Yahtzee
Group Project

CIS-17B
Danielle F, Logan O,
Ismael P, Amir J
Date: 04/24/23

Table of Context

Introduction

Title: Yahtzee

Choose a starting player by any method (oldest player, youngest player, highest roll of the dice, etc.)

Beginning with the starting player, players will take turns one at a time in clockwise order. The game consists of thirteen rounds, and at the end of the thirteenth round, the game will end. (All the categories on the players' scorecards will be completely filled in at that point.)The dice can be scored after any of the rolls, but scoring the dice ends the player's turn. Setting dice aside after one roll does not prevent one or more of them from being rolled again on any subsequent roll if the player so chooses.

Each player's goal is to try and score as high as possible in one of the thirteen categories shown on their scorecard.

To score the dice, the player selects one of the categories on their scorecard and writes the score into it. Each category can be scored only once per game (except for the Yahtzee category). Categories can be filled in any order. A player must score the dice on their turn even if it turns out that there are no good categories remaining to score in. Once a category is filled it may not be changed.

A player may write a score of zero in any category if they have rolled no point-generating results or simply choose to do so. After marking their score on their scorecard, the player's turn ends, and play proceeds to the player on their left.

Summary

Project Size: ~2485 lines

The entire User and Admin class was written by Danielle with around 28 versions of the code. In the final version of the code, the User class allows you to sign in and saves your high score to a binary file. While the Admin class gives the admin the ability to rewrite a specific user's high score, allows the admin to see all profiles created, can pick a specific profile to look at, etc. Her program also allows the player to play as a guest if they didn't want to sign in. The Yahtzee portion of the project was written by Ismael with some help from Danielle to fix some bugs and clean up the code to make it a little more readable. The Yahtzee program consists of two helper classes, the scorecard class, and the dice class. When first starting the game it rolls 5 random dice and allows the user to pick a category from the scorecard if they wish. If not, it allows the user to keep or reroll specific dice in order to get the desired outcome for the scorecard.

