BACS2063 Data Structures and Algorithms

ASSIGNMENT 202005

Student Name : Joan Hau

Student ID : 20WMR09465

Programme : RSF – Bachelor of Computer Science (Honours) in Software Engineering Year 2 Semester 1

Tutorial Group : 5

Team Name :
1. Joan Hau
2. Lee Ling
3. Cheong Yin Lam
4. Lim Ming Yeu
5. Tan Yong Kang

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University College's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Student's signature

9/8/2020

Date

Table of Contents

BACS2063 Data Structures & Algorithms - Assignment Rubrics			Name: Joan Hau		
Programme: RSF2			**Group/Team:** Joan Hau Lee Ling Cheong Yin Lam Lim Ming Yeu Tan Yong Kang		

No.	Aspect	Developing 0 - 4 marks	Approaching 5-7 marks	Ideal 8-10 marks	Score	Remarks
1	**Idea (based on overview and user guide)**	The idea has a low degree of creativity and novelty among all students submissions.	The idea has a moderate degree of creativity and novelty among all students submissions.	The idea has a high degree of creativity and novelty among all students submissions.		
2	**Entity class diagram (Team Mark)**	Reflects poor ability to analyze the entities required in the application.	Reflects moderate ability to analyze the entities required in the application.	Reflects excellent ability to analyze the entities required in the application.		
3	**Implementation of entity class(es)**	Entity class(es) is implemented incorrectly.	Entity class(es) is implemented correctly but is inconsistent with the entity class diagram.	Entity class(es) is implemented correctly and is consistent with the entity class diagram.		
4	**Collection abstract data type (ADT) specification**	ADT specification is missing 2 or more of the elements: conciseness, correctness, completeness.	ADT specification is missing 1 of the elements: conciseness, correctness, completeness.	ADT specification is concise, correct and complete.		
5	**Implementation of collection ADT**	Reflects poor ablity to synthesize to create new collection ADT.	Reflects moderate ablity to synthesize to create new collection ADT.	Reflects excellent ablity to synthesize to create new collection ADT.		
6	**Selection of collection ADTs for the client program**	ADT is not appropriate and weak justification is provided.	ADT is appropriate and some justification is provided.	ADT is appropriate and sound justification is provided.		
7	**Implementation of client program**	Reflects poor ablity to synthesize to create new solutions using appropriate ADTs **with use of standard Java naming convention**.	Reflects moderate ablity to synthesize to create new solutions using appropriate ADTs **with use of standard Java naming convention**.	Reflects excellent ablity to synthesize to create new solutions using appropriate ADTs **with use of standard Java naming convention**.		
8	**Overall solution (Team Mark)**	Overall solution demonstrates low ability to apply knowledge and skills of data structures and algorithms.	Overall solution demonstrates moderate ability to apply knowledge and skills of data structures and algorithms.	Overall solution demonstrates high ability to apply knowledge and skills of data structures and algorithms.		
9	**Team skills - contribution to integration of modules.**	Student demonstrates poor collaborative skills in integrating the application.	Student demonstrates moderate in collaborative skills in integrating the application.	Student demonstrates excellence in collaborative skills in integrating the application.		
10	**Team skills - peer assesssment**	Student demonstrates poor levels in interactive communications, relationships and collaborative skills in managing relationships in the team.	Student demonstrates moderate ability in interactive communications, relationships and collaborative skills in managing relationships in the team.	Student demonstrates excellence in interactive communications, relationships and collaborative skills in managing relationships in the team.		
				Total / 100		

CLO2 **Cognitive Skills** relates to thinking or intellectual capabilities and the ability to apply knowledge and skills. The capacity to develop levels of intellectual skills progressively begins from understanding, critical/creative

CLO3 **Team Skills** include: **Relationship-building:** Ability to build strong relationship, interact and work effectively with people to attain the same objectives.
Respect: Ability to identify and respect the attitudes, behaviours and beliefs of others.

