Comprehensive Exercise Report

Team <<X>> of Section <<000>>

Andrea Torres Lucas (24451521Z), Virginia Torres Lucas (24451520J)

Requirements/Analysis

Week 2

Journal

The following prompts are meant to aid your thought process as you complete the requirements/analysis portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
  - The client wants an interface for playing BUNCO game
    - Result of rolling the dice
    - For each of the three dice that is equal to the number of the round add 1 point
    - When the numbers in the three dice are equal to the round number add 21 points → BUNCO!
    - When the numbers in the three dice are equal in a round number different from the dice add 5 points → MINI BUNCO!
    - The players keeps rolling until they roll no points
    - There are 6 rounds, 4 players team
    - When the final score points are bigger or equal than 21, that is a WIN
    - When the final score points are lower or equal than 21, that is a LOSE
- After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.
  - Number of tables and number of players
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
  -
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
  - Small child
  - Adults
- Describe how each user would interact with the software
  - One software for a table
- What features must the software have? What should the users be able to do?
  - Write username
  - Roll the dice
  - Read the Rules
  - See the Scoreboard
  - Start a new game
- Other notes:
  -

Software Requirements

- Result of rolling the dice
- For each of the three dice that is equal to the number of the round add 1 point
- When the numbers in the three dice are equal to the round number add 21 points → BUNCO!
- When the numbers in the three dice are equal in a round number different from the dice add 5 points → MINI BUNCO!
- The players keeps rolling until they roll no points
- There are 6 rounds, 4 players team
- When the final score points are bigger or equal than 21 that is a WIN
- When the final score points are lower or equal than 21 that is a LOSE

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
    - UserName String
    - Score
    - Players
    - Dice
    - Bunco Game
    - Botton 'Roll The Dice'
    - Button 'Menu'
    - Button 'Menu/New Game'
    - Button 'Menu/Rules'
    - Button 'Menu/Scoreboard'
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Result from rolling the dice
    - Cumulative score of every player
    - Names of the players
    - Rules
    - Scoreboard
    - Number of actual round
    - Points added in every round to the players
    - Round Winner
- What equivalence classes can the input be broken into?
- What boundary values exist for the input?
    - None for the inputs
- Are there other cases that must be tested to test all requirements?
  The tests have to validate the rules of the game.
    - The score of every player (it wins if it only has 21 points or more)
    - The number of times a user rolls the dice so that it is known when it is the turn of the players (in case that one of the dice was the same number of the round number, the user can roll again the dice, if not the dice is rolled one time)
    - The number of the round (it is incremented when a player has won a round)
    - When the number of every dice is equal to the round number it wins 21 points (round 4, dice 1 is 4, dice 2 is 4 and dice 3 is 4), if it is not equal than the round it should add 5 points (round 2, dice 1 is 4, dice 2 is 4 and dice 3 is 4)

    - When a player wins the last round, it has to start the following round (because one of the rules of Bunco is that if in the last turn a player has earned points, it must continue rolling until it gets no points)
- Other notes:

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Test ID	**Description**	**Expected Results**	**Actual Results**
1	Checking players scores	When a player wins 21 or more points, it is considered the winner of the round and the scores turn to 0 for starting the next round	Expected result
2	Increment of rounds	When a player wins a round the number of rounds is incremented and when it reaches number 6 (6 rounds), the game ends.	Expected result
3	“Roll the dice” button	When this button is clicked, the dice and their random value must be shown on the screen and the values of the points of the players are updated. When the game has ended, this button makes no action.	Expected result
4	Scoreboard values	While the game is being played, the scoreboard is updated with the winner values.	Expected result
5	Menu button and inside dialog	The buttons of the dialog (new game, scoreboard and rules) must show the information they indicate.	Expected result
6	Points that every player wins every time the dice are rolled	Showing if they have won points, a little bunco or a big bunco	Expected result

