CSE2260 - Principles of Programming Languages
PROJECT -1 REPORT

Java:

In this project I implemented all the rules of the Mangala Game according to the given link in project document. My code is running effectively without any missing information or any misleading rules. I tested it several times and I did not find any wrong moves.

I explained the details of the program with screenshots in the following:

The **User class** (Line 16) which I defined the player and its features, such as where its wallet location, where rival's wallet location and whose turn this is. With the help of constructor, I created two instances of this class and kept them in a user array with size of two.

```
  // User defining class
 public static class User {
     // To determine the wallet location of the user. mangalaGameBoard[6] for the first player and
     // mangalaGameBoard[13] for the second player
     public final int userWalletLocation;
     // while distributing the pieces to skip the rival's wallet we define this
     public final int rivalWalletLocation;
     // To determine whose turn is this
     public final int whoseTurn;

// User class constructor
     public User( int turn, int playerWalletLocation, int rivalWalletLocation) {
         this.userWalletLocation = playerWalletLocation;
         this.rivalWalletLocation = rivalWalletLocation;
         this.whoseTurn = turn;
     }
 }
```

In gameMechanic() method, the game is constructed according to the given position. It is explained in detail in the codes lines for game rules section of my report.

```
   public boolean gameMechanic(int chosenPosition) {
       int pieceAmount = mangalaGameBoard[chosenPosition];
       int initChosenPosition = chosenPosition;
       int initHandAmount = pieceAmount;
 // if there is more than 1 piece it distributes normally
       if (pieceAmount>1){
           mangalaGameBoard[chosenPosition] = 1;
           pieceAmount = pieceAmount - 1;
       }
  // if there is only 1 piece it moves the piece in to the next hole
       else mangalaGameBoard[chosenPosition] = 0;

       while (pieceAmount > 0) {
          // System.out.println("chosen position track :"+chosenPosition);
           chosenPosition = (chosenPosition + 1) % 14;
           pieceAmount = pieceAmount - 1;
           if (chosenPosition == who.rivalWalletLocation) {
              chosenPosition = (chosenPosition + 1) % 14;
           }
           mangalaGameBoard[chosenPosition] = mangalaGameBoard[chosenPosition] + 1;
       }
```

There is a **mangalaGameBoard** array (Line 7) of size 14. Two locations of it for the wallet locations of the players and the rest are the holes for the stones. I declared the wallet positions manually. For user1 the wallet location is 6 and for user2 its 13 by default.

```
public static void main(String[] args) throws IOException {
    Mangala game = new Mangala();
    int user1WalletLocation = 6;
    int user2WalletLocation = 13;
    User user1 = new User( turn: 1,user1WalletLocation,  rivalWalletLocation: 13);
    User user2 = new User( turn: 2,user2WalletLocation,  rivalWalletLocation: 6);
    users[0] = user1 ;
    users[1] = user2 ;
```

The game starts with the **flipCoin()** method (Line 108). User presses enter button, flip coin method is executed and generates a random number. So, one of the two players starts the game every time with randomly.

```
    public int flipCoin() {
    Random randomNum = new Random();
    int result = randomNum.nextInt( bound: 2);
    if (result == 0) {
        return 1;
    } else {
        return 2;
    }
}
```

Initializing the game, with filling the holes with 4 stones each and wallet locations with 0.

```
// Starting the game
public void startGame() {
    int STARTING_AMOUNT_OF_PIECES = 4;
    Arrays.fill(mangalaGameBoard, STARTING_AMOUNT_OF_PIECES);
    for (User user : users) {
        mangalaGameBoard[user.userWalletLocation] = 0;
    }
    who = users[flipUser-1];
}
```

Game is played within a loop (Line 52). So, until one of the players empties all its holes, the game continues. **isEmpty** boolean variable in the **isGameOver()** method is checked every time a player makes a move.

```
// Returns true if the game is over (one side has no pieces)
public boolean isGameOver() {
    boolean isEmpty = false;
    int user1Sum = sum(users[0]);
    int user2Sum = sum(users[1]);
    if(user1Sum == 0 || user2Sum == 0){
        isEmpty = true;
    }
    return isEmpty;
}
// calculates the number of pieces on player's side of the board
public int sum(User player) {
    int sum = 0;
    int start = (player.rivalWalletLocation + 1) % 14;
    for (int i = start; i < start + 13 / 2; i++) {
        sum += mangalaGameBoard[i];
    }
    return sum;
}
```