BACS2063 Data Structures and Algorithms - Team Skills Peer Assessment				
Relationship-building:	Ability to build strong relationship, interact and work effectively with people to attain the same objectives.			
Respect:	Ability to identify and respect the attitudes, behaviours and beliefs of others.			
Instructions: Indicate marks and justification for all team members' including yourself.				
Name	**Relationship (5 marks)**	**Respect (5 marks)**	**Total marks (10 marks)**	**Justification for the marks awarded (Required)**
Joan Hau	5	5	10	All of us give the best cooperation for this assignment.
Lee Ling	5	5	10	All of us give the best cooperation for this assignment.
Cheong Yin Lam	5	5	10	All of us give the best cooperation for this assignment.
Lim Ming Yeu	5	5	10	All of us give the best cooperation for this assignment.
Tan Yong Kang	5	5	10	All of us give the best cooperation for this assignment.

1. Introduction to the Game

The game that developed by our team is the connect four games. It is a multiple player connection board game, in which the players choose a token and then take turn dropping token into an eight-column, nine row vertically suspended grid. The token will occupy the lowest available position within the row. The player can be win in different way at least the number of tokens is connected to each other. The game can be win in the token connect to each other vertically, horizontally or oblique. Connect Four is a solved game. The first player can always win by playing the right moves.

2. User Guide

The game that develops by our team is the connect four game. At the beginning of the game, it will display the logo of our game. A menu will be provided for the players to select which step they want to proceed. Based on the input given by the players, the game will proceed to following pages:
(1) Play Game; (2) Ranking Board; (3) Exit

Input '1' (Play Game):
When the player key in the value '1', the game will continue display a sub-menu for the players to make the decision whether they want to have a (1) classic mode or (2) tournament mode.

Input '1' (Classic Mode):

*The number of the player for the classic mode is set by the player.

STEP 1:
When the player input '1', the players will need to enter the total number of players for the particular round. After that, the game will ask the players to enter their name. The system will validate the input of the players. The players are not allowed to enter the same name at the player registration. Duplicate player name will make the players confuse who is the next player to insert the token. Then, the players can choose the token they want in alphabet (Aa - Zz). Duplicate token is not allowed to enter to the system which means each of the player have their own specific token. The player can only choose (Aa - Zz) for token, the system will prompt an error message if there is any invalid input detected.

STEP 2:
The player details will be displayed after the player entered all the player name. Then, the system will ask the player to choose the mode for the game. The game can be carried out in various types of mode which include (1) Simple Mode – Connect 3; (2) Medium Mode – Connect 4; (3) Hard Mode – Connect 5. The difference between the mode is the connection of the token.

STEP 3:
Thirdly, the system will display the board of the game for the players to insert the token. The system will display the player name in order to inform the player to insert the token. The player can only choose which column (from 1 to 8) to insert the token. The token will be automatically insert to the lowest position of the row which is available.

STEP 4:
Fourthly, if the number of players more than 2, there will be having more than one round for the game. The round of the game is depending on the number of players deduct by one. For example, if there are four players in the game, then the game will be having 3 rounds. When first winner appears, the winner will be continuing play with the following player in the game.

STEP 5:
The game will display the winner for each round. The total token used for each of the player will also displayed after one round. Besides, if there is no next round to be continue in the game, the game will display the ranking of the player which include the total player, total time spent and the player details. The ranking will display the highest score of the game and the winner for the overall game.

Input '2' (Tournament Mode):

*The number of the player can only be four or eight players.

STEP 1:
The player will need to enter the number of players for the game. The number of players that allowed for this mode is only four players or eight players. The game will require the players to enter their name and token before they start the game.

STEP 2:
After that, the system will display the tournament tree so that the players can check their own sequence. The player will need to choose the mode for the game. The mode consists of (1) Simple Mode – Connect 3; (2) Medium Mode – Connect 4; (3) Hard Mode – Connect 5.

STEP 3:
The game will be started and the system will display the player turn. The player will need to enter the column number in order to insert the token into the board.

STEP 4:
After each round of the game, the winner name will be displayed to inform the player who is the winner for the particular round. The tournament tree will also be displayed which allow the player to observe the result among them.

STEP 5:
Then, when all the player finished the game. The game will display the total number of tokens used by each of the player and also display the tournament tree of the game. The players are able to observed the final winner for the game based on the tournament tree displayed.