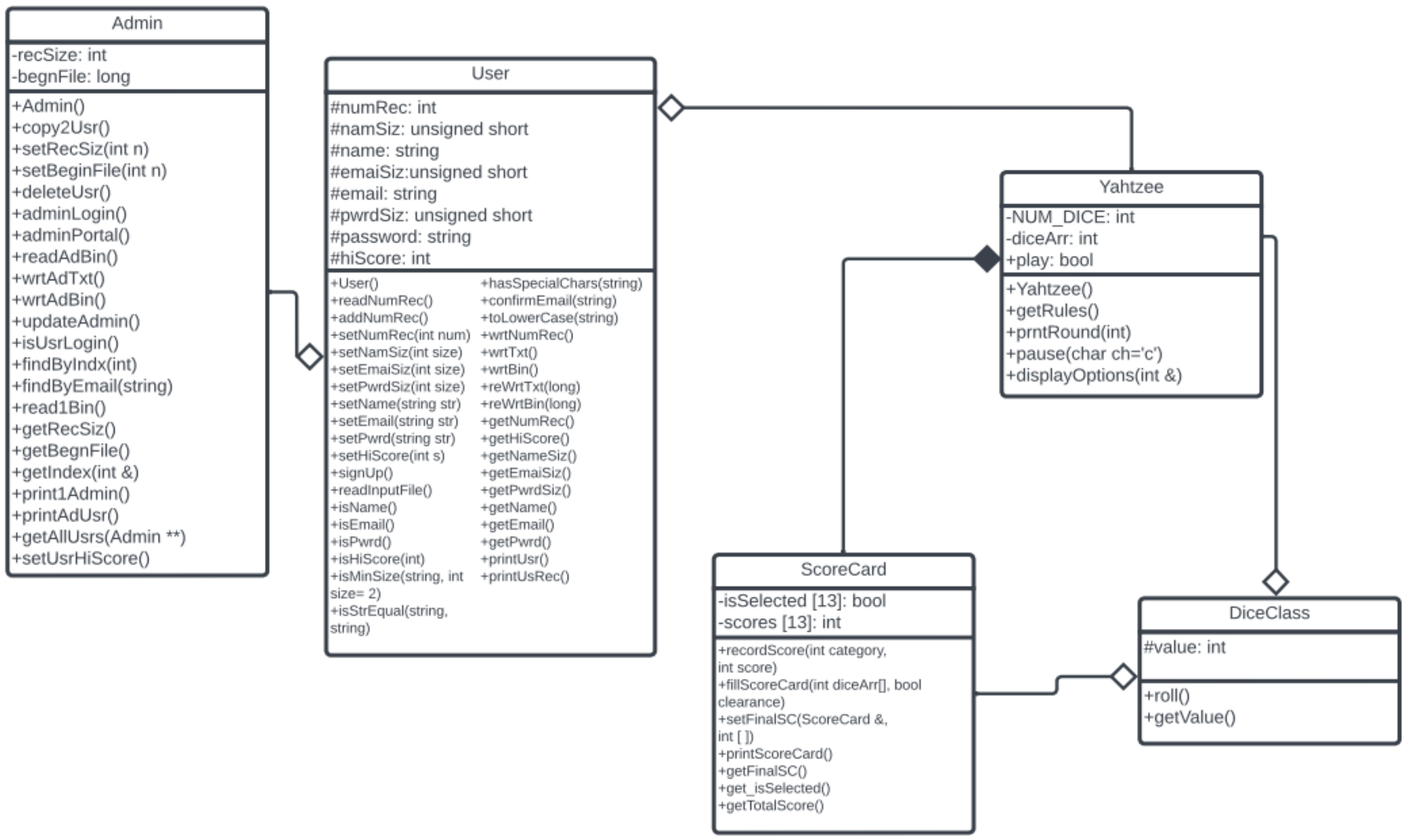
Description

This is a Yahtzee Program, it allows you to play Yahtzee by yourself or play against a friend. Whoever gets the higher score wins, and if the person logged in gets a new personal high score, that score will be saved to a binary file.

