No	Input	Expected output	Pass
0	Running the program	Showing instructions	Yes
ADDING CLUB			
1	Entering 1 in main menu to show instructions about adding member options	Showing instructions about adding members	Yes
2	Entering 1 in sub menu to add football club to the system	Asking for input to create club object. After adding showing how many clubs manger can add. Showing added clubs' details.	Yes
3	Adding already added club again	Showing "Invalid, already a club exist with similar details"	Yes
4	Adding more than 20 club	Showing "Invalid, Maximum number of club limit already passed!"	Yes
5	Entering 2 in submenu to go back to main menu	Showing main menu	Yes

DELETING CLUB			
6	Entering 2 in main menu to show instructions about deleting options	Showing instruction about deleting existing member	Yes
7	Entering 1 in sub menu to continue to delete club from the system	Showing Clubs with numbers to choose a club by a number.	Yes
8	Entering number to select specific delete a specific club	Selected club deleted	Yes
9	Entering 2 in sub menu to go back to main menu	Showing main menu	Yes
DISPLAY CLUB STATISTICS			
10	Entering 3 in main menu to show instructions about displaying club statistics.	Showing instructions about displaying statistics.	Yes
11	Entering 1 in sub menu to continue to display statistics	Printing the number and clubs' names to choose one	Yes

12	Entering number to select a club to view its statistics	Showing selected members details.	Yes
13	Entering 2 in sub menu to go back to main menu	Showing main menu	Yes

DISPLAY CLUBS TABLE VIEW			
14	Entering 4 in main menu to show instructions about sorted club statistics table view	Asking to continue or go back.	Yes
15	Entering 1 in sub menu to display statistics table view	Showing table view	Yes
16	Entering 2 in submenu to go back to main menu	Showing main menu	Yes
UPDATE MEMBERS			
17	Entering 5 in main menu to update club	Showing instruction and asking whether to continue or go back	Yes

18	Entering 1 in sub menu to continue	Asking for input to create match. After creating match, automatically updating club details	Yes

19	Entering 2 in sub menu to go back to main menu	Showing main menu	Yes
GUI			
20	Entering 6 in main menu to show instruction to run	Showing Instruction.	Yes
21	Using search field to search club's details based on their date.	Showing matches happened at specific date.	Yes
22	Using Button to show sorted order of clubs. (Sort by difference between goal scored and goal received, points and wins)	Showing sorted clubs' details in the gui	Yes
23	Using Button to show sorted order of matches and creating random matches. (Sort by date, random match)	Showing sorted way and creating random matches successfully	Yes
QUIT			
24	Entering 7 in main menu to stop the program	Showing "Program quitting" .	Yes

Validation			
25	User entering in lower case	Working as always.	Yes
26	User entering invalid inputs	Showing invalid message. And asking for correct input again.	Yes

DateTest.java

```
public class DateTest {

   @Test
   public void getFullDate() {
      Date date = new Date(12,11,2021);
      assertEquals("12/11/2021", date.getFullDate());
   }

   @Test
   public void setFullDate() {
      Date date = new Date(12,11,2021);
      date.setFullDate("12/11/2021");
      assertEquals("12/11/2021", date.getFullDate());
   }

   @Test
   public void getMonth() {
      Date date = new Date(12,11,2021);
      assertEquals(11, date.getMonth());
   }

   @Test
   public void setMonth() {
      Date date = new Date(12,11,2021);
      date.setMonth(12);
      assertEquals(12, date.getMonth());
   }

   @Test
   public void getYear() {
      Date date = new Date(12,11,2021);
      assertEquals(2021, date.getYear());
   }

   @Test
   public void setYear() {
      Date date = new Date(12,11,2021);
      date.setYear(2020);
      assertEquals(2020, date.getYear());
   }
```

```
    @Test
    public void getDay() {
        Date date = new Date(12,11,2021);
        assertEquals(12, date.getDay());
    }

    @Test
    public void setDay() {
        Date date = new Date(12,11,2021);
        date.setDay(11);
        assertEquals(11, date.getDay());
    }

    @Test
    public void isYearCorrect() {
        assertTrue(Date.isDayCorrect(2020, 12, 13));
    }

    @Test
    public void isMonthCorrect() {
        assertTrue(Date.isDayCorrect(2020, 12, 13));
    }

    @Test
    public void isDayCorrect() {
        assertTrue(Date.isDayCorrect(2020, 12, 13));
    }
}
```

FootballclubTest.java

```
public class FootballClubTest {

    @Test
    public void getClubName() {
        FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
        assertEquals("Chelsea", f1.getClubName());
    }

    @Test
    public void getLocationName() {
        FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
        assertEquals("Chelsea", f1.getLocationName());
    }

    @Test
    public void getFoundedYear() {
        FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
        assertEquals(1905, f1.getFoundedYear());
    }

    @Test
    public void setClubName() {
        FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
        f1.setClubName("Burnley");
        assertEquals("Burnley", f1.getClubName());
    }

    @Test
    public void setLocationName() {
        FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
        f1.setLocationName("Burnley");
```

```
    assertEquals("Burnley", f1.getLocationName());
}

@Test
public void setFoundedYear() {
    FootballClub f1 = new FootballClub("Chelsea", "Chelsea", 1905);
    f1.setFoundedYear(1882);
    assertEquals(1882, f1.getFoundedYear());
}

@Test
public void testEquals() {
    FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
    FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
    assertEquals(f1, f2);
}

@Test
public void getNumberOfWin() {
    FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
    f1.setNumberOfWin(2);
    assertEquals(2, f1.getNumberOfWin());
}

@Test
public void setNumberOfWin() {
    FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
    f1.setNumberOfWin(3);
    assertEquals(3, f1.getNumberOfWin());
}

@Test
public void getNumberOfDraw() {
    FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
    f1.setNumberOfDraw(3);
    assertEquals(3, f1.getNumberOfDraw());
}

@Test
public void setNumberOfDraw() {
    FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
    f1.setNumberOfDraw(3);
    assertEquals(3, f1.getNumberOfDraw());
}
```

```
@Test
public void getNumberOfDefeat() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfDefeat(3);
   assertEquals(3, f1.getNumberOfDefeat());
}

@Test
public void setNumberOfDefeat() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfDefeat(3);
   assertEquals(3, f1.getNumberOfDefeat());
}

@Test
public void getNumberOfGoalReceived() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfGoalReceived(3);
   assertEquals(3, f1.getNumberOfGoalReceived());
}

@Test
public void setNumberOfGoalReceived() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfGoalReceived(3);
   assertEquals(3, f1.getNumberOfGoalReceived());
}

@Test
public void getNumberOfGoalScored() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfGoalScored(3);
   assertEquals(3, f1.getNumberOfGoalScored());
}

@Test
public void setNumberOfGoalScored() {
   FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
   f1.setNumberOfGoalScored(3);
   assertEquals(3, f1.getNumberOfGoalScored());
}

@Test
```

```
    public void getNumberOfPoint() {
        FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
        f1.setNumberOfPoint(3);
        assertEquals(3, f1.getNumberOfPoint());
    }

    @Test
    public void setNumberOfPoint() {
        FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
        f1.setNumberOfPoint(3);
        assertEquals(3, f1.getNumberOfPoint());
    }

    @Test
    public void getNumberOfPlayedMatch() {
        FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
        f1.setNumberOfPlayedMatch(3);
        assertEquals(3, f1.getNumberOfPlayedMatch());
    }

    @Test
    public void setNumberOfPlayedMatch() {
        FootballClub f1 = new FootballClub("Burnley", "Burnley", 1882);
        f1.setNumberOfPlayedMatch(3);
        assertEquals(3, f1.getNumberOfPlayedMatch());
    }

}
```