Every time a player makes a move the **printCurrentBoardStatus()** method sets the current view of the game board on the console (Line 128-148).

```
To start the game please press enter and flip a coin..

User 2 will start..

              (1) (2) (3) (4) (5) (6)
             | 4 | 4 | 4 | 4 | 4 | 4 |
              <<<<<<<<<<<<<<<<<<<<<<
User 2> |0  |----------------------|  0| <User 1
              >>>>>>>>>>>>>>>>>>>>>>
             | 4 | 4 | 4 | 4 | 4 | 4 |
              (1) (2) (3) (4) (5) (6)
User 2, enter a  hole number (1-6):3
-----------------------------------------------------------------
              (1) (2) (3) (4) (5) (6)
             | 5 | 5 | 1 | 4 | 4 | 4 |
              <<<<<<<<<<<<<<<<<<<<<<
User 2> |1  |----------------------|  0| <User 1
              >>>>>>>>>>>>>>>>>>>>>>
             | 4 | 4 | 4 | 4 | 4 | 4 |
              (1) (2) (3) (4) (5) (6)
The last one is in your wallet. Great!! Play again User 2
User 2, enter a  hole number (1-6):1
-----------------------------------------------------------------
```

Code lines of the game rules:

If all the holes of a user are empty, then it collects the remaining stones of the other user (Lines 64-74) and the final scores are printed the according to this adding operation.

```
//if all the holes of user1 are empty then it collects the remaining stones of user2.
if(user1Sum == 0){
    for(int i = 0 ; i < 6 ; i++ ){
        mangalaGameBoard[users[0].userWalletLocation] += mangalaGameBoard[i+7];
    }
}
//if all the holes of user2 are empty then it collects the remaining stones of user1.
else if(user2Sum == 0){
    for(int i = 0 ; i < 6 ; i++ ){
        mangalaGameBoard[users[1].userWalletLocation] += mangalaGameBoard[i];
    }
}
```

Determining the winner with the check of who has the more stones (Lines 80-85). I they are equal then it is a draw

```
//default random winner value
int winner = 0;
// who has the more stones is the winner
if(mangalaGameBoard[users[1].userWalletLocation] < mangalaGameBoard[users[0].userWalletLocation]){
    winner = 1;
}else if (mangalaGameBoard[users[0].userWalletLocation] < mangalaGameBoard[users[1].userWalletLocation]){
    winner = 2;
}else{
    System.out.println("The game is over! No winner :)  ");System.exit( status: 0);
}
System.out.println("The game is over! The winner is User "+ winner);
System.out.println("Scores:\nUser 1: " + mangalaGameBoard[users[0].userWalletLocation]+
        " User 2: " + mangalaGameBoard[users[1].userWalletLocation]);
System.exit( status: 0);
```

If last stone is put in an empty hole of playing user then it collects other player's opposite hole stones (if there is at least one stone) into its wallet.

```
// if last piece is put in an empty hole of yours then
// you get other user's opposite hole pieces into your wallet
if(who.whoseTurn == users[0].whoseTurn && chosenPosition != who.userWalletLocation && chosenPosition < 6 &&
        mangalaGameBoard[chosenPosition] == 1 && mangalaGameBoard[12- chosenPosition] != 0){
    mangalaGameBoard[who.userWalletLocation] += 1 + mangalaGameBoard[12 - chosenPosition];
    mangalaGameBoard[chosenPosition] = 0; mangalaGameBoard[12 -chosenPosition] = 0;
    System.out.println("That's an empty hole. Great job User "+users[0].whoseTurn +", you got your rival's stones");
}
else if (who.whoseTurn == users[1].whoseTurn && chosenPosition != who.userWalletLocation && chosenPosition > 6  &&
        mangalaGameBoard[chosenPosition] == 1 && mangalaGameBoard[12 - chosenPosition] != 0){
    mangalaGameBoard[who.userWalletLocation] += 1 + mangalaGameBoard[12 - chosenPosition];
    mangalaGameBoard[chosenPosition] = 0; mangalaGameBoard[12 - chosenPosition] = 0;
    System.out.println("That's an empty hole. Great job User "+users[1].whoseTurn +", you got your rival's stones");
}
```