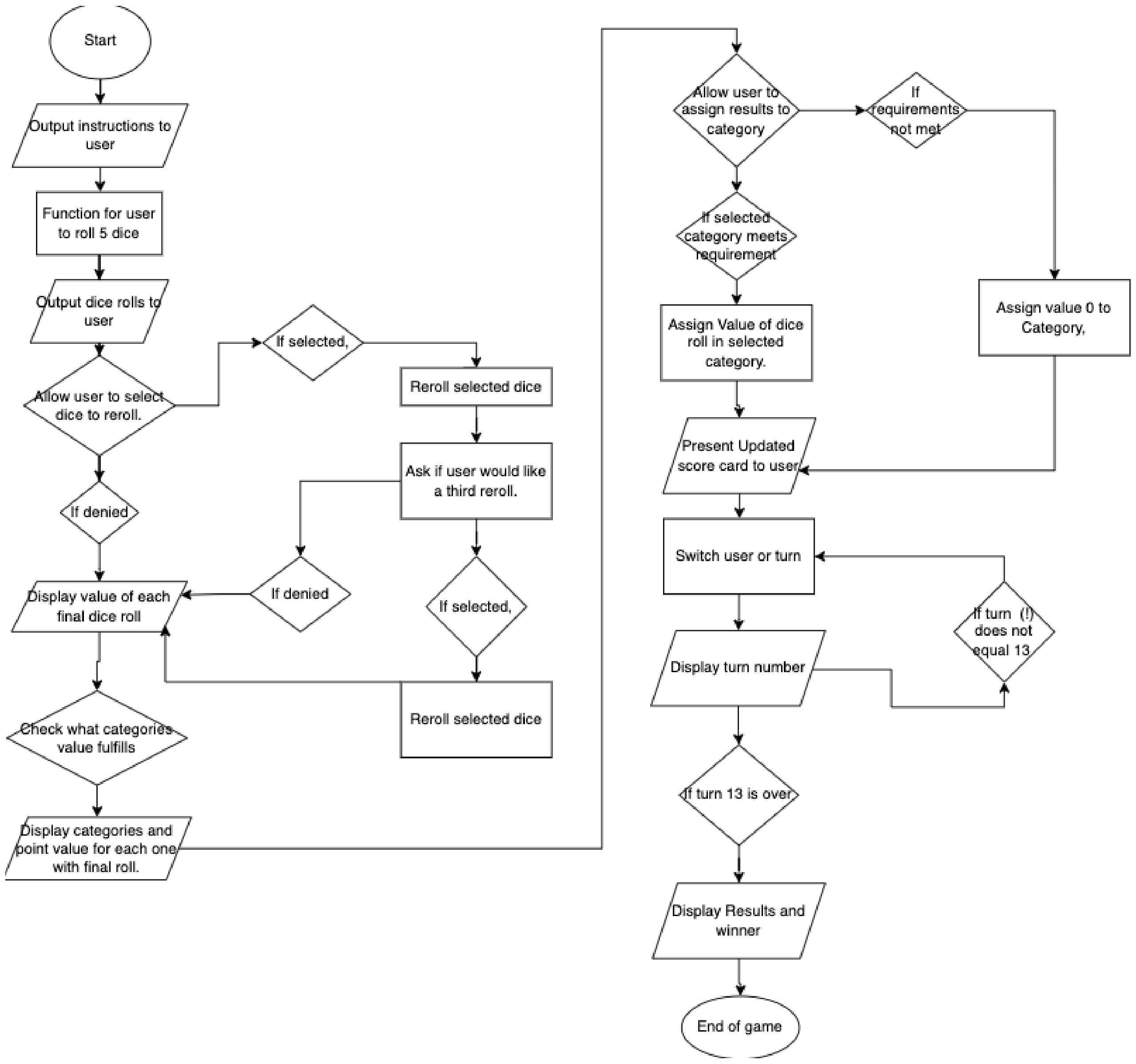
GitHub Links

- Danielle's Github: https://github.com/koa2019/yahtzee
- Ismael's Github: https://github.com/Error1417/Yahtzee_Project

UML Chart



FlowChart


Start
Output instructions to user
Function for user to roll 5 dice
Output dice rolls to user
Allow user to select dice to reroll.
If selected,
Reroll selected dice
Ask if user would like a third reroll.
If denied
If denied
If selected,
Reroll selected dice
Display value of each final dice roll
Check what categories value fulfills
Display categories and point value for each one with final roll.
Allow user to assign results to category
If requirements not met
If selected category meets requirement
Assign Value of dice roll in selected category.
Assign value 0 to Category,
Present Updated score card to user
Switch user or turn
Display turn number
If turn (!) does not equal 13
If turn 13 is over
Display Results and winner
End of game

Gantt Chart

Yahtzee_4

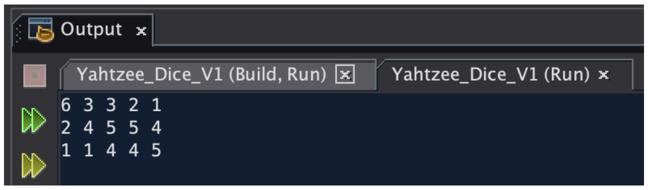
Incomplete	% Complete
⬜	🟩

Name	Task	3/19 - 3/25	3-25 - 3/31	4/1 - 4/8	4/9 - 4/15	4/16-4/24
Danielle	User class	🟩	100 %	🟩		
Danielle	Admin class	🟩	🟩	100%	🟩	
Danielle	Game flow - round & player control	🟩	100 %	🟩		
Danielle	Gantt Chart	🟩	🟩	90%	🟩	🟩
Danielle/Ismael	Weekly Group Write up	🟩	🟩	100%	🟩	🟩
Ismael	UML Chart	🟩	🟩	100%	🟩	🟩
Ish	Scorecard class. 1 round working for 1 player	🟩	100%			
Ish/Logan	Dice class	🟩	100%			
Logan	Flowchart	🟩	100%			
Ish	Style scorecard print out	🟩	100%			
Danielle	Merge game flow(round+player control) to main.cpp		🟩	100%		
Danielle/Ish	Merge User & Yahtzee classes			100%	🟩	
Logan/Danielle	Compose to-do list for documentation			🟩	100%	
Logan/Everyone	Start documentation				50%	🟩
Everyone	Plan presentation				0%	🟩
Danielle/Ismael	Finalize documentation	🟩	🟩	95 %	🟩	🟩
Everyone	Practice presentation					100%
Logan	Style html page with game's name & scorecard			100%	🟩	
Danielle	DRY. Clean up repeative code in Admin & User				100%	🟩
Ish	DRY. Clean up repeative code in Yahtzee & ScoreCard				🟩	100%
Ish	Check the game for runtime errors				🟩	100%

Yahtzee Class Summaries

Yahtzee_Dice_V1

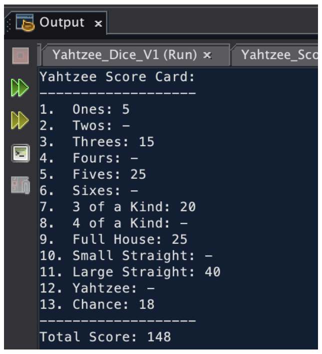
-In this version of the Yahtzee game, I create a Dice class that creates 5 random dice and shows them to the user. I output these 5 random dice three times to test their randomness.



```
6 3 3 2 1
2 4 5 5 4
1 1 4 4 5
```