MatchTest.java

```
public class MatchTest {

    @Test
    public void getClub1() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        assertEquals(f1, match.getClub1());
    }

    @Test
    public void setClub1() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        match.setClub2(f3);
        assertEquals(f3, match.getClub2());
    }

    @Test
    public void getClub2() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        assertEquals(f2, match.getClub2());
    }

    @Test
    public void setClub2() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
```

```
    FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
    Date d = new Date(12, 11, 2021);
    Match match = new Match(f1, f2, 12, 13, d);
    match.setClub2(f3);
    assertEquals(f3, match.getClub2());

}

@Test
public void getClub1Score() {
    FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
    FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
    FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
    Date d = new Date(12, 11, 2021);
    Match match = new Match(f1, f2, 12, 13, d);
    assertEquals(12, match.getClub1Score());

}

@Test
public void setClub1Score() {
    FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
    FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
    FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
    Date d = new Date(12, 11, 2021);
    Match match = new Match(f1, f2, 12, 13, d);
    match.setClub1Score(14);
    assertEquals(14, match.getClub1Score());

}

@Test
public void getClub2Score() {
    FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
    FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
    FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
    Date d = new Date(12, 11, 2021);
    Match match = new Match(f1, f2, 12, 13, d);
    match.setClub2Score(12);
    assertEquals(12, match.getClub2Score());

}

@Test
```

```
public void setClub2Score() {

   FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
   FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
   FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
   Date d = new Date(12, 11, 2021);
   Match match = new Match(f1, f2, 12, 13, d);
   match.setClub2Score(12);
   assertEquals(12, match.getClub2Score());

}

@Test
public void getMatchDate() {

   FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
   FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
   FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
   Date d = new Date(12, 11, 2021);
   Match match = new Match(f1, f2, 12, 13, d);
   assertEquals(d.getFullDate(), match.getMatchDate());
}

@Test
public void getMatchWinner() {

   FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
   FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
   FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
   Date d = new Date(12, 11, 2021);
   Match match = new Match(f1, f2, 12, 13, d);
   assertEquals(f2, match.getMatchWinner());

}

@Test
public void getMatchLooser() {

   FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
   FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
   FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
   Date d = new Date(12, 11, 2021);
   Match match = new Match(f1, f2, 12, 13, d);
   assertEquals(f1, match.getMatchLooser());
```

```
    }

    @Test
    public void isMatchDraw() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        assertFalse(match.isMatchDraw());

    }

    @Test
    public void getWinnerClubScore() {
        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        assertEquals(13, match.getWinnerClubScore());

    }

    @Test
    public void getLooserClubScore() {

        FootballClub f1 = new FootballClub("Aston Villa", "Aston Villa", 1886);
        FootballClub f2 = new FootballClub("Burnley", "Burnley", 1882);
        FootballClub f3 = new FootballClub("Burn", "Burn", 1982);
        Date d = new Date(12, 11, 2021);
        Match match = new Match(f1, f2, 12, 13, d);
        assertEquals(12, match.getLooserClubScore());

    }
}
```


Run:
MatchTest
Tests passed: 14 of 14 tests – 46 ms
MatchTest (entities) 46 ms
getLooserClubScore 39 ms
getClub1Score 1 ms
getClub2Score 0 ms
getWinnerClubScore 1 ms
getMatchDate 0 ms
getMatchLooser 1 ms
setClub1Score 1 ms
setClub2Score 0 ms
getMatchWinner 0 ms
setClub1 1 ms
setClub2 0 ms
isMatchDraw 1 ms
getClub1 0 ms
1 ms
/Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java ...
Process finished with exit code 0
Tests passed: 14

