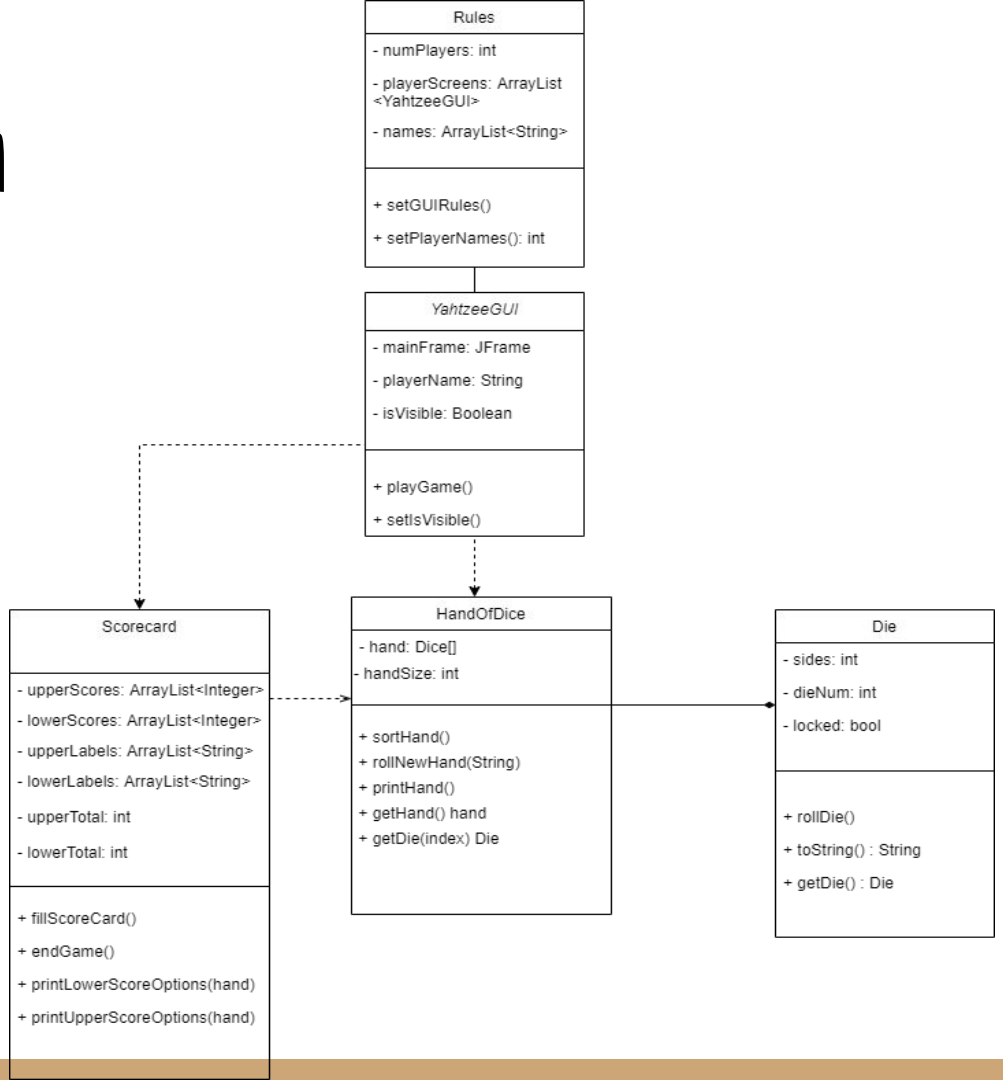
Project Requirements

- Functional Requirements
  - User should be able to select number of players
  - Display current dice on screen
  - Display possible scores
- Non-Functional
  - Give each player selected a name
  - Sports scores of players and displays them in order

Project Solution Approach

- Creation of the YahtzeeGUI class
  - For each player in the game, YahtzeeGUI is instantiated which gives the player their own screen when it is their turn.
  - This was crucial for the implementation of multiplayer functionality
- Addition of setPlayerNames method in Rules class
  - Runs the code to prompt the user/s to name their player at the beginning of the game
  - Result is passed to YahtzeeGUI to be used further on in the program
- Make visible player's scorecard on their YahtzeeGUI
  - We felt that by adding this feature, it greatly improves the UI design of the program
- Display of final game results
  - Program announces the winner of the game and their respective score

UML Design


Rules
- numPlayers: int
- playerScreens: ArrayList <YahtzeeGUI>
- names: ArrayList<String>
+ setGUIRules()
+ setPlayerNames(): int
YahtzeeGUI
- mainFrame: JFrame
- playerName: String
- isVisible: Boolean
+ playGame()
+ setIsVisible()
Scorecard
- upperScores: ArrayList<Integer>
- lowerScores: ArrayList<Integer>
- upperLabels: ArrayList<String>
- lowerLabels: ArrayList<String>
- upperTotal: int
- lowerTotal: int
+ fillScoreCard()
+ endGame()
+ printLowerScoreOptions(hand)
+ printUpperScoreOptions(hand)
HandOfDice
- hand: Dice[]
- handSize: int
+ sortHand()
+ rollNewHand(String)
+ printHand()
+ getHand() hand
+ getDie(index) Die
Die
- sides: int
- dieNum: int
- locked: bool
+ rollDie()
+ toString() : String
+ getDie() : Die

Team Collaboration Approaches

- As a group, communicated through GitHub and text messaging
  - Communicated about issues we had, and on meetings time and deadlines
- Used GitHub issues board to post what was needed to work on by the end of the project
  - Was able to refer to the issues board as a reminder of what was needed by the end
- Created branches on GitHub after new changes/updates made to code
- Working with a group...
  - Learned about group communication and how important it is to be able to communicate about your progress with your group
- Some issues we coded in group sessions, and in some issues we coded individually

Testing, Validation, and Acceptance Plan

- User testing was the main source of testing
- Testing code little by little, seeing if the output is what is desired
  - Test certain functions/classes to determine if it will further the project, if there's an error, or if it does not add to the project
- Allowed others to test the project
  - Receive feedback from another POV
  - Get more input on what could be changed/what is favored in the project
- Is the project deliverable?
  - The project meets all requirements
  - It can handle several amount of players during the game
  - At the end, will sort player's scores, and output the winner of the game

Live Demo Time

Experience

- It is difficult to coordinate work for a group
  - An experience we had to learn
- The project outcome from a group is different
  - Every member had their own ideas to add
- Git Experience