Input '2' (Ranking Board):

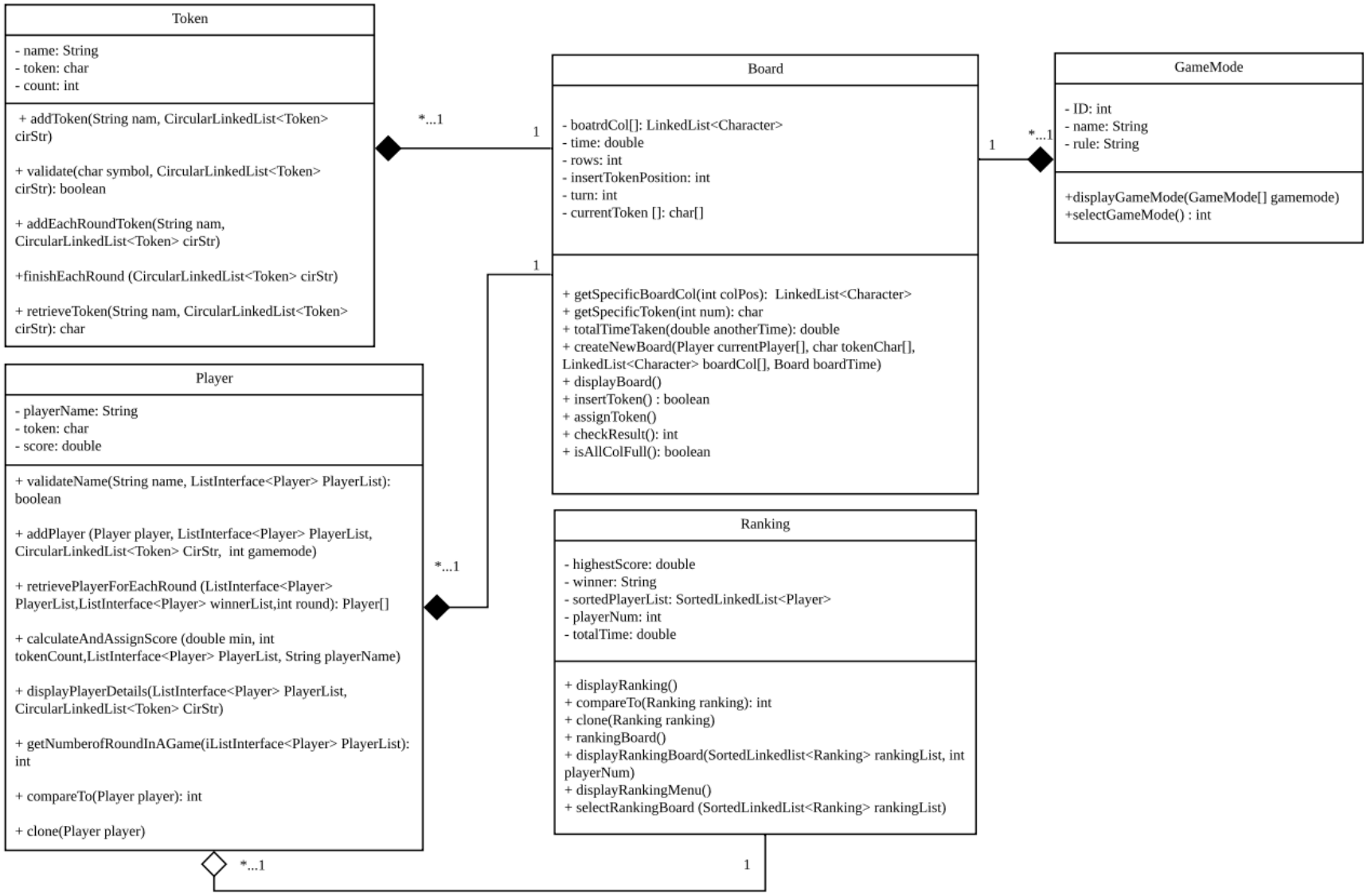
STEP 1:
A ranking board menu will de displayed for the player to choose which type of ranking board they want to view. There are 5 ranking board which are (1) 2 Player Ranking Board; (2) 3 Player Ranking Board; (3) 4 Player Ranking Board; (4) 5 Player Ranking Board; (5) More than 5 Player. The difference between each of the selection is the number of players for the game.

STEP 2:
After the player input the selection, the system will display a ranking board which include the rank, player name and score that obtained by the player.