ConsoleApplication.java

```
public class ConsoleApplication{


   // Calling the default constructor to create object for premier league manager
   // So we can call PremierLeagueManger methods inside static method

   // Programming to the super type not to the subtype / Programming to interface not to the implementation
(Good behaviour)
   // We can only methods which are in the supertype / interface
   static LeagueManager manager = new PremierLeagueManager();
   static ConsoleService consoleService = new ConsoleService();

   public static void main(String [] args) throws IOException, ClassNotFoundException {

//================================================ Welcome message And Displaying options
================================================================================================
============================================================

      // Welcome message will be displayed when program starts
      System.out.println("\n\t\t\t\t -- Welcome to Premier League --");
      System.out.println("\t\t\t\t  -- Command Line Interface --\n");

      // Looping until user enters specific number to exit out of the loop (Number : 7)
      boolean run = true;
      while(run) { // Loop until run variable become false
         int selectedOption;
         int subSelectedOption;

         // Instructions to call the specific functions
         System.out.println("-----------------------------------------------------------------------------------------");
         System.out.println("Level : Activating functionalities\n");
         System.out.println("Type the number to activate specific functionality");
         System.out.println("\t(1) Add new football club");
         System.out.println("\t(2) Delete existing club");
         System.out.println("\t(3) Display statistics for specific club");
         System.out.println("\t(4) Display premier league table");
         System.out.println("\t(5) Update club");
         System.out.println("\t(6) GUI");
         System.out.println("\t(7) Exit");
         System.out.println("-----------------------------------------------------------------------------------------");
```

```
        /*
         * limitedIntegerValidation is a hardcoded method which will get two String parameters (question, invalid message) and two integers (minimum and maximum values).
         * And ask for the user input and checking whether it's correct or not (Only the inputs which are between min and max will be returned)).
         * If it's correct return the user entered input to the variable.
         * If not method will ask for the input again and again until user enters correct info.
         * Inside the method there is a do while loop to loop until user enters expected input.
         */
        selectedOption = Validation.limitedIntegerValidation("What function do you want to activate? (1-7)","Invalid, Unavailable option!",1,7);

        switch (selectedOption){
           case 1:

              // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different integer user don't need to rerun
              System.out.println("-----------------------------------------------------------------------------------------");
              System.out.println("Type the number to activate specific functionality");
              System.out.println("\t(1) Add club");
              System.out.println("\t(2) Back");
              System.out.println("-----------------------------------------------------------------------------------------");

              subSelectedOption = Validation.limitedIntegerValidation("What do you want to do? (Continue/Back)","Invalid, Unavailable option!",1,2);
              if (subSelectedOption==1){

                 /*
                  * addClub() method is used to add the created club into array list
                  * This method will get FootballClub objects as parameter
                  * getInputAndCreateClub() method will ask for user input and create a club and return it
                  */
                 manager.addClub(getInputAndCreateClub());
              }
              break;


           case 2:

              // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different integer user don't need to rerun
              System.out.println("-----------------------------------------------------------------------------------------");
              System.out.println("Type the number to activate specific functionality");
              System.out.println("\t(1) Delete club");
```

```
            System.out.println("\t(2) Back");
            System.out.println("-----------------------------------------------------------------------------------------");

            subSelectedOption = Validation.limitedIntegerValidation("What do you want to do?
(Continue/Back)","Invalid, Unavailable option!",1,2);
            if (subSelectedOption==1){
               /*
                * deleteExistingClub() method is used to delete the existing club from the list
                * This method will get integer (index number of the club index) as parameter to delete the specific
club
                */

               if (consoleService.getNoOfClubs()>0){
                  manager.deleteExistingClub(getIndexAndReturnClub("What's the club index number?","Invalid,
Unavailable option!",1,consoleService.getNoOfClubs()), getFinalConfirmation());
               }else{
                  System.out.println("Invalid, Not any clubs have been saved to the system to remove!");
               }
            }
            break;

         case 3:
            // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different
integer user don't need to rerun
            System.out.println("-----------------------------------------------------------------------------------------");
            System.out.println("Type the number to activate specific functionality");
            System.out.println("\t(1) Display statistics for specific club");
            System.out.println("\t(2) Back");
            System.out.println("-----------------------------------------------------------------------------------------");

            subSelectedOption = Validation.limitedIntegerValidation("What do you want to do?
(Continue/Back)","Invalid, Unavailable option!",1,2);
            if (subSelectedOption==1){
               /*
                * displayStatistics() method is used to display the statistics of existing club from the list
                * This method will get integer (index number of the club) as parameter to display that specific club
statistics
                */
               if (consoleService.getNoOfClubs()>0){
                  manager.displayStatistics(getIndexAndReturnClub("What's the club index number?","Invalid,
Unavailable option!",1,consoleService.getNoOfClubs()));
               }else{
                  System.out.println("Invalid, Not any clubs have been saved to the system to display!");
               }
```

```
            }
            break;


        case 4:
            // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different integer user don't need to rerun
            System.out.println("-----------------------------------------------------------------------------------------");
            System.out.println("Type the number to activate specific functionality");
            System.out.println("\t(1) Display statistics in table view");
            System.out.println("\t(2) Back");
            System.out.println("-----------------------------------------------------------------------------------------");

            subSelectedOption = Validation.limitedIntegerValidation("What do you want to do? (Continue/Back)","Invalid, Unavailable option!",1,2);
            if (subSelectedOption==1){
                // displayPremierLeagueTable() method is used to print premier league club's statistics in table view in console.
                if (consoleService.getNoOfClubs()>0){
                    manager.displayPremierLeagueTable();
                }else{
                    System.out.println("Invalid, Not any clubs have been saved to the system to display!");
                }
            }
            break;

        case 5:
            // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different integer user don't need to rerun
            System.out.println("-----------------------------------------------------------------------------------------");
            System.out.println("Type the number to activate specific functionality");
            System.out.println("\t(1) Update club");
            System.out.println("\t(2) Back");
            System.out.println("-----------------------------------------------------------------------------------------");

            subSelectedOption = Validation.limitedIntegerValidation("What do you want to do? (Continue/Back)","Invalid, Unavailable option!",1,2);
            if (subSelectedOption==1){
                /*
                 * updateClub() method is used to update club details
                 * This method will get Match object as parameter
                 * Then updateClub() method will update the clubs statistics which were participated in the match
                 * getInputAndReturnMatch() method will get input from user and create club and return it
```

```
                 */
                if (consoleService.getNoOfClubs()>0){
                   manager.updateClubs(getInputAndReturnMatch());
                }else{
                   System.out.println("Invalid, Not any clubs have been saved to the system to update!");
                }
             }
             break;

          case 6:
             // User can go back to the main menu by inserting 2. So, if a user accidentally inserted an different integer user don't need to rerun
             System.out.println("-----------------------------------------------------------------------------------------");
             System.out.println("Type the number to activate specific functionality");
             System.out.println("\t(1) Display gui");
             System.out.println("\t(2) Back");
             System.out.println("-----------------------------------------------------------------------------------------");

             subSelectedOption = Validation.limitedIntegerValidation("What do you want to do? (Continue/Back)","Invalid, Unavailable option!",1,2);

             if (subSelectedOption==1){
//                Runtime.getRuntime().exec("sbt run");

                System.out.println("Please follow the bellow steps to run gui");
                System.out.println("\t(1) Open cmd and type change the directory using \"cd ui\"");
                System.out.println("\t(2) Now type \"npm install\" and click enter");
                System.out.println("\t(3) Open another cmd or terminal and type \"sbt run\"");
                System.out.println("\t(4) Then click enter");
                System.out.println("\t(5) Now open another new cmd or terminal");
                System.out.println("\t(6) In the new terminal or cmd type \"ng serve --open\"");
                System.out.println("\t(7) Then click enter");
                System.out.println("\t(8) Now wait for a minute gui will open in your brownser");
                System.out.println();
                System.out.println("* In the above steps when opening the cmd or terminal always set the directory to project folder");
             }
             break;

          case 7:
             run = false;
             System.out.println("Thank you for using this program!");
             break;
```

```
            default:
                // If some bug let user to input other integer that are not in limit (min,max), program will throw and
error and stop
                throw new IllegalArgumentException("Unexpected error in calling the functionalities!");
            }
        }
    }


//============================================================== Getting Input
=============================================================================================
=====================================================================

    // getInputAndCreateClub() method is used to get inputs from user and create a FootballClub object with user
inputs and returning the FootballClub object
    public static FootballClub getInputAndCreateClub(){

        // Usually clubName and clubLocation will have a-z values and space characters
        // So if a user inserts any integers or other characters for clubName and clubLocation stringValidation() method
will ask for the input again until user enters valid input for clubName and clubCharacters
        String clubName = Validation.stringValidation("What's the club name?","Invalid, Club name should contain
only alphabets!");
        String clubLocation = Validation.stringValidation("What's the club location?","Invalid, Club name should
contain only alphabets!");

        // Only integer inputs that are between 1850 and 2020 will get accept
        // If user inserts any other integers that are not between 1850 and 2020, then limitedIntegerValidation()
method will ask the user to reenter until user enters valid input
        // Because user can't enter invalid year values
        int foundedYear = Validation.limitedIntegerValidation("What's the club founded year?","Invalid, Year can't be
less than 1750 or greater than 2021!",1750,2021);

        return new FootballClub(clubName,clubLocation,foundedYear);
    }

    // getInputAndReturnMatch() method is used to get inputs from user and create Match object with user inputs
and returning the match object
    public static Match getInputAndReturnMatch() throws IOException, ClassNotFoundException {
        Scanner input = new Scanner(System.in);

        FootballClub club1;
        FootballClub club2;

        do {
```

```
        club1 = getIndexAndReturnClub("What's the club-1 index number?","Invalid, Unavailable index number!",1,consoleService.getNoOfClubs());
        club2 = getIndexAndReturnClub("What's the club-2 index number?","Invalid, Unavailable index number!",1,consoleService.getNoOfClubs());

        if (club1.equals(club2)) System.out.println("Invalid, Both clubs can't be similar in a match!");

    }while (club1.equals(club2));

    // Getting club scores from users and assigning to the variables
    // Goals can be between 0 and 15 because goal record in a match is 13
    int club1Score = Validation.limitedIntegerValidation("What's the club-1 score?","Invalid, Club score can't be less than 0 or greater than 20!",0,20);
    int club2Score = Validation.limitedIntegerValidation("What's the club-2 score?","Invalid, Club score can't be less than 0 or greater than 20!",0,20);

    // Calling Date class method (getDateFromUserInput())
    // This method is used to get string input for date of the match
    Date dateOfMatch = Validation.dateInputValidation("What's the date of the match?","Invalid, Match date can't be past!");

    return new Match(club1,club2,club1Score,club2Score,dateOfMatch);
  }


  public static FootballClub getIndexAndReturnClub(String question, String invalidMsg, int min, int max) throws IOException, ClassNotFoundException {
    // manager.displayClubNameIndex() method will print club name and their index
    consoleService.displayClubNameWithIndex();

    // manger.getClubNameForIndex() method will get user input
    return consoleService.getClubForIndex(Validation.limitedIntegerValidation(question, invalidMsg, min, max));
  }

  public static boolean getFinalConfirmation(){
    return Validation.booleanConfirmation("Do you want to delete the club? (Yes/No)","Invalid, Unavailable option!");
  }



}
```