Yahtzee_ScoreCard_V2

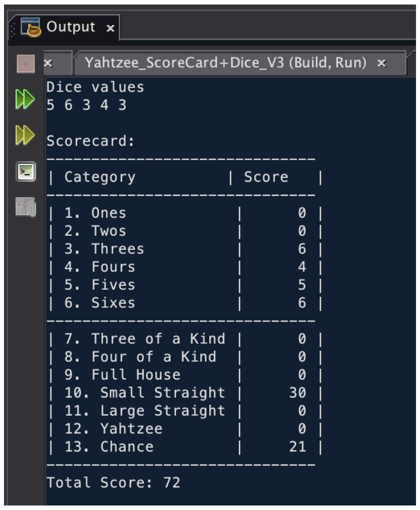
-The next part of making this game for me was creating the scorecard class on which the values of the dice would be displayed. This class created a scorecard and for the moment only outputs random numbers into the places of the categories.



```
Yahtzee Score Card:
-------------------
1.  Ones: 5
2.  Twos: -
3.  Threes: 15
4.  Fours: -
5.  Fives: 25
6.  Sixes: -
7.  3 of a Kind: 20
8.  4 of a Kind: -
9.  Full House: 25
10. Small Straight: -
11. Large Straight: 40
12. Yahtzee: -
13. Chance: 18
-------------------
Total Score: 148
```

Yahtzee_ScoreCard+Dice_V3

-In this version of the game, I combined the ScoreCard and Dice class in order to create an output to the scorecard that wasn't just random. The scorecard takes in the random values of the dice and tests them to see if they fulfill the requirements for any of the categories.



```
Dice values
5 6 3 4 3

Scorecard:
------------------------------
| Category           | Score  |
------------------------------
| 1. Ones             |      0 |
| 2. Twos             |      0 |
| 3. Threes           |      6 |
| 4. Fours            |      4 |
| 5. Fives            |      5 |
| 6. Sixes            |      6 |
------------------------------
| 7. Three of a Kind |      0 |
| 8. Four of a Kind  |      0 |
| 9. Full House       |      0 |
| 10. Small Straight |     30 |
| 11. Large Straight |      0 |
| 12. Yahtzee         |      0 |
| 13. Chance          |     21 |
------------------------------
Total Score: 72
```

Yahtzee_Game_V4

-This version of the game was when I added the Yahtzee class, and it started looking more like a playable game. I added the ability to pick which dice you wanted to keep and which dice you wanted to reroll. I also added the ability to pick a category, but that feature wasn't working yet. There are some bugs in this version of the code such as not being able to enter letters when the input asks for a number. This would cause an infinite loop. Also, this version allowed the user to break the game by having more than 5 dice. I can't show the output of this one or any other version from here on out because it would be too large.

Yahtzee_Game_V4.1

-This version of the game moves around some of the outputs of the program in hopes to make it like the real game. The only problem is that I was outputting the options for the game before actually rolling the dice for the user, so the user was allowed to pick a category before the dice were rolled. This version also added input verification.