Input '3' (Exit):

The game will be stop when the player enters three into the system in the main menu.

3. Entity Class Diagram



4. Implementation of Entity Class(es)

```
public class Player implements Comparable<Player>{

    private String name;
    private char token;
    private double score;
```

Diagram 4.1 Variables in the Player Entity Class

```
@Override
public int compareTo(Player player){
    return (int)((this.score - player.score)*100);
}

public void clone(Player player){
    this.name = player.name;
    this.score = player.score;
    this.token = player.token;
}
```

Diagram 4.2 Functions implement in Player Entity Class

5. ADT Specification

The ADT that I use in this game is List ADT. The class that implements the List ADT is ArrayList. ArrayList is a list where uses a dynamic array for storing the elements. It is like an array, but there is no size limit.

boolean add (T newEntry)
Description : Adds a new entry to the end of the list. Entries currently in the list are unaffected. The list's size is increased by 1.
Parameter : *newEntry* the object to be added as a new entry.
Postcondition : The element added with new entry.
Return : Return a true if the element added successfully.

public boolean add (int newPosition, T newEntry)
Description : Adds a new entry at a specified position within the list. Entries originally at and above the specified position are at the next higher position within the list. The list's size is increased by 1.
Parameter : *newPosition* an integer that specifies the desired position of the new entry
newEntry the object to be added as a new entry
Precondition : Check whether newPosition >= 0 and newPosition < size
Postcondition : The element added with newEntry at the specific position.
Return : Return a true if the addition is successful, or false if either the list is full, newPosition < 0, or newPosition >= size

public T getEntry (int givenPosition)
Description : Retrieves the entry at a given position in the list.
Parameter : *givenPosition* an integer that indicates the position of the desired entry
Precondition : Check whether the list is empty and check whether givenPosition >= 0 and givenPosition < size
Postcondition : The List remain unchanged
Return : Return a reference to the indicated entry or throw the IndexOutOfBoundsException, if either the list is empty, givenPosition < 0, or givenPosition >= size

public T remove (int givenPosition)
Description : Removes the entry at a given position from the list. Entries originally at positions higher than the given position are at the next lower position within the list, and the list's size is decreased by 1.
Parameter : *givenPosition* an integer that indicates the position of the entry to be removed
Precondition : Check whether newPosition >= 0 and newPosition < size
Postcondition : The element removed at the specific position.
Return : Return a reference to the indicated entry removed or throw the IndexOutOfBoundsException, if either the list is empty, givenPosition < 0, or givenPosition >= size

public void clear ()
Description : Removes the entries from the list. Reset the List size to zero.
Postcondition : The list will become empty.

public boolean replace (int givenPosition, T newEntry)
Description : Replaces the entry at a given position in the list.
Parameter : *givenPosition* an integer that indicates the position of the entry to be replaced
newEntry the object that will replace the entry at the position givenPosition
Precondition : Check whether the list not isEmpty and Check whether newPosition >= 0 and newPosition < size
Postcondition : The element will be replaced at the specific position
Return : Return a true if the replacement occurs, or false if either the list is empty, givenPosition < 0, or givenPosition >= size

public String subList (int start, int end)
Description : Retrieves the entry between a given position in the list.
Parameter : *start* an integer that represent the first data want to retrieves
end an integer that represent the last data want to retrieves
Precondition : Check whether the List not isEmpty and the start > 0 or start < size and end > 0 and end > start and end < size
Postcondition : The List remain unchanged
Return : Return the list of elements at the given position or IndexOutOfBoundsException if either the list isEmpty or the start < 0, or start >= size, or end < 0, or end >= size, or end < start

public int indexOf (T anEntry)
Description : Retrieve the position of anEntry in the List
Parameter : anEntry
Postcondition : The List remain unchanged.
Return : Return the position of the anEntry in the List

public boolean contains (T anEntry)
Description : Sees whether the list contains a given entry.
Parameter : anEntry
Postcondition : The List remain unchanged.
Return : Return a true if the element exists.

public int getLength()
Description : Gets the number of entries in the list.
Postcondition : The List remain unchanged.
Return : Return the integer number of entries currently in the list.