LeagueController.java

```
public class LeagueController extends Controller {

   static LeagueService leagueService = new LeagueService();

   public Result index(Http.Request request){
      return ok("You made a "+request+"!");
   }

   public Result clubListSortedByPointsAndGoalScored() throws IOException, ClassNotFoundException {
      ArrayList<FootballClub> clubs = leagueService.getClubsListSortedByPoints();
      ObjectMapper mapper = new ObjectMapper();
      JsonNode jsonClubObjects = mapper.convertValue(clubs,JsonNode.class);
      return ok((jsonClubObjects));
   }

   public Result clubListSortedByNoOfWins() throws IOException, ClassNotFoundException {
      ArrayList<FootballClub> clubs = leagueService.getClubsListSortedByWins();
      ObjectMapper mapper = new ObjectMapper();
      JsonNode jsonClubObjects = mapper.convertValue(clubs,JsonNode.class);
      return ok(jsonClubObjects);
   }

   public Result clubListSortedByNoOfGoalScored() throws IOException, ClassNotFoundException {
      ArrayList<FootballClub> clubs = leagueService.getClubsListSortedByGoalScored();
      ObjectMapper mapper = new ObjectMapper();
      JsonNode jsonClubObjects = mapper.convertValue(clubs,JsonNode.class);
      return ok(jsonClubObjects);
   }

   public Result createRandomMatch() throws IOException, ClassNotFoundException {
      ArrayList<Match> matches = leagueService.createRandomMatches();
      ObjectMapper mapper = new ObjectMapper();
      JsonNode jsonMatchObjects = mapper.convertValue(matches,JsonNode.class);
      return ok(jsonMatchObjects);
   }

   public Result matchList() throws IOException, ClassNotFoundException {
      ArrayList<Match> matches = leagueService.getMatchesList();
      ObjectMapper mapper = new ObjectMapper();
      JsonNode jsonMatchObjects = mapper.convertValue(matches,JsonNode.class);
      return ok(jsonMatchObjects);
```

```
    }

    public Result matchListSortedByDate() throws IOException, ClassNotFoundException {
        ArrayList<Match> matches = leagueService.getMatchesListSortedByDate();
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonMatchObjects = mapper.convertValue(matches,JsonNode.class);
        return ok(jsonMatchObjects);
    }

    public Result matchesInSpecificDate(String fullDate) throws IOException, ClassNotFoundException {
        ArrayList<Match> matches = leagueService.getSpecificMatches(fullDate);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonMatchObjects = mapper.convertValue(matches,JsonNode.class);
        return ok(jsonMatchObjects);
    }
}
```

Date.java

```
package entities;

public class Date implements Serializable {
    private static final int END_YEAR = 2022;
    private static final int START_YEAR = 2020;

    private String fullDate;
    private int day;
    private int month;
    private int year;

   public Date(int day, int month, int year){
        super();
        this.setMonth(month);
        this.setYear(year);
        this.setDay(day);

this.setFullDate((getDay()<10?("0"+getDay()):getDay())+"/"+(getMonth()<10?("0"+getMonth()):getMonth())+"/"+ge
tYear());
    }
```

```
public String getFullDate() {
   return fullDate;
}

public void setFullDate(String fullDate) {
   this.fullDate = fullDate;
}

public int getMonth() {
   return month;
}

public void setMonth(int month) {
   if (isMonthCorrect(month)){
      this.month = month;
   }else{
      throw new IllegalArgumentException("Invalid, Month is incorrect");
   }
}

public int getYear() {
   return year;
}

public void setYear(int year) {
   if (isYearCorrect(year)){
      this.year = year;
   }else {
      throw new IllegalArgumentException("Invalid, Year is incorrect");
   }
}

public int getDay() {
   return day;
}

public void setDay(int day) {
   if (isDayCorrect(getYear(),getMonth(),day)){
      this.day = day;
   }else throw new IllegalArgumentException("Invalid, Date is incorrect");
}
```

```
    public static boolean isYearCorrect(int year) {
        return year >= START_YEAR && year <= END_YEAR;
    }


    public static boolean isMonthCorrect(int month) {
        return month >= 1 && month <= 12;
    }

and year is correct, method will return true
    public static boolean isDayCorrect(int year, int month, int day) {
        if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) {
            return day >= 1 && day <= 31;
        }
        else if (month == 4 || month == 6 || month == 9 || month == 11) {
            return day >= 1 && day <= 30;
        }
        else if(month == 2){
            if (year % 4 == 0) {
                return day >= 1 && day <= 29;
            } else {
                return day >= 1 && day <= 28;
            }
        }
        return false;
    }

}
```

Match.java

```
public class Match implements Serializable, Comparable<Match>{
   private FootballClub club1;
   private FootballClub club2;
   private int club1Score;
   private int club2Score;
   private final Date matchDateObj;
   private final String matchDate;

   public Match(FootballClub club1, FootballClub club2, int club1Score, int club2Score, Date matchDateObj){
      this.setClub1(club1);
      this.setClub2(club2);
      this.setClub1Score(club1Score);
      this.setClub2Score(club2Score);
      this.matchDateObj = matchDateObj;
      this.matchDate = matchDateObj.getFullDate();
   }

   public FootballClub getClub1() {
      return club1;
   }

   public void setClub1(FootballClub club1) {
      this.club1 = club1;
   }

   public FootballClub getClub2() {
      return club2;
   }

   public void setClub2(FootballClub club2){
      this.club2 = club2;
   }

   public int getClub1Score() {
      return club1Score;
   }

   public void setClub1Score(int club1Score) {
      if (club1Score >=0 && club1Score <= 20){
         this.club1Score = club1Score;
      }else{
```

```
        throw new IllegalArgumentException("Invalid, Score can't be more than 15!");
    }
}

public int getClub2Score() {
    return club2Score;
}

public void setClub2Score(int club2Score) {
    if (club2Score >=0 && club2Score <= 20){
        this.club2Score = club2Score;
    }else{
        throw new IllegalArgumentException("Invalid, Score can't be more than 15!");
    }
}

public String getMatchDate() {
    return matchDate;
}

public FootballClub getMatchWinner(){
    //
    return getClub1Score()>getClub2Score() ? getClub1():getClub2();
}

public FootballClub getMatchLooser(){
    //
    return getClub1Score()>getClub2Score() ? getClub2():getClub1();
}

public boolean isMatchDraw(){
    //
    return getClub1Score() == getClub2Score();
}

public int getWinnerClubScore(){
    //
    return Math.max(getClub1Score(), getClub2Score());
}

public int getLooserClubScore(){
    //
    return Math.min(getClub1Score(), getClub2Score());
}
```

```
    @Override
    public String toString() {
        return  "Club1      : " + club1.toString() +
                "\nClub2      : " + club2.toString() +
                "\nClub1Score  : " + club1Score +
                "\nClub2Score  : " + club2Score +
                "\nMatchDate   : " + matchDate;
    }

    @Override
    public int compareTo(Match o) {
        if (this.matchDateObj.getYear()==o.matchDateObj.getYear()) {
            if (this.matchDateObj.getMonth()==o.matchDateObj.getMonth()){
                return matchDateObj.getDay()-o.matchDateObj.getDay();
            }else{
                return this.matchDateObj.getMonth()-o.matchDateObj.getMonth();
            }
        }else{
            return this.matchDateObj.getYear()-o.matchDateObj.getYear();
        }
    }
}
```