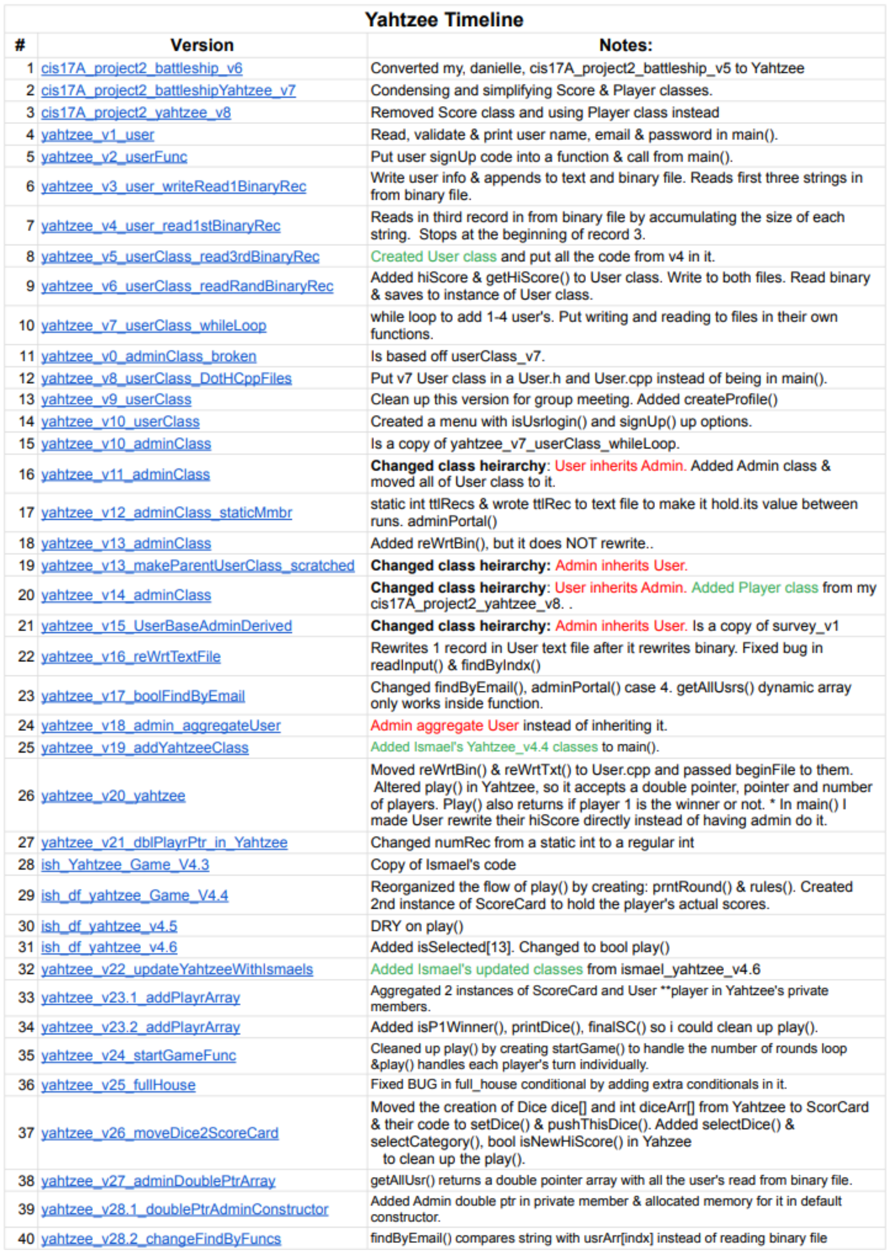
Yahtzee_Game_V4.2

-This version was more focused on fixing the selection system for the category, and fixing a bug I was having with the parameters of the fillscorecard function. I decided to move the category selection system into this function because it fixed the problem I was having with the parameter problems. In order to fix it, I did need to pass in a bool that said if the player was allowed to pick a category or not.

Yahtzee_Game_V4.3

-This last version of my code was mainly for me to test out what would work when trying to save the category a user had chosen. My attempts were unsuccessful, but thankfully my partner Danielle was able to fix the problems with the category selection system and was able to make the code look and run much cleaner.