public boolean isEmpty()
Description : Sees whether the list is empty.
Postcondition : The List remain unchanged.
Return : Return a true if the list is empty, or false if not

public boolean isFull()
Description : Sees whether the list is full.
Postcondition : The List remain unchanged.
Return : Return a true if the list is full, or false if not

public String isArrayFull ()
Description : Sees whether the size of the list is equals to the array length
Postcondition : The List remain unchanged.
Return : Return a true if the size of the List is equal to the array length.

private void reallocate()
Description : To doubled up the size of he array
Postcondition : The List size will be multiple by two

private void removeGap (int givenPosition)
Description : To remove the gap between the element in the List
Parameter : *givenPosition* an integer that indicates the position of the desired entry
Postcondition : The gap of the element at the givenPosition will be removed

private void addGap (int givenPosition)
Description : To add the gap between the element in the List
Parameter : *givenPosition* an integer that indicates the position of the desired entry
Postcondition : The gap of the element at the givenPosition will be added

6. ADT Implementation

6.1 List Interface

```
public interface ListInterface<T> {

  /** Task: Adds a new entry to the end of the list ...7 lines */
  public boolean add(T newEntry);

  /** Task: Adds a new entry at a specified position within the list ...12 lines */
  public boolean add(int newPosition, T newEntry);

  /** Task: Removes the entry at a given position from the list ...11 lines */
  public T remove(int givenPosition);

  /** Task: Removes all entries from the list ...3 lines */
  public void clear();

  /** Task: Replaces the entry at a given position in the list ...10 lines */
  public boolean replace(int givenPosition, T newEntry);

  /** Task: Retrieves the entry at a given position in the list ...8 lines */
  public T getEntry(int givenPosition);

  /** Task: Retrieves the entry between a given position in the list ...7 lines */
  public String subList(int start, int end);

  /** Task: Sees whether the list contains a given entry ...6 lines */
  public boolean contains(T anEntry);

  /** Task: Gets the number of entries in the list ...5 lines */
  public int getLength();

  /** Task: Sees whether the list is empty ...5 lines */
  public boolean isEmpty();

  /** Task: Sees whether the list is full ...5 lines */
  public boolean isFull();

  /** Task: To get the position of the current entry ...5 lines */
  public int indexOf(T anEntry);

  /** Task: Sees whether the list meet the size assigned ...4 lines */
  public boolean isArrayFull();

}
```