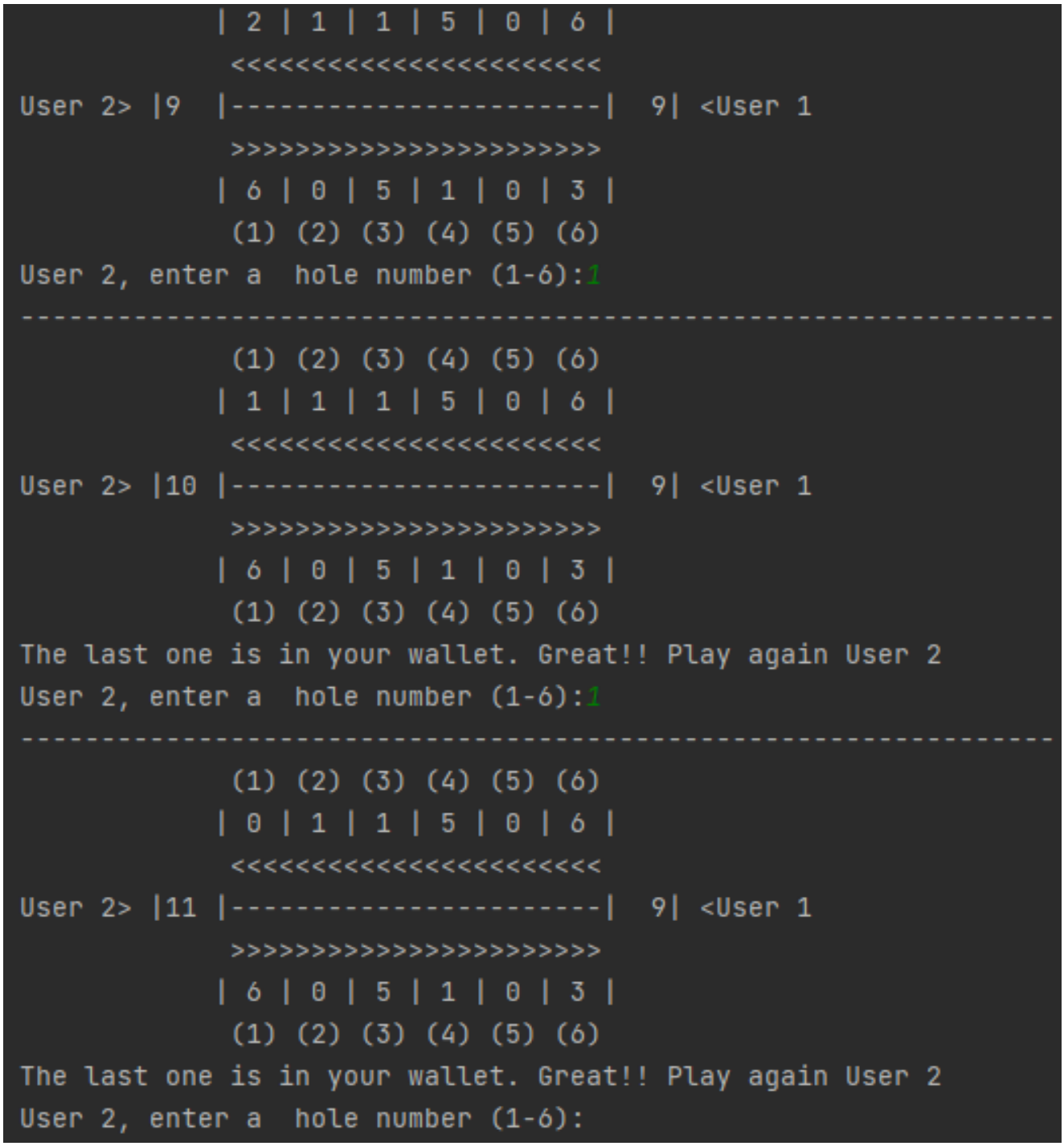
If last piece is put opponent's hole and make it even then you get your rival's stones into your wallet.

```
 // if last piece is put opponent's hole and make it even
 // then you get your rival's stones into your wallet
 if(who.whoseTurn == users[0].whoseTurn &&(initChosenPosition+initHandAmount)>7
         && mangalaGameBoard[chosenPosition] % 2 ==0){
     mangalaGameBoard[who.userWalletLocation] += mangalaGameBoard[chosenPosition];
     mangalaGameBoard[chosenPosition] = 0;
     System.out.println("That's an even number. Great job "+users[0].whoseTurn +", you got your rival's stones");
 }
 if(who.whoseTurn == users[1].whoseTurn &&(initChosenPosition+initHandAmount)>14
         && mangalaGameBoard[chosenPosition] % 2 ==0){
     mangalaGameBoard[who.userWalletLocation] += mangalaGameBoard[chosenPosition];
     mangalaGameBoard[chosenPosition] = 0;
     System.out.println("That's an even number. Great job "+users[1].whoseTurn +", you got your rival's stones");
 }
```