Yahtzee Version Timeline Table



Yahtzee Timeline

#	Version	Notes:
1	cis17A_project2_battleship_v6	Converted my, danielle, cis17A_project2_battleship_v5 to Yahtzee
2	cis17A_project2_battleshipYahtzee_v7	Condensing and simplifying Score & Player classes.
3	cis17A_project2_yahtzee_v8	Removed Score class and using Player class instead
4	yahtzee_v1_user	Read, validate & print user name, email & password in main().
5	yahtzee_v2_userFunc	Put user signUp code into a function & call from main().
6	yahtzee_v3_user_writeRead1BinaryRec	Write user info & appends to text and binary file. Reads first three strings in from binary file.
7	yahtzee_v4_user_read1stBinaryRec	Reads in third record in from binary file by accumulating the size of each string. Stops at the beginning of record 3.
8	yahtzee_v5_userClass_read3rdBinaryRec	Created User class and put all the code from v4 in it.
9	yahtzee_v6_userClass_readRandBinaryRec	Added hiScore & getHiScore() to User class. Write to both files. Read binary & saves to instance of User class.
10	yahtzee_v7_userClass_whileLoop	while loop to add 1-4 user's. Put writing and reading to files in their own functions.
11	yahtzee_v0_adminClass_broken	Is based off userClass_v7.
12	yahtzee_v8_userClass_DotHCppFiles	Put v7 User class in a User.h and User.cpp instead of being in main().
13	yahtzee_v9_userClass	Clean up this version for group meeting. Added createProfile()
14	yahtzee_v10_userClass	Created a menu with isUsrlogin() and signUp() up options.
15	yahtzee_v10_adminClass	Is a copy of yahtzee_v7_userClass_whileLoop.
16	yahtzee_v11_adminClass	**Changed class heirarchy**: User inherits Admin. Added Admin class & moved all of User class to it.
17	yahtzee_v12_adminClass_staticMmbr	static int ttlRecs & wrote ttlRec to text file to make it hold.its value between runs. adminPortal()
18	yahtzee_v13_adminClass	Added reWrtBin(), but it does NOT rewrite..
19	yahtzee_v13_makeParentUserClass_scratched	**Changed class heirarchy:** Admin inherits User.
20	yahtzee_v14_adminClass	**Changed class heirarchy**: User inherits Admin. Added Player class from my cis17A_project2_yahtzee_v8. .
21	yahtzee_v15_UserBaseAdminDerived	**Changed class heirarchy:** Admin inherits User. Is a copy of survey_v1
22	yahtzee_v16_reWrtTextFile	Rewrites 1 record in User text file after it rewrites binary. Fixed bug in readInput() & findByIndx()
23	yahtzee_v17_boolFindByEmail	Changed findByEmail(), adminPortal() case 4. getAllUsrs() dynamic array only works inside function.
24	yahtzee_v18_admin_aggregateUser	Admin aggregate User instead of inheriting it.
25	yahtzee_v19_addYahtzeeClass	Added Ismael's Yahtzee_v4.4 classes to main().
26	yahtzee_v20_yahtzee	Moved reWrtBin() & reWrtTxt() to User.cpp and passed beginFile to them. Altered play() in Yahtzee, so it accepts a double pointer, pointer and number of players. Play() also returns if player 1 is the winner or not. * In main() I made User rewrite their hiScore directly instead of having admin do it.
27	yahtzee_v21_dblPlayrPtr_in_Yahtzee	Changed numRec from a static int to a regular int
28	ish_Yahtzee_Game_V4.3	Copy of Ismael's code
29	ish_df_yahtzee_Game_V4.4	Reorganized the flow of play() by creating: prntRound() & rules(). Created 2nd instance of ScoreCard to hold the player's actual scores.
30	ish_df_yahtzee_v4.5	DRY on play()
31	ish_df_yahtzee_v4.6	Added isSelected[13]. Changed to bool play()
32	yahtzee_v22_updateYahtzeeWithIsmaels	Added Ismael's updated classes from ismael_yahtzee_v4.6
33	yahtzee_v23.1_addPlayrArray	Aggregated 2 instances of ScoreCard and User **player in Yahtzee's private members.
34	yahtzee_v23.2_addPlayrArray	Added isP1Winner(), printDice(), finalSC() so i could clean up play().
35	yahtzee_v24_startGameFunc	Cleaned up play() by creating startGame() to handle the number of rounds loop &play() handles each player's turn individually.
36	yahtzee_v25_fullHouse	Fixed BUG in full_house conditional by adding extra conditionals in it.
37	yahtzee_v26_moveDice2ScoreCard	Moved the creation of Dice dice[] and int diceArr[] from Yahtzee to ScorCard & their code to setDice() & pushThisDice(). Added selectDice() & selectCategory(), bool isNewHiScore() in Yahtzee to clean up the play().
38	yahtzee_v27_adminDoublePtrArray	getAllUsr() returns a double pointer array with all the user's read from binary file.
39	yahtzee_v28.1_doublePtrAdminConstructor	Added Admin double ptr in private member & allocated memory for it in default constructor.
40	yahtzee_v28.2_changeFindByFuncs	findByEmail() compares string with usrArr[indx] instead of reading binary file

Summary of Pivotal User/Admin Versions

yahtzee_v5_userClass_read3rdBinaryRec:
I created a User class. The program can read in three strings (name, email, password) and validate them. Each string is validated before it is saved to an instance of User class. Then it writes the User object's values to a text file as well as a binary file. The program opens the User's binary file and looks for the third record. Once it locates the record, it reads the data in and saves it to a User object.

yahtzee_v6_userClass_readRandBinaryRec :

- Added int hiScore=0 to User class.
    - Added getHiScore(), setHiScore(int score).
    - Wrote hiScore to text and binary files.
    - Confirmed hiScore is being read from binary file.
- Read 1 record from User's binary file and saved it to an instance of User class.

yahtzee_v11_adminClass :
Added Admin class & moved all of User class to it, so the User class inherits the Admin class.

yahtzee_v12_adminClass_staticMmbr :

