PBL Project Morra Odds and Evens Game

Software Development



***Lecturer:***

*Adriana E. Chis*

***Group F:***

[*Niall Deely*](https://app.slack.com/team/UTG4XCMK9)

*George Tudor Pascu*

*Alexandre Zurcher*

**“Morra Odds and Evens” game rules**

1. Two players game. One player is the Odds and the other is Evens.
2. In each round of the game, the players will simultaneously hold out between 1 and 10 ﬁngers.
3. The winner of the round is decided based on the sum of ﬁngers shown by both players, namely if the sum is an even number then the “Evens” player wins, otherwise if the sum is an odd number then the “Odds” player wins.
4. The winner of the round receive 3 points.
5. In addition, the player whose number of fingers is closer to the sum, receives two extra points.
6. The winner of the game is the ﬁrst player who accumulates 12 points.

**Application development of Morra Odds and Evens**

1. At the beginning of each game the user will be prompted to choose whether he/she would like to be either the “Odds” or the “Even” player.
2. In each round of the game, the user must decide the number of ﬁngers to show (i.e. between 1 and 10).
3. Similarly, in each round of the game the computer will randomly pick one number between 1 and 10.
4. In each round, the game displays the computer’s choice.
5. After each round the game displays the number of points each player has, and whether the user or the computer won the round.
6. The game finished when one of the players accumulated 12 points.
7. At the end of a game, the game displays who the winner is, and a history of the numbers of ﬁngers shown by both the user and the computer per round.
8. Once a game has ﬁnished the application asks the player if he/she would like to play another game.
9. At the end of all games, display a history of games played. The history shows, for each game, the number of rounds won and lost by the human player, and how many even and odd numbers have been chosen by each player, and the extra points received by each player per game. All the history elements of the game should be coded using arrays.

**Project deliverables**

1. **The project submission should include a project report outlining very clearly who has coded which part of the code:**

The code has been segmented in parts Alexandre Zurcher contributed with gameEnd () and playRound() methods on MorraGame class.

Each integrant have analysed, revised and tested the code

1. **The report should include a description of the input, main processing and output (IPO):**

**Inputs**: The first input is the answer if the user wants to play the game (y/n). The second input is the team chosen (numbers: 0 for evens, 1 for odds). The third inputs are the player moves for the respective rounds (numbers from 1 - 10). After the first game the software prompt the user if he wants to play again. If yes, the second and thirds inputs described above are repeated

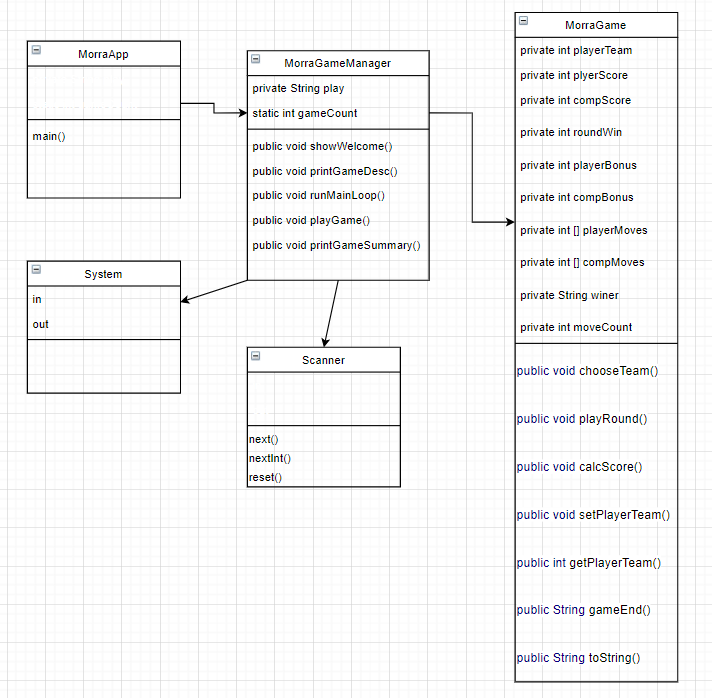
**Process:** The first batch process is to compute/check who was the winner/loser, who get the bonus and calculate the score for player and computer on each respective round. The second batch process is to compute/check who was the winner/loser of the overall game based on the overall score.

**Outputs:** The first batch of outputs appears in the end of the first round and are the messages showing the number (moves) chosen by the user and the computer in the round, who was the winner/loser in the round and how much is the score for user and computer in the end of the round.

The second batch of outputs appears in the end of each game and shows who won the game and each moves chosen by user and computer per round.

The third batch of outputs occurs with a game summary there you can see who the overall winner of the series of games played. It is possible to see also how many games the user won and lost, how many times user chooses evens and odds and how many extra points the user and the computers received on the overall series of games.

1. **Class diagram for your application.**



1. **Decisions you take in designing and implementing your project should be specified in the report**

**Three classes**

It was ultimately decided that this project would be divided into 3 classes; MorraApp, MorraGameManager, MorraGame.

**MorraGame:** Controls all functionality for a single game of Morra and stores all relevant information (teams, moves, round wins, game winner, etc).

**MorraGameManager:** Controls the functionality for interacting with the user about playing a game/games, storing the games played, and printing out the game summary of each played when the user has indicated they’re done playing.

**MorraApp:** Contains the main method, which simply runs an instance of MorraGameManager, and could easily be rewritten to run a single MorraGame instance if someone wants to just play one game.

**MorraGameManager**

As ArrayLists could not be used, an array of size 10 was chosen to store games. If the user exceeds this, they are asked if they would like to continue playing. If they do, an array with space for 10 more games is created, the previous games are copied to it, and the original array is overwritten by the new one.

**MorraGame**

Player and CPU moves were stored in an array of size 6, because there is no possible scenario where more than 6 moves can be played in a single game. A game requires a minimum of 3 moves, and a maximum of 6 for one of the players to reach or exceed 12 points.

The toString method was used in the MorraGame class to print the summary of a game in a string, allowing the MorraGameManager class to print each class to show the summary when the user is finished playing, rather than relying on printing a method for each class.

Grammar fixes were used in the toString method of MorraGame to prevent grammar mistakes such as “1 even numbers” instead of “1 even number”.

**MorraGame/MorraGameManager**

Line breaks and formatting (eg. “=== GAME 1 ===”) were added to improve console readability in certain sections.

1. **The report should also include a section on manual testing (describing testing scenarios and corresponding input data) you performed to show that the game is working according to the specification (here you could include screenshots of the tests performed).**

Extensive testing was done by all team members to try to find bugs in the code.

Bugs found and fixed:

The code would crash when the scanner was expecting a nextInt(), but received a String. To fix this, we implemented hasNextInt() to check for an int in advance, and looped the code until a valid int was input.

The toString method of MorraGame would output incorrect grammar where the number was 1, such as “You chose 1 odd numbers, and …”. To fix this, the code was changed so that numbers were checked beforehand, and in cases where the number was 1, the word “number” was used instead of “numbers”.

OBS

1. A separate submission of a short summary that outlines the tasks contributed to the project be each team member is required to be submitted by each team member
2. The code should be commented to explain what the code is doing. The code should follow good programming practice e.g. naming conventions, indentation.
3. An in-class test will take place following the submission of your group project. This test will include questions based on the topic of the project and submitted application. Please check Moodle for the exact date of the in-class test.