FootballClub.java

```
public class FootballClub extends SportsClub implements Comparable<FootballClub>{

   private int numberOfWin;
   private int numberOfDraw;
   private int numberOfDefeat;
   private int numberOfGoalReceived;
   private int numberOfGoalScored;
   private int numberOfPoint;
   private int numberOfPlayedMatch;

   public FootballClub(String clubName, String location, int foundedYear) {
      super(clubName, location, foundedYear);
      this.numberOfWin = 0;
      this.numberOfDraw = 0;
      this.numberOfDefeat = 0;
      this.numberOfGoalReceived = 0;
      this.numberOfGoalScored = 0;
      this.numberOfPoint = 0;
      this.numberOfPlayedMatch = 0;
   }

   public int getNumberOfWin() {
      return numberOfWin;
   }

   public void setNumberOfWin(int numberOfWin) {
      this.numberOfWin = numberOfWin;
   }

   public int getNumberOfDraw() {
      return numberOfDraw;
   }

   public void setNumberOfDraw(int numberOfDraw) {
      this.numberOfDraw = numberOfDraw;
   }
```

```
public int getNumberOfDefeat() {
   return numberOfDefeat;
}

public void setNumberOfDefeat(int numberOfDefeat){
   this.numberOfDefeat = numberOfDefeat;
}

public int getNumberOfGoalReceived() {
   return numberOfGoalReceived;
}

public void setNumberOfGoalReceived(int numberOfGoalReceived) {
   this.numberOfGoalReceived = numberOfGoalReceived;
}

public int getNumberOfGoalScored() {
   return numberOfGoalScored;
}

public void setNumberOfGoalScored(int numberOfGoalScored) {
   this.numberOfGoalScored = numberOfGoalScored;
}

public int getNumberOfPoint() {
   return numberOfPoint;
}


public void setNumberOfPoint(int numberOfPoint) {
   this.numberOfPoint = numberOfPoint;
}

public int getNumberOfPlayedMatch() {
   return numberOfPlayedMatch;
}

public void setNumberOfPlayedMatch(int numberOfPlayedMatch) {
   this.numberOfPlayedMatch = numberOfPlayedMatch;
}
```

```
    @Override
    public String toString() {
        return   super.toString()
                + "\nWins         : " + getNumberOfWin()
                + "\nDraws        : " + getNumberOfDraw()
                + "\nDefeats      : " + getNumberOfDefeat()
                + "\nGoalsReceived : " + getNumberOfGoalReceived()
                + "\nGoalsScored   : " + getNumberOfGoalScored()
                + "\nPoints       : " + getNumberOfPoint()
                + "\nPlayedMatches : " + getNumberOfPlayedMatch();
    }

    @Override
    public int compareTo(FootballClub o) {
        if (this.numberOfPoint!=o.numberOfPoint){
            return ((this.getNumberOfPoint())-(o.getNumberOfPoint()));
        }else{
            return ((this.getNumberOfGoalScored()-this.getNumberOfGoalReceived())-(o.getNumberOfGoalScored()-
o.getNumberOfGoalReceived()));
        }
    }

}
```

SportsClub.java

```
package entities;

import java.io.Serializable;

public abstract class SportsClub implements Serializable {

   private String clubName;
   private String locationName;
   private int foundedYear;

   public SportsClub(String clubName, String location, int foundedYear){
      super();
      this.setClubName(clubName);
      this.setLocationName(location);
      this.setFoundedYear(foundedYear);
   }

   public String getClubName(){
      return clubName;
   }

   public String getLocationName(){
      return locationName;
   }

   public int getFoundedYear() {
      return foundedYear;
   }

   public void setClubName(String clubName){
      this.clubName = clubName;
   }


   public void setLocationName(String locationName){
      this.locationName = locationName;
   }

   public void setFoundedYear(int foundedYear) {
```

```
        if (foundedYear>=1750 && foundedYear<=2021){
            this.foundedYear = foundedYear;
        }else{
            throw new IllegalArgumentException("Invalid founded year, Please try again!");
        }
    }


    @Override
    public String toString(){
        return    "Club Name     : " + getClubName()
                + "\nFounded Year  : " + getFoundedYear()
                + "\nLocation Name : " + getLocationName();
    }

    @Override
    public boolean equals(Object o) {
        SportsClub sportsClub = (SportsClub) o;
        return this.foundedYear == sportsClub.foundedYear && this.clubName.equals(sportsClub.clubName) &&
this.locationName.equals(sportsClub.locationName);
    }

}
```

LeagueManager.java

```
public interface LeagueManager {

    void addClub(FootballClub club) throws IOException, ClassNotFoundException;

    void deleteExistingClub(FootballClub club, boolean deleteFromList) throws IOException,
ClassNotFoundException;

    void displayStatistics(FootballClub club) throws IOException, ClassNotFoundException;

    void displayPremierLeagueTable() throws IOException, ClassNotFoundException;

    void updateClubs(Match match) throws IOException, ClassNotFoundException;

    void updateClubWithRandomMatchDetails() throws IOException, ClassNotFoundException;

    void saveClubs(ArrayList<FootballClub> clubs) throws IOException;

    ArrayList<FootballClub> loadClubsAsArray() throws IOException,ClassNotFoundException;

    void saveMatches(ArrayList<Match> matches) throws IOException;

    ArrayList<Match> loadMatchesAsArray() throws IOException,ClassNotFoundException;

}
```

LeagueServices.java

```
public class LeagueService {

   static LeagueManager manager = new PremierLeagueManager();

   public ArrayList<FootballClub> getClubsListSortedByPoints() throws IOException, ClassNotFoundException {
      return  ApplicationUtil.bubbleSortByClubPointsAndGoalScored(manager.loadClubsAsArray());
   }

   public ArrayList<FootballClub> getClubsListSortedByWins() throws IOException, ClassNotFoundException {
      return  ApplicationUtil.bubbleSortByClubWins(manager.loadClubsAsArray());
   }

   public ArrayList<FootballClub> getClubsListSortedByGoalScored() throws IOException, ClassNotFoundException {
      return  ApplicationUtil.bubbleSortByClubGoalScored(manager.loadClubsAsArray());
   }

   public ArrayList<Match> createRandomMatches() throws IOException, ClassNotFoundException {
      manager.updateClubWithRandomMatchDetails();
      return manager.loadMatchesAsArray();
   }

   public ArrayList<Match> getMatchesList() throws IOException, ClassNotFoundException {
      return manager.loadMatchesAsArray();
   }

   public ArrayList<Match> getMatchesListSortedByDate() throws IOException, ClassNotFoundException {
      return ApplicationUtil.bubbleSortByMatchDate(manager.loadMatchesAsArray());
   }

   public ArrayList<Match> getSpecificMatches(String fullDate) throws IOException, ClassNotFoundException {
      ArrayList<Match> matchesForSpecificDate = new ArrayList<>();
      for (Match match: manager.loadMatchesAsArray()) {
         if (match.getMatchDate().equals(fullDate)){
            matchesForSpecificDate.add(match);
         }
      }
      return matchesForSpecificDate;
   }


}
```