6.2 ArrayList

```
public class ArrayList<T> implements ListInterface<T> {

    private T[] arr;
    private int size = 0;
    private static final int DEFAULT_CAPACITY = 10;
    private static int DEFAULT_ERROR_CODE = -1;
    private static int FACTOR  = 2;

    //Creates Empty ArrayList with Default Capacity = 10
    public ArrayList(){
      this(DEFAULT_CAPACITY);
    }

    //Creates Empty ArrayList with Specified initial Capacity
    public ArrayList(int capacity){
        size = 0;
        arr = (T[]) new Object[capacity];
    }

    @Override
    public boolean add(T newEntry){

      if(this.size == arr.length){
          reallocate();
      }

      arr[size] = newEntry;
      size++;

      return true;
    }
```

```
@Override
public boolean add(int newPosition, T newEntry){

    if(newPosition >= 0 || newPosition < arr.length){

        if(newPosition < this.size){

            if(this.size == arr.length){
                reallocate();
            }

            addGap(newPosition);

            arr[newPosition] = newEntry;
            size++;

        }

    }
    else if(newPosition == arr.length){
        add(newEntry);
    }
    else{
        return false;
    }
    return true;
}

@Override
public T getEntry(int givenPosition){

 if(!isEmpty()){
      if(givenPosition < 0 || givenPosition >= size){
       throw new IndexOutOfBoundsException();
   }
   else{
       return arr[givenPosition];
   }
 }

 throw new IndexOutOfBoundsException();
}
```

```
@Override
public T remove(int givenPosition){

    if(!isEmpty()){
        if(givenPosition < 0 || givenPosition >= size){
            throw new IndexOutOfBoundsException();
        }

     T returnEntry = arr[givenPosition];

     removeGap(givenPosition);


     size--;
     return returnEntry;
    }
    else{
        throw new IndexOutOfBoundsException();
    }



}
@Override
public int getLength(){
   return this.size;
}
@Override
public boolean replace(int givenPosition, T newEntry){

    if(!isEmpty()){
        if(givenPosition < 0 || givenPosition >= size){
            return false;
        }
        else{
            arr[givenPosition] = newEntry;
            return true;
        }
    }
    return false;

}
```

```
@Override
public int indexOf (T anEntry){

    for(int i = 0; i < this.size; i++){

        if(arr[i].equals(anEntry)){
            return i;
        }
    }

    return DEFAULT_ERROR_CODE;
}

@Override
public String subList(int start, int end){

    if(isEmpty()){
        throw new IndexOutOfBoundsException();
    }

    if(start < 0 || start >= size || end < 0 || end >= size || end < start){
        throw new IndexOutOfBoundsException();
    }
    String str = "";

    for(int i = start; i <= end; i++){

      str += arr[i] + "\n";
    }

    return str;
}

@Override
public boolean contains(T anEntry){

    if(anEntry != null){

        for(int i = 0; i < size; i++){
            if(arr[i] == anEntry){
                return true;
            }
        }
    }
    return false;
}
```

```
@Override
public boolean isEmpty() {
    return this.size == 0;
}

@Override
public boolean isFull() {
    return false;
}

@Override
public void clear(){
    this.size = 0;
}
@Override
public String toString(){
    String str = "";

    for(int index = 0; index < size; ++index){
        str += arr[index] + "\n";
    }

    return str;
}

@Override
public boolean isArrayFull(){
    return size == arr.length;
}

private void reallocate(){

   int arrLength = arr.length;

   T[] tempArr = (T[]) new Object [FACTOR*arrLength];

   for(int i = 0; i < this.size; i++){
       tempArr[i] = arr[i];
   }

   arr = tempArr;
}
```

```
    private void removeGap(int givenPosition){

        for(int i = givenPosition + 1; i < this.size; i++){
            arr[i - 1] = arr[i];
        }
    }

    private void addGap(int newPosition){
         for(int i = this.size; i >= newPosition; i--){
            arr[i + 1] = arr[i];
         }
    }

}
```

7. Selection of Collection ADTs

The entity that I have implemented is Player. The ADT that I use for the player entity is List ADT. The class that implements the List ADT is ArrayList. The way I use the ArrayList that I implement in the client program is when the player enter the player name since our game is allow the player to enter the number of player of the game process, therefore no matter how many number of player that the user want to play for each of the round in the game, the player details will be store by using the ArrayList for further references. Before storing the player name, the validation function will be taken place which will compare the current player name that had been store in the player list. Then, the player details will be display on the screen in order to inform the player before the game started. Before the process of the game, I will retrieve the first two player in the list using the method in the ADT. I will use my ADT to get the current players in order to assigned who will be the next player to play the game. Besides, we will also display the player name at each time of display to inform whose turn. After the first round of the game, the game will be having a winner for the first round of the game and I will calculate and store the score of the player in the player list by comparing the name of the player. The winner will also be store in the player entity by using the ArrayList. Then, the game will continue played by the next player and will also call again my retrieve player function to assigned current player. The reason I choose the ArrayList for my player entity is because I will need to keep repeating calling the player in the player list therefore, the ArrayList will be easier for me to assigned and retrieve the player into the list and also the ArrayList is very useful for the program to repeatedly retrieve the element at the specific position without go through the overall element in the list.

8. Implementation of Client Program

8.1 Add Player Function

```
public void addPlayer(Player player, ListInterface<Player> PlayerList, CircularLinkedList<Token> CirStr,int gamemode){

    int numOfPlayer = 0;
    int check = 0;
    String playerName = "";

    Scanner scan = new Scanner(System.in);
    System.out.printf("         ==========================\n");
    System.out.printf("             Player Registration\n");
    System.out.printf("         ==========================\n");


    if(gamemode==1){
    do{

        System.out.printf("Please Enter The Number Of Player For This Round:    ");

       try{
           numOfPlayer = scan.nextInt();

           if(numOfPlayer < 2){
               System.out.println("Invalid Input Detected!");
           }
       }
       catch (InputMismatchException ex){
           System.out.println("Input Must be Positive Integer");
           scan.next();
       }

    }while(numOfPlayer < 2);
```