- Adding static int ttlRecs and static member() to Admin class and set its value by reading a text file each time one of Admin's constructors were called. This allowed ttlRec to hold its value in between runs.
    - Added ttlRec to wrtTxt(), wrtBin(), readBin().
- Moved signUp() & readInputFile() to User class.
- Created adminPortal() with a menu inside of the admin.cpp. You can only access this menu if Admin is signed in.
- Add max retries in user validation functions
- Added overloaded readBin() that accepts an email.
- Added wrt1Record() and called it in both readBin()
- Added wrtAdminTxt() & wrtAdminBin() as private Admin members
- Added char arguments to wrtTxt()
- Added a switch menu in main() with options to sign up, user login, admin login.
- Fixed User overwriting ttlRec & ID by passing the values from Admin to User in main().
- Changed bool readBin() to long findEmail(), so I can pass the cursor when I need to rewrite a file in binary. If email is not found in binary, then it sets the cursor to -99.

yahtzee_v14_adminClass :
Added cis17A_project2_yahtzee_v8's Player class to main(). Player::yahtzee() runs for 13 rounds and play() handles each player's turn.

yahtzee_v18_admin_aggregateUser :
Admin aggregate an instance of User instead of inheriting it. My group thought the Admin class should aggregate an instance of User class instead of using inheritance. Aggregating requires a lot more code than inheritance.

yahtzee_v19_addYahtzeeClass :
- Added Ismael's Yahtzee, Score_Card, and Dice classes to main().
- Added setUsrHiScore(score) & called it main(), so Admin can rewrite the User's binary and text files.

yahtzee_v20_yahtzee :
- Moved wrtTxt(), wrtBin(), reWrtBin() & reWrtTxt() to User.cpp and passed beginFile to them.
- In main(), I made User rewrite their hiScore directly instead of having admin do it.
- Created an array of Users, so the game can run with 2 players.
    - Altered Yahtzee::play(), so it accepts a double pointer of Users.
    - Play() also returns if player 1 is the winner or not.

ish_df_yahtzee_v4.6 :
- I reorganized and cleaned up the flow of play() by creating prntRound(), rules(), setFinalSC().
- Created a 2nd instance of ScoreCard to hold the player's actual scores.
- Added a conditional if numRolls==3, then force switch case 2 else displayOptions();
- Added a pause(), so it makes the game more readable between rolls
- Created bool isSelected[13] to prevent already selected categories from being displayed in possible points scoreboard. Called it inside of recordScore().
- Reset isSelected() flags in setFinalSC() to stop categories from being overwritten.
- Changed play() to bool play(int &userHiScore).
    - If player 1 is winner AND their current score is larger than user's high score, then
        - Reassign userHiScore's value and return true.

yahtzee_v22_updateYahtzeeWithIsmaels :
Added Ismael's updated classes from ismael_yahtzee_v4.6

yahtzee_v23.1_addPlayrArray
- Aggregated 2 instances of ScoreCard in Yahtzee's private members. Made them static arrays.
- Moved **players Yahtzee's default constructor. This way I can access player[ ] in other Yahtzee functions.
    - Added for(pIndx<nPlayer) to control each player's turn
    - Reset values inside switch() default so it could jump out of all the loops and print final scorecards.
- Added getUpLowSums() to calculate the totals from the scorecard's upper and lower sections.
- Created a specific function to print the player's actual scorecard rather than using the function that prints their possible points.

yahtzee_v24_startGameFunc :
Cleaned up play() by creating startGame() to handle the number of rounds loop
and play() handles each player's turn individually.

yahtzee_v26_moveDice2ScoreCard:

- Moved the creation of Dice dice[ ] and int diceArr[ ] and their code from Yahtzee to ScoreCard.
- Cleaned up play() by adding selectDice(), selectCategory(), and isNewHiScore().
- Moved constant variables to the top of Classes h file

yahtzee_v27_adminDoublePtrArray:
I eliminated redundant code by having Admin::getAllUsr() be the only function that reads the User's binary file. Now getAllUsr() returns a double pointer array with all the user's records read from the binary file. Initially findByEmail(), findByIndx(), and getAllUsr() read the User's binary file.

yahtzee_v28.1_doublePtrAdminConstructor:
Added Admin **usrArr as a private member & allocated memory for it in Admin's default constructor. Now there's an array Admin can reference anywhere in its class.

yahtzee_v28.2_changeFindByFuncs:

- findByEmail() and findByIndx() can search through an array instead of searching a binary file.
- startGame() passes record's location to Yahtzee, so it can rewrite the user's record inside of it rather than in main().
- Cleaned up adminPortal() by putting each case in its own function.
- Updated Admin with survey_v5's Admin class

References

1. Gaddis, Tony. Starting out with C++: From Control Structures through Objects. 9th ed., Pearson, 2018.
2. Lehr, Mark. "2023_Spring_CSC_CIS_17B · ml1150258/2023_spring_csc_cis_17b." GitHub, 2023, https://github.com/ml1150258/2023_Spring_CIS_CSC_17B.