PremierLeagueManager.java

```
public class PremierLeagueManager implements LeagueManager{

   protected static final File clubsFile = new File("app/models/clubStatistics.txt");
   protected static final File matchesFile = new File("app/models/matches.txt");

   private static final int WIN_POINTS = 3;
   private static final int DRAW_POINTS = 1;
   private static final int DEFEAT_POINTS = 0;

   @Override
   public void addClub(FootballClub club) throws IOException, ClassNotFoundException {
      ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
      boolean notExist = true;
      if (clubs.size()<20) {

         for (FootballClub footballClub: clubs){
            if (footballClub.equals(club)) {
               notExist = false;
               break;
            }
         }

         if (notExist){
            clubs.add(club);
            saveClubs(clubs);

            System.out.println("Club successfully added to the system");
            System.out.println("-----------------------------------------------------------------------------------------");
            System.out.println("Name         : " + club.getClubName());
            System.out.println("Location     : " + club.getLocationName());
            System.out.println("Founded year  : " + club.getFoundedYear());
            System.out.println("-----------------------------------------------------------------------------------------");
            System.out.println("Stored clubs  : " + clubs.size());
            System.out.println("Space        : " + (20- clubs.size()));
         }else{
            System.out.println("Invalid, Already a club exist with similar details");
         }
      }else {
         System.out.println("Invalid, Maximum number of club limit already passed!");
      }
   }
```

```
    @Override
    public void deleteExistingClub(FootballClub club, boolean deleteFromTheList) throws IOException,
ClassNotFoundException {
        ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
        for (FootballClub footballClub : clubs) {
            if (footballClub.equals(club)){
                if (deleteFromTheList) {
                    clubs.remove(footballClub);
                    saveClubs(clubs);
                    System.out.println(footballClub.getClubName() + " club removed from the system");
                }else{
                    System.out.println(footballClub.getClubName()+ " club not removed from the system");
                }
                return;
            }
        }System.out.println("Invalid, Check the club index!");
    }


    @Override
    public void displayStatistics(FootballClub club) throws IOException, ClassNotFoundException {
        ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
        for (FootballClub footballClub : clubs){
            if (footballClub.equals(club)){
                System.out.println(footballClub.toString());
                return;
            }
        }System.out.println("Invalid, Check the club index!");

    }


    @Override
    public void displayPremierLeagueTable() throws IOException, ClassNotFoundException {
        ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
        String values = "| %-21s | %-5d | %-5d | %-6d | %-5d | %-5d | %-6d | %-5d |%n";

        System.out.format("+----------------------+-------+-------+--------+-------+-------+--------+-------+%n");
        System.out.format("|      Club Name       |  Win  | Draws | Defeat |  GR   |  GS   | Points | Match |%n");
        System.out.format("+----------------------+-------+-------+--------+-------+-------+--------+-------+%n");

        for (FootballClub club : bubbleSortByClubPointsAndHighestGoal(clubs)){
```

```
System.out.printf(values,club.getClubName(),club.getNumberOfWin(),club.getNumberOfDraw(),club.getNumberOf
Defeat(),club.getNumberOfGoalReceived(),club.getNumberOfGoalScored(),club.getNumberOfPoint(),club.getNumb
erOfPlayedMatch());
      }

      System.out.format("+----------------------+-------+-------+--------+-------+-------+--------+-------+%n");
   }


   public static ArrayList<FootballClub> bubbleSortByClubPointsAndHighestGoal(ArrayList<FootballClub>
footballClubArrayList){
      int listSize = footballClubArrayList.size();
      for (int i = 0; i < listSize - 1; i++) {
         for (int j = 0; j < listSize - i - 1; j++) {
            if (footballClubArrayList.get(j).compareTo(footballClubArrayList.get(j + 1)) < 0) {
               FootballClub temp = footballClubArrayList.get(j);
               footballClubArrayList.set(j, footballClubArrayList.get(j + 1));
               footballClubArrayList.set(j + 1, temp);
            }
         }
      }return footballClubArrayList;
   }


   @Override
   public void updateClubs(Match match) throws IOException, ClassNotFoundException {
      ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
      ArrayList<Match> matches = new ArrayList<>(loadMatchesAsArray());

      if (match.isMatchDraw()) {
         updateClub(match.getClub1(), 1, 0, 0, DRAW_POINTS, match.getClub2Score(), match.getClub1Score());
         updateClub(match.getClub2(), 1, 0, 0, DRAW_POINTS, match.getClub1Score(), match.getClub2Score());
      } else if (!match.isMatchDraw()) {
         updateClub(match.getMatchWinner(), 0, 1, 0, WIN_POINTS, match.getLooserClubScore(),
match.getWinnerClubScore());
         updateClub(match.getMatchLooser(), 0, 0, 1, DEFEAT_POINTS, match.getWinnerClubScore(),
match.getLooserClubScore());
      } else {
         System.out.println("Unexpected error while updating club details!");
      }
      matches.add(match);
      saveMatches(matches);
      saveClubs(clubs);
```

```
}

@Override
public void updateClubWithRandomMatchDetails() throws IOException, ClassNotFoundException {
   ArrayList<FootballClub> clubs = new ArrayList<>(loadClubsAsArray());
   ArrayList<Match> matches = new ArrayList<>(loadMatchesAsArray());

   FootballClub randomClub1 = null;
   FootballClub randomClub2 = null;

   do {
      try{
         randomClub1 = clubs.get(((int)((Math.random() * (clubs.size()))))));
         randomClub2 = clubs.get((int)((Math.random() * (clubs.size()))));
      }
      catch(IndexOutOfBoundsException e){
         System.out.println("Invalid, Please add two clubsList to the system to create random match!");
         break;
      }
   }while(randomClub1.equals(randomClub2));

   int randomClub1Score = (int) (Math.random() * 15);
   int randomClub2Score = (int) (Math.random() * 15);

   int day;
   int month;
   int year;
   do{

      year = (int) (Math.random() * 2021) + 2020;
      month = (int) (Math.random() * 13)+ 1;
      day = (int) (Math.random() * 32)+ 1;

   }while(!Date.isYearCorrect(year) || !Date.isMonthCorrect(month) || !Date.isDayCorrect(year,month,day));

   Date date = new Date(day,month,year);
   Match match = new Match(randomClub1,randomClub2,randomClub1Score,randomClub2Score,date);

   updateClubs(match);

   matches.add(match);
   saveClubs(clubs);
   saveMatches(matches);
}
```

```
  public void updateClub(FootballClub club, int draw, int win, int defeat, int numOfPoints, int goalReceived, int goalScored){
      club.setNumberOfPlayedMatch(club.getNumberOfPlayedMatch() + 1);
      club.setNumberOfDraw(club.getNumberOfDraw() + draw);
      club.setNumberOfWin(club.getNumberOfWin() + win);
      club.setNumberOfDefeat(club.getNumberOfDefeat() + defeat);
      club.setNumberOfPoint(club.getNumberOfPoint()+ numOfPoints);
      club.setNumberOfGoalReceived(club.getNumberOfGoalReceived()+ goalReceived);
      club.setNumberOfGoalScored(club.getNumberOfGoalScored()+ goalScored);
  }

  @Override
  public void saveClubs(ArrayList<FootballClub> clubs) throws IOException {
      FileOutputStream fos = new FileOutputStream(clubsFile);
      ObjectOutputStream oos = new ObjectOutputStream(fos);

      for (FootballClub footballClub:clubs) {
          oos.writeObject(footballClub);
      }

      oos.flush();
      fos.close();
      oos.close();
  }


  @Override
  public ArrayList<FootballClub> loadClubsAsArray() throws IOException, ClassNotFoundException{
      FileInputStream fis = new FileInputStream(PremierLeagueManager.clubsFile);

      ArrayList<FootballClub> clubs = new ArrayList<>();
      try{
          ObjectInputStream ois = new ObjectInputStream(fis);
          for (;;) {
              try{
                  FootballClub club = (FootballClub) ois.readObject();
                  clubs.add(club);
              } catch (EOFException e) {
                  break;
              }
          }
          ois.close();
      }catch (EOFException ignore){}
```

```
        fis.close();
        return clubs;
    }

    @Override
    public void saveMatches(ArrayList<Match> matches) throws IOException {
        FileOutputStream fos = new FileOutputStream(PremierLeagueManager.matchesFile);
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        for (Match match:matches) {
            oos.writeObject(match);
        }

        oos.flush();
        fos.close();
        oos.close();
    }


    @Override
    public ArrayList<Match> loadMatchesAsArray() throws IOException, ClassNotFoundException{
        FileInputStream fis = new FileInputStream(PremierLeagueManager.matchesFile);

        ArrayList<Match> matches = new ArrayList<>();
        try{
            ObjectInputStream ois = new ObjectInputStream(fis);
            for (;;) {
                try{
                    Match match = (Match) ois.readObject();
                    matches.add(match);
                } catch (EOFException e) {
                    break;
                }
            }
            ois.close();
        }catch (EOFException ignore){}
        fis.close();
        return matches;
    }


}
```