Diagram 8.1.1 Add Player Function (1)

```
}
else {

    do{

    System.out.printf("Please Enter The Number Of Player For Tournament (Only 4/8 player allowed) :    ");

      try{
       numOfPlayer = scan.nextInt();

       if(numOfPlayer != 4 && numOfPlayer != 8){
           System.out.println("Input Must be 4 or 8");
       }
      }
      catch (InputMismatchException ex){
       System.out.println("Invalid Input Detected!");
       scan.next();
   }


    }while(numOfPlayer != 4 && numOfPlayer != 8);
}

do{
    System.out.printf("\nPlease Enter The Player Name:    \n");

    scan.nextLine();

    for(int i = 0; i < numOfPlayer; i++){

        System.out.printf("\nPlayer%2d Name:    ", i+1);

        playerName = scan.nextLine();
```

Diagram 8.1.2 Add Player Function (2)

```
            if(validateName(playerName,PlayerList)){


                player = new Player(playerName);

                PlayerList.add(player);

                TokenCount tok = new TokenCount();
                tok.addToken(playerName, CirStr);

                for(int j = 0; j < PlayerList.getLength(); j++){

                    if(PlayerList.getEntry(j).equals(player)){
                        PlayerList.getEntry(j).setToken(CirStr.getEntry(j + 1).getToken());
                    }
                }

            }
            else if (-1 == check || !validateName(playerName,PlayerList)){
                i--;
                System.out.println("\nError Detected! Please Enter Again");
            }


        }
    }while("".equals(playerName));

}
```

Diagram 8.1.3 Add Player Function (3)

```
play.addPlayer(playerN, PlayerList, CirStr,gamemode);

numOfPlayer = PlayerList.getLength();
```

Diagram 8.1.4 Function call to add the player in the main and get the number of players for the game

8.2 Retrieve Player for Each Round Function

```
public Player[] retrievePlayerForEachRound(ListInterface<Player> PlayerList,ListInterface<Player> winnerList,int round){

    Player[] player = new Player[2];
    int length = winnerList.getLength();




    if(length == 0 && round == 1){

        player[0] = PlayerList.getEntry(0);
        player[1] = PlayerList.getEntry(1);

    }
    else if (length > 0 && round > 1){

        player[0] = winnerList.getEntry(length - 1);
        player[1] = PlayerList.getEntry(length + 1);

    }

    return player;
}
```

Diagram 8.2.1 Retrieve Player for each round Function

```
do{
    if(gamemode==1)
currentPlayer = play.retrievePlayerForEachRound(PlayerList, winnerList,round);
    else currentPlayer=tour.retrievePlayerForEachRound(numOfPlayer);
currentToken[0] = token.retrieveToken(currentPlayer[0], CirStr);
currentToken[1] = token.retrieveToken(currentPlayer[1], CirStr);

System.out.println(currentPlayer[0] + " >> "+ currentPlayer[1]);
Board mainBoard = new Board(currentPlayer, currentToken);
mainBoard.setBoardCol(boardCol);
mainBoard.createNewBoard(rows);
mainBoard.setTime();
startTime = mainBoard.getTime();
```

Diagram 8.2.2 Function call to retrieve the current player of each round in the main

8.3 Calculate and Assign Score Function

```
public void calculateAndAssignScore(double min, int tokenCount,ListInterface<Player> PlayerList, String playerName){

    int length = PlayerList.getLength();
    double playerScore;

    for(int i = 0; i < length; i++){

        if(PlayerList.getEntry(i).getName().equals(playerName)){
            playerScore = PlayerList.getEntry(i).getScore();
            double score = (double) ((tokenCount/min)*100);

             if(playerScore == 0){
                    PlayerList.getEntry(i).setScore(score);
             }else{
                    score += PlayerList.getEntry(i).getScore();
                    PlayerList.getEntry(i).setScore(score);
             }
        }
    }

}
```