Pseudo Code

1. **Parent: Admin class**

    a. Dilemma. Option 1: User class will be the parent and then the Admin and Player classes will inherit User?

        i. Admin & Player both use User class's variables & functions, but how will the Player ask Admin to test their hiScore variable?

b. Dilemma. Option 2: Admin is the parent, list wrtBinary() as private and then User inherits Admin?
    i. Admin doesn't need hiScore when it creates a new object for itself, but it needs to be able to reassign hiScore.

c. Admin object:
    i. Private members:
        1. Static int num records /* Doesn't hold value between runs */
        2. unique ID
        3. name
        4. password
        5. email
        6. hiScore ? Should this be in User or Admin?
        7. readBinary()
        8. checkHiScore()
        9. setHiScore()

d. Read inputs for 1 Admin record:
    i. Confirm inputs
    ii. Save 1 admin record to object
    iii. Write the object to text and binary files.
    iv. Print message if successful or not

e. ONLY Admin is allowed to read binary files

f. Admin login()
    i. Read first record in Admin's binary file
    ii. Save it to an object
    iii. Test login
    iv. Print message if login was correct or not
    v. If login is correct, then allow them access to admin only functions

g. Read User's binary file:
    i. User login()
        1. Find a record by email
        2. Save it to an object
        3. Test login
        4. Print message if login was correct or not
        5. If login is correct, then allow them to view their profile and play game

    ii. Be able to delete/edit a record or member's of record
        1. Accept an ID or record as an object AND score from User, test if it's bigger than their current hiSCore and update it accordingly

2. **Child: User class**
    a. Create User Class. User and game classes will have to be combined at some point.
    b. User object:
        i. Private members:
            1. Record #
            2. Unique ID
            3. name
            4. password
            5. email
            6. hiScore ? Should this be in User or Admin?
            7. scorecard[ ]
            8. dice array or vector
    c. Sign up for a new account. Read in name, email, password
        i. Confirm user inputs before saving their info to object
    d. Write and append each User to a binary file
        i. ONLY has permission to write to binary file
    e. Write and append each User's to text file
        i. ONLY has permission to write to text file
    f. Play game as guest or as logged in user
    g. When they win they should send their ID or their record as an object AND their current score to Admin, so then Admin can test if it's bigger than the hiScore saved to their record

3. **Game Play**
    a. **Dice**
        i. Function returns a random num between 1 and 6
        ii. Function that saves up to 5 dice in an array or vector to represent which dice the player is keeping towards their score
            1. Needs to add and remove a dice between rolls.
        iii. Print the value of 1-5 different dice
            1. Test dice against 13 categories and print a score for categories player can potential get points from
                a. Ask user if they want to:

            i. Keep one of the potential points
                1. If yes, then turn ends and save points to scorecard
            ii. or keep 1-4 dice and roll again
                1. If yes, get which dice they want to keep and roll the remaining dice again.
                2. Needs to keep count of how many dice the player is keeping on each roll and the value of each dice.
    iv. Allow user to remove dice if they still rolls remaining
        1. Consider using vectors for which dice the player is keeping between rolls?

b. **Scorecard class:** 13 scoring categories + 3 different sums

    i. Upper Section categories
        1. 6 categories that summing each side of dice
            a. Sum function that accepts x number of dice and adds their face value?
        2. Total score from all 6 sides of a dice
        3. Bonus – if score is 63 or over, then +35 pts
        4. Total of upper section

    ii. Lower Section categories: has to check if the dice meet the conditions for each of these categories. If they do, then it needs to return points.
        1. pts = three of a kind ? sum of all 3 dice : 0;
        2. pts = four of a kind ? sum of all 4 dice : 0;
        3. pts = full House (threeKind + twoKind) ? 25 pts : 0;
        4. pts = small Straight ? 30 pts : 0;
        5. pts = large Straight ? 40 pts : 0;
        6. pts = Yahtzee (5 of kind) ? 50 pts : 0;
            a. pts = Yahtzee bonus? ? 100 pts : 0;
        7. pts = chance pts = sum of all 5 dice : 0;

    iii. Total of lower section

    iv. Grand total = total of lower + upper section