club.component.ts

```
@Component({
  selector: 'app-clubs',
  templateUrl: './clubs.component.html',
  styleUrls: ['./clubs.component.css']
})

export class ClubsComponent implements OnInit {
  clubs:FootballClub[];

  constructor(private managerService:ManagerService) { }

  ngOnInit(): void {
    this.managerService.getClubs().subscribe((clubs)=>{
      this.clubs = clubs;
    });
  }

  onClickSortByWins(){
    this.managerService.getClubsSortedByWins().subscribe((clubs)=>{
      this.clubs = clubs;
    });
  }

  onClickSortByGoals(){
    this.managerService.getClubsSortedByGoals().subscribe((clubs)=>{
      this.clubs=clubs;
    });
  }

}
```

clubs.component.html

```
<div id="club-section">
  <div id="club-table">
    <table>
      <tr>
        <th>Name</th>
        <th>Win</th>
        <th>Draw</th>
        <th>Defeat</th>
        <th>Goal Received</th>
        <th>Goal Scored</th>
        <th>Point</th>
        <th>Played Match</th>
      </tr>
      <tr *ngFor="let club of clubs">
        <td>{{club.clubName}}</td>
        <td>{{club.numberOfWin}}</td>
        <td>{{club.numberOfDraw}}</td>
        <td>{{club.numberOfDefeat}}</td>
        <td>{{club.numberOfGoalReceived}}</td>
        <td>{{club.numberOfGoalScored}}</td>
        <td>{{club.numberOfPoint}}</td>
        <td>{{club.numberOfPlayedMatch}}</td>
      </tr>
    </table>
  </div>
  <div id="club-inputs">
    <div class="input-section">
      <button (click)="onClickSortByWins()" class="btn">Sort by wins</button>
      <h6>Sort by number of wins per club</h6>
    </div>
    <div class="input-section">
      <button (click)="onClickSortByGoals()" class="btn">Sort by goals</button>
      <h6>Sort by number of scored goals per club</h6>
    </div>

  </div>
</div>
```

match.component.ts

```
@Component({
 selector: 'app-matches',
 templateUrl: './matches.component.html',
 styleUrls: ['./matches.component.css']
})
export class MatchesComponent implements OnInit {
 matches:Match[];
 date:string='';
 invalid:boolean=false;

 constructor(private managerService:ManagerService) { }

 ngOnInit(): void {
  this.showMatches();
 }

 showMatches(){
  this.managerService.getMatches().subscribe((matches)=>{
   this.matches=matches;
  })
 }

 onClickCreateRandomMatch(){
  this.managerService.getRandomMatches().subscribe((matches)=>{
   this.matches=matches;
  })
 }

 onClickGetMatchesAtSpecificDate(){
  this.managerService.getMatchesAtSpecificDate(this.date).subscribe((matches)=>{
   if (matches.length===0){
    this.showMatches();
    this.invalid = true;
   }else{
    this.invalid = false;
    this.matches=matches;
   }
  });
 }

 onClickGetMatchesSortedByDate(){
```

```
    this.managerService.getMatchesSortedByDate().subscribe((matches)=>{
      this.matches=matches;
    })
  }


}
```

matches.compoent.html

```
<div id="match-section" >
    <div id="match-table">
        <table>
            <tr>
                <th>Club-1 Name</th>
                <th>Club-2 Name</th>
                <th>Club-1 Score</th>
                <th>Club-2 Score</th>
                <th>Match Date</th>
            </tr>
            <tr *ngFor="let match of matches">
                <td>{{match.club1.clubName}}</td>
                <td>{{match.club2.clubName}}</td>
                <td>{{match.club1Score}}</td>
                <td>{{match.club2Score}}</td>
                <td>{{match.matchDate}}</td>
            </tr>
        </table>
    </div>
    <div id="match-inputs">
        <div class="input-section">
            <button (click)="onClickCreateRandomMatch()" class="btn">Create random match</button>
            <h6>Create random match and add it to the matches list</h6>
        </div>
        <div class="input-section">
            <form #dateForm="ngForm" ngSubmit="onSubmit(dateForm)">
                <label>Date:</label>
                <input type="text"
                name="date"
                placeholder="dd/mm/yyyy"
                [(ngModel)]="date"
                #dateMatch="ngModel"
                required
```

```
            minlength="10"
            pattern="^(?:(?:31(\/|-|\.)(?:0?[13578]|1[02]))\1|(?:(?:29|30)(\/|-|\.)(?:0?[13-9]|1[0-2])\2))(?:(?:1[6-9]|[2-9]\d)?\d{2})$|^(?:29(\/|-|\.)0?2\3(?:(?:(?:1[6-9]|[2-9]\d)?(?:0[48]|[2468][048]|[13579][26])|(?:(?:16|[2468][048]|[3579][26])00))))$|^(?:0?[1-9]|1\d|2[0-8])(\/|-|\.)(?:(?:0?[1-9])|(?:1[0-2]))\4(?:(?:1[6-9]|[2-9]\d)?\d{2})$"
            [ngClass]="{'is-invalid':dateMatch.errors}">
            <div class="invalid-feedback" [hidden]="!dateMatch.errors?.pattern">
              Invalid date
            </div>
            <div class="invalid-feedback" [hidden]="!invalid">
              Unavailable match
            </div>
          </form>
          <button class="btn" [disabled]="dateForm.form.invalid" (click)="onClickGetMatchesAtSpecificDate()">Get Mathces</button>
        </div>
        <div class="input-section">
          <button (click)="onClickGetMatchesSortedByDate()" class="btn">Get match sorted by date</button>
          <h6>Sort matches by date</h6>
        </div>
      </div>
    </div>
```

Guide For Console

Console Instruction

The above picture shows what are the things a user can do with this program. So user can call those methods complete his necessary task. User can add a club by choosing one or remove club by choosing two or view the statistics of each club separetly by choosing three or view all clubs statistics in one table view by choosing four or update a club by creating a match between two clubs by choosing five or user can open gui by choosing option six or user can exit by choosing option seven.