If last piece is put in the player's wallet, then this player can play one more round.

```
   // if last piece is put in the player's wallet then this player can play one more round
   if (!isGameOver() && chosenPosition == who.userWalletLocation) {
       System.out.println("The last one is in your wallet. Great!! Play again User " + who.whoseTurn);
       return true;
   }
   return false;
```

GAME DYNAMIC SCREENSHOTS:



```
            | 1 | 5 | 1 | 5 | 5 | 5 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |2  |----------------------|  2| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 5 | 5 | 5 | 1 | 5 | 1 |
             (1) (2) (3) (4) (5) (6)
User 1, enter a  hole number (1-6):5
-----------------------------------------------------------------
That's an even number. Great job 1, you got your rival's stones
             (1) (2) (3) (4) (5) (6)
            | 1 | 5 | 1 | 5 | 0 | 6 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |2  |----------------------|  9| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 5 | 5 | 5 | 1 | 1 | 2 |
             (1) (2) (3) (4) (5) (6)
User 2, enter a  hole number (1-6):2
-----------------------------------------------------------------
That's an even number. Great job 2, you got your rival's stones
             (1) (2) (3) (4) (5) (6)
            | 2 | 1 | 1 | 5 | 0 | 6 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |9  |----------------------|  9| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 6 | 0 | 5 | 1 | 1 | 2 |
             (1) (2) (3) (4) (5) (6)
User 1, enter a  hole number (1-6):
```

```
             (1) (2) (3) (4) (5) (6)
            | 0 | 1 | 2 | 6 | 3 | 0 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |20 |----------------------| 12| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 0 | 0 | 1 | 2 | 0 | 1 |
             (1) (2) (3) (4) (5) (6)
User 2, enter a  hole number (1-6):4
------------------------------------------------------------------
             (1) (2) (3) (4) (5) (6)
            | 1 | 2 | 3 | 1 | 3 | 0 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |21 |----------------------| 12| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 1 | 0 | 1 | 2 | 0 | 1 |
             (1) (2) (3) (4) (5) (6)
User 1, enter a  hole number (1-6):1
------------------------------------------------------------------
That's an empty hole. Great job User 1, you got your rival's stones
             (1) (2) (3) (4) (5) (6)
            | 1 | 0 | 3 | 1 | 3 | 0 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |21 |----------------------| 15| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 0 | 0 | 1 | 2 | 0 | 1 |
             (1) (2) (3) (4) (5) (6)
User 2, enter a  hole number (1-6):
```

```
            | 0 | 0 | 1 | 0 | 0 | 0 |
             (1) (2) (3) (4) (5) (6)
The last one is in your wallet. Great!! Play again User 1
User 1, enter a  hole number (1-6):3
-----------------------------------------------------------------
             (1) (2) (3) (4) (5) (6)
            | 1 | 0 | 0 | 0 | 0 | 0 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |25 |----------------------| 21| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 0 | 0 | 0 | 1 | 0 | 0 |
             (1) (2) (3) (4) (5) (6)
User 2, enter a  hole number (1-6):1
-----------------------------------------------------------------
             (1) (2) (3) (4) (5) (6)
            | 0 | 0 | 0 | 0 | 0 | 0 |
             <<<<<<<<<<<<<<<<<<<<<<
User 2> |26 |----------------------| 21| <User 1
             >>>>>>>>>>>>>>>>>>>>>>
            | 0 | 0 | 0 | 1 | 0 | 0 |
             (1) (2) (3) (4) (5) (6)
The game is over! The winner is User 2
Scores:
User 1: 21 User 2: 27

Process finished with exit code 0
```

References:

- https://github.com/create/commandline-mancala/blob/master/src/Mancala.java
- https://www.mangala.com.tr/mangala-nasil-oynanir