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

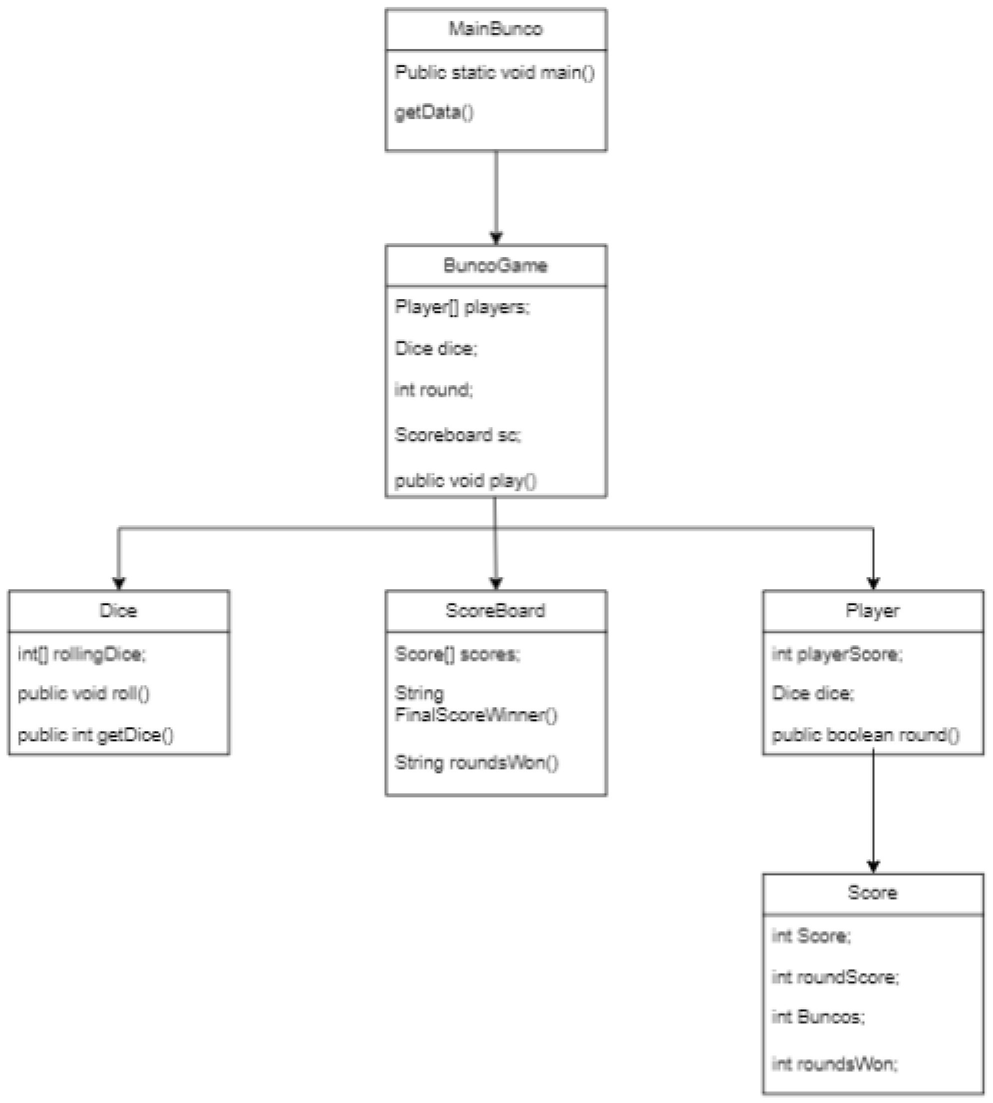
The following prompts are meant to aid your thought process as you complete the design portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- List the nouns from your requirements/analysis documentation.
    - Players
    - Round
    - Roll
    - Menu
    - New Game
    - Rules
    - Scoreboard
    - Points
- Which nouns potentially may represent a class in your design?
    - Players
    - New Game
    - Dice
    - Points
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Player (points)
    - Bunco Game (players)
    - Score (dice1, dice2, dice3, player)
- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
    - 1st design → The actual player would have to register, introduce its name in a command or input line, then the name would be shown (printed onto the screen) in text. Bunco game can be played by 4 players. This game is played in a round-robin manner where each player is going to take a turn. When the player gets its turn to play it has to roll the dice (set of three dice) by clicking the bottom 'Roll' and depending on the results of the dice the player will get its points and the accumulated value would be printed on the screen just below its name. When a player has scored 21 or more points there will be an instant win, and the winner of that round will be announced. At the end of the game, when the final round is played, a summary of the scores of each player in each round and the number of the buncos played will be reported in a scoreboard.The winner with the most rounds won is the overall game winner. If there is a tie, the player with more Buncos wins. Players can check the scoreboard at any time, as well as the rules, and can also start a new game.

    - 2nd design → At the start of the game the player is assigned a fixed name (ex. Player1, player2, etc). In this design every player can roll the dice, so that real players can play the game with the same interface. The difference from the first design is that in this one there is only one player rolling the dice, the other players play virtually, not by pushing the bottom of rolling the dice. About the points in this design, they are going to perform the same as in the first design: depending on the value of the three dice, it will be added to the points of a concrete player name. When some of the players have a score of 21 or more in a round, it will be a win and the winner will be announced. The player who has won more rounds will be the winner. The new game will start resetting all the last values and a new game will start.
- Which design do you plan to use? Explain why you have chosen this design.
  2nd design. As we started creating the game without a graphic interface and the first results were done automatically. When we incorporated the graphic interface it was easier to make the game this way so that every player has to push the button to play, not just one.
- List the verbs from your requirements/analysis documentation.
    - Adding points
    - Checking points
    - Changing player
    - Rolling dice
    - Starting game
    - Printing name of the actual player
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Adding points (Points class)
    - Checking points (Points class)
    - Starting game (New game)
- Other notes:

Software Design

<<Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will be used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.***>>



Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.
    - To distribute better the work in group we have been using the Agile Methodology
    - Design considerations: assumptions, dependencies, constraints, requirements, objectives, methodologies
- Other notes:

Implementation Details

<<Use your notes from above to write code and complete this section of the formal documentation with a README for the user that explains how he/she will interact with the system.>>

The user must download the .zip file of the github repository in order to run the programme in its IDE.
As it is a java project, the main class (MainBunco.java) must be run as a java project in order to interact with the game.
The game starts with the player1 sequentially. To start playing the user must press the button “Roll the dice”.
At any moment the players can take a look at the scoreboard, the rules or start a new game if needed by clicking the “Menu” button.