Add Club

Club-1 Name	Club-2 Name	Club-1 Score	Club-2 Score	Match Date
Arsenal	Burnley	8	14	13/04/2022
Burnley	Arsenal	7	0	08/02/2022
Burnley	Arsenal	0	6	16/03/2022
Burnley	Arsenal	10	9	17/07/2020
Burnley	Arsenal	7	8	13/09/2020
Burnley	Chelsea	12	13	23/12/2020
Burnley	Chelsea	12	13	12/12/2021
Burnley	Chelsea	12	0	24/12/2022
Chelsea	Burnley	3	5	20/02/2022
Burnley	Chelsea	9	0	10/01/2021
Burnley	Chelsea	12	14	03/05/2020
Burnley	Chelsea	7	4	15/03/2021
Burnley	Chelsea	4	10	06/03/2021
Chelsea	Burnley	7	2	19/12/2020
Chelsea	Burnley	7	9	03/01/2021
Burnley	Chelsea	2	7	14/11/2022
Chelsea	Burnley	3	7	11/01/2022
Chelsea	Burnley	12	8	10/09/2022

Create random match

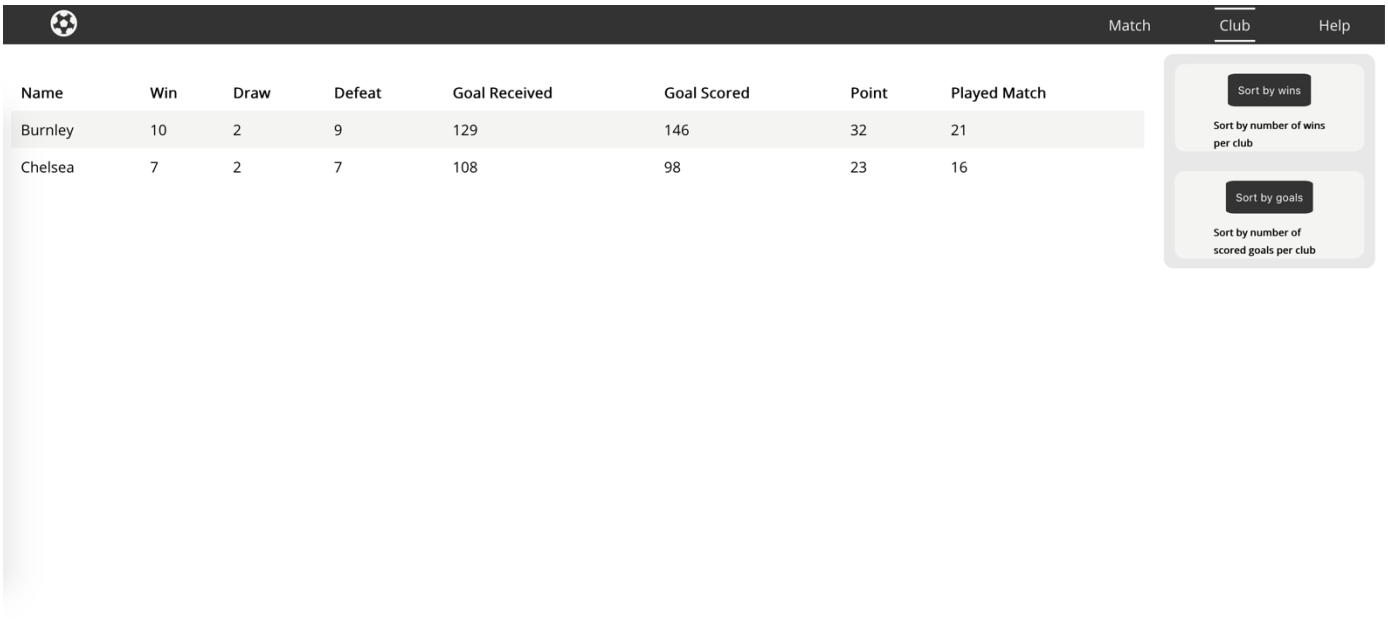
Create random match and add it to the matches list

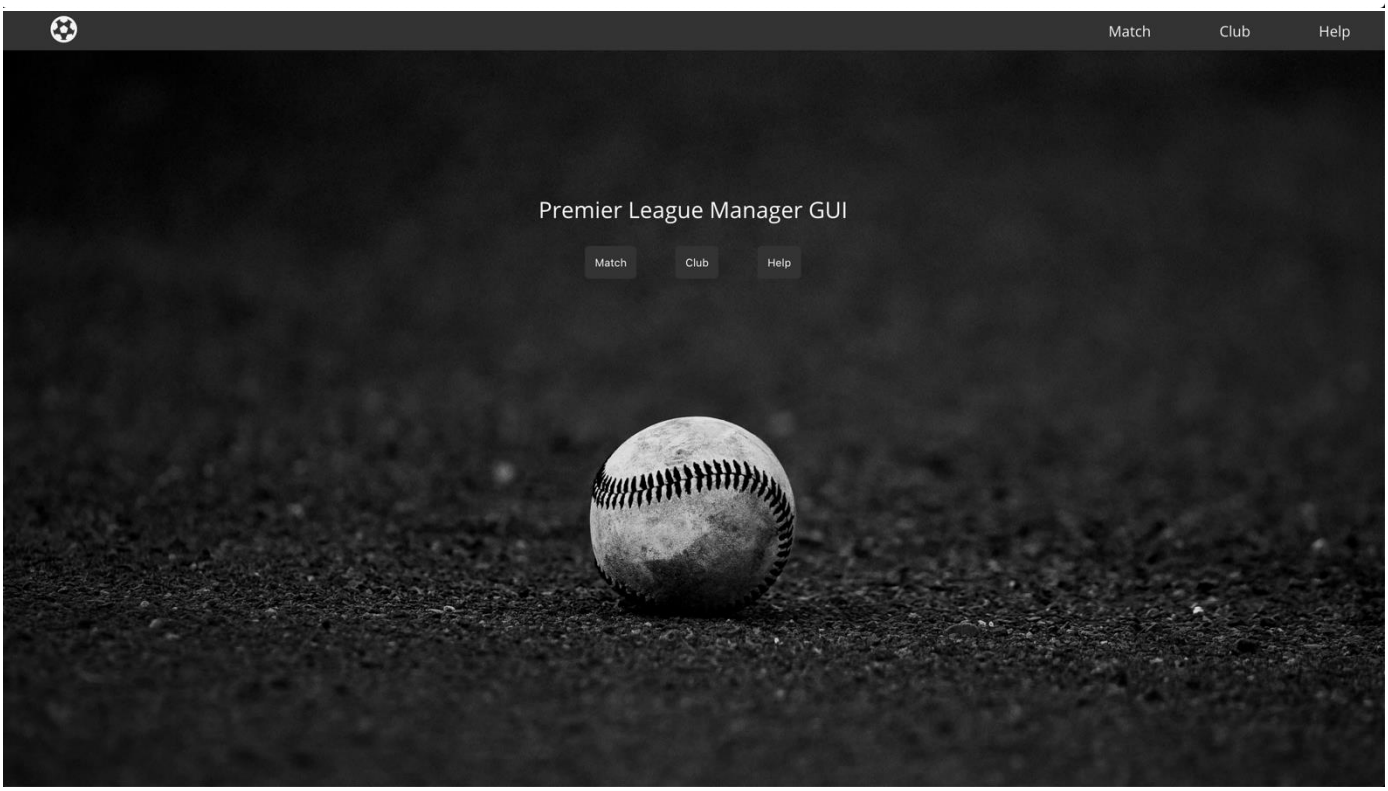
Date: dd/mm/yyyy

Get Mathces

Get match sorted by date

Sort matches by date


Match
Club
Help
Name
Win
Draw
Defeat
Goal Received
Goal Scored
Point
Played Match
Burnley
10
2
9
129
146
32
21
Chelsea
7
2
7
108
98
23
16
Sort by wins
Sort by number of wins per club
Sort by goals
Sort by number of scored goals per club


Match
Club
Help
Premier League Manager GUI
Match
Club
Help