Diagram 8.3.1 Calculate score and store into the player list

```
if(checkResult != 0){
    if(checkResult %2 == 0){
        winnerName = currentPlayer[0].getName();
        Player playerW = new Player();
        board.setTime();
        timeTaken = board.totalTimeTaken(startTime);
        totalTime += timeTaken;
        playerW.setName(currentPlayer[0].getName());
        if(gamemode==1){
        play.calculateAndAssignScore(timeTaken,CirStr.getEntry(1).getCount(),PlayerList,currentPlayer[0].getName());
        winnerList.add(playerW);
        }else tour.stepUp(playerW,numOfPlayer);
        round++;

    }else{
        winnerName = currentPlayer[1].getName();
        Player playerW = new Player();
        board.setTime();
        timeTaken = board.totalTimeTaken(startTime);
        totalTime += timeTaken;
        playerW.setName(currentPlayer[1].getName());
        if(gamemode==1){
        play.calculateAndAssignScore(timeTaken,CirStr.getEntry(2).getCount(),PlayerList,currentPlayer[1].getName());
        winnerList.add(playerW);
        }
        else tour.stepUp(playerW,numOfPlayer);
        round++;
    }
```

Diagram 8.3.2 Assigned the winner into the player list and calculate and store the score for the particular player

8.4 Display Player Details Function

```
public void displayPlayerDetails(ListInterface<Player> PlayerList, CircularLinkedList<Token> CirStr){

    System.out.printf("\t\t\t============================\n");
    System.out.printf("\t\t\t     Player Details\n");
    System.out.printf("\t\t\t============================\n");

    System.out.printf("=================================================================\n");
    System.out.printf("||\t Player\t\t\tPlayer Name\t\t\tToken\t||\n");
    for(int i = 0; i < PlayerList.getLength(); i++){

         System.out.println(String.format("||%12d %-18s %-31s %c\t||",
                (i+1),
                " ",
                PlayerList.getEntry(i).getName(),
                PlayerList.getEntry(i).getToken()));
    }
    System.out.printf("=================================================================\n");
}
```

Diagram 8.4.1 Display Player Details

```
play.displayPlayerDetails(PlayerList, CirStr);
if(gamemode==2)
tour.display();
```

Diagram 8.4.2 Function call to display the player details in main class

8.5 Get Number of Round in a Game

```
public int getNumberofRoundInAGame(ListInterface<Player> PlayerList){

    int round = PlayerList.getLength() - 1;

    return round;
}
```

Diagram 8.5.1 Function to get number of rounds in a game

```
            play.calculateAndAssignScore(timeTaken,CirStr.getEntry(2).getCount(),PlayerList,currentPlayer[1]);
            winnerList.add(playerW);
            }
            else tour.stepUp(playerW,numOfPlayer);
            round++;
        }

        mainBoard.displayBoard();
        token.finishEachRound(CirStr);
        System.out.println("\nWinner is "+ winner.toUpperCase());

        if(gamemode==2)
        tour.display();

        break;
    }


}while(insertSuccess);


}while(round <= play.getNumberofRoundInAGame(numOfPlayer));
```

Diagram 8.5.2 Retrieve the number of rounds in a game in main class

8.6 Validate Name Function

```
public  boolean validateName(String name, ListInterface<Player> PlayerList) {



    String currentName = name ;
    String regexName = "^[aA-zZ]\\w{0,29}$";
    int check = 0;

    for(int i = 0; i < PlayerList.getLength(); i++){
        if(PlayerList.getEntry(i).getName().equals(name)){
            check = -1;
        }
    }
    return !(name.length() < 0 || !currentName.matches(regexName) || (check == -1)) ;
}
```

Diagram 8.6.1 Function validate player name

```
for(int i = 0; i < numOfPlayer; i++){

    System.out.printf("\nPlayer%2d Name:    ", i+1);

    playerName = scan.nextLine();

    if(validateName(playerName,PlayerList)){


        player = new Player(playerName);

        PlayerList.add(player);

        TokenCount tok = new TokenCount();
        tok.addToken(playerName, CirStr);

        for(int j = 0; j < PlayerList.getLength(); j++){

            if(PlayerList.getEntry(j).equals(player)){
                PlayerList.getEntry(j).setToken(CirStr.getEntry(j + 1).getToken());
            }
        }

    }
    else if (-1 == check || !validateName(playerName,PlayerList)){
        i--;
        System.out.println("\nError Detected! Please Enter Again");
    }


}
```

Diagram 8.6.2 Function call to validate player name

9. Connect Four Game Source Code

Available At: https://github.com/yongkang0827/Connect4