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
  - For coordinating the data that was going to be shown in the graphic interface we needed to add some methods and change some variables, to get every piece of data we wanted.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
  - **BuncoGame.java → class Score{} & class Player extends Score{}**
    - By visualizing results of the methods in the terminal in method main from MainBunco class.
    - In this class we have the dynamics of the game, where the players are selected to play its corresponding turns, where the rounds are added once someone has reached the maximum of points and where we can get any data from any player.
    - Inside this class we use classes such as Dice, Scoreboard and Player with its Score.
  - **Dice.java**
    - By visualizing results of the methods in the terminal in method main from MainBunco class.
    - This class only gives a random value from 1-6 to the three playing dice.
  - **Scoreboard.java**
    - By visualizing results of the methods in the terminal in method main from MainBunco class.
    - This class helps the game have an actualized scoreboard with all the points from each user, and also to know which player has more rounds won, buncos won and little buncos won
    - Inside this class we use the class Score, which is for every player
  - **MainBunco.java**
    - By visualizing results of the methods in the terminal in method main from MainBunco class.
    - This class is where the game starts and contains all the code for the graphic interface, where the JFrame is created, buttons, labels, panel, sounds added and design of them.
    - Inside this class we need instances of the classes BuncoGame, Player, Dice and the ones that are from JFrame.
- Other notes:
  - <<Insert notes>>

Testing Details

<<Use your notes from above to write your test programs and complete this section of the formal documentation by creating a list of your test programs along with descriptions of what they are testing. You will also complete the black-box test plan by running the program and filling in the Actual Results column.>>

Basically, in the MainBunco() class is where all the tests were completed.
We could see if there was something wrong in the terminal, so that helped us to correct any error.

The test could be done by seeing the results in the terminal by using System.out.printl() in the different methods. Also the testing could be checked in the graphical interface.

Presentation

Instructions:Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - The final project is a completely functional BUNCO game, based on 4 players playing at the same time. In the graphic interface the game starts by clicking ‘Roll the dice’, also you can see the button Menu that leads to Scoreboard, so you can see at any time the state of the game and who is winning at the moment, New game, so you can start a new game maybe when you’ve finished or whenever you want, and lastly, button rules, where you can look at the rules at any time if you forget something at a time or if you don’t know how to choose the final winner because there is a tie.
- Describe your requirement assumptions/additions.
    - We have used the Visual Studio IDE in which we had before installed all the extensions for having the libraries needed and executing java.
    - In java we had to use the JFrame library in order to create the graphical interface. Also, for the buttons we had to use the library Action Listener to create an action for every button.
    - As an addition to the graphical interface we downloaded sounds for the buttons and when there was a winner in the game or any player had won a determined value of points. These sounds were added to help us to follow the game and make it more interactive.
- Describe your design options and decisions. How did you weigh the pros and cons of the different designs to make your decision?
    - As we chose java for our project, design has not so many actions such as html and css could have given to us because in css there are more internal methods for designing, not as in java.
    - The main problems were about adjusting texts to a frame ,but were resolved by using Grids.
    - As we started creating the game without a graphic interface and the first results were done automatically. When we incorporated the graphic interface it was easier to make the game as in the second design. In the first design we had to use more JFrame classes so we decided to implement the 2nd design.
- How did the extension affect your design?
    - Some of the classes were created in the same java file.
    - The sounds were added to help us to follow the game and make it more interactive.
- Describe your tests (e.g., what you tested, equivalence classes).
    - In MainBunco class it was needed to do a lot of tests in order to coordinate the data of our methods in the classes and the graphic interface
    - In the terminal we checked how the number of points, the dice and the rounds changed.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
    - It helps when coding so that we don’t miss any essential code needed.
    - Simplicity after all, testing in small blocks.

    - UML diagrams to have a clearer vision at first of the program before developing the software.
- What functionalities are you going to demo?
    - We can show the entire game, but that will need a lot of time, so we'll show how the rounds keep changing and who wins them until it reaches round 6, which is the last one. Also, we'll see when you can have a little bunco(5 points) and a bunco(21 points), and the extra points shown in the rules. We'll see how you can create a new game, see the scoreboard and have a look at the rules.All these events have different sounds added, you can know when an event happens because of the sound effects added.
- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
    - Virginia Torres Lucas → presentation of the demo of the game
    - Andrea Torres Lucas → brief explanation of the code, design of the game
- Other notes:
    - <<Insert notes>>

Andrea:

- Presentation of ourselves
- Description of our game
- Description of the IDE used
- Demo of the game

Virginia:

- Description of classes
- Overview of graphic interface of the game