Swinburne University of Technology Faculty of Science, Engineering and Technology

COS70006: Object Oriented Programming Project 2
10% of your final mark
Due by 3pm Friday 27 April 2018

Name: Lakshmi Saketh
Student ID: 101734216
Tutorial time:12:30-2:30

Marking Scheme

Items	Max Marks	Marks Awarded
Hard copy submitted as required: cover page (Project 2, Subject code, studentID, Name, Tutorial time), marking scheme, design documents and source code.	2	
Code readability: • Javadoc for all class headers and methods headers • Proper comments for variables and blocks • Proper indentations and use of blank lines • Use of proper Java naming conventions • Logic of code is easy to follow	10	
Detailed class diagram (all classes): • Identification of correct classes • Appropriate attributes and methods (including access modifiers) • Appropriate relationship between classes • Multiplicity used	12	
Appropriate implementation of functionality using well-structured OO classes • Implementing all the required OO classes/ methods appropriately with appropriate code • Pre-conditions are checked in methods • Implementing inheritance and polymorphism appropriately • Each function in the menu works as required • The overall product reflects OO design	20	
Appropriate implementation of user interface class • UI separated from business logic classes • Broken down into single purposed methods • Proper use of variables, access modifiers, imports etc. • Proper user messages for user inputs and outputs • The user input is safe and will not crash the program • Output is correctly formatted	12	
Saving object data between executions including File reading, writing and managing any errors	8	
Implementing an interface to sort based on appropriate field(s)	6	
Total	70	

Table of Contents

Marking Scheme 1
OPERATION 3
Start.java 3
UserInterface.java 3
Utility 11
FileUtility.java 11
DateUtility.java 12
Business 15
Sport.java 15
Basketball.java 19
Badminton.java 20
Clubs&Members 21
club.java 21
Member.java 28
Court.Java 33
Booking.Java 35

OPERATION

Start.java

```
/**
 * Write a description of class Start here.
 *
 * @author Lakshmi Saketh
 * @version 27042018
 */
import java.io.IOException;
import java.io.FileNotFoundException;
public class Start
{


    public static void main(String[] args)
    {
        Club sportsClub = new Club("Sports Club");
        UserInterface consoleApp = new UserInterface(sportsClub);
        try
        {
          sportsClub.fileReadSports();//read the Sports.txt
          sportsClub.fileReadMembers(sportsClub);//read members.txt
          sportsClub.readBookings(sportsClub);//read bookings.txt
        }
        catch(IOException e)
        {
          e.printStackTrace();
        }
        consoleApp.run();
        try
        {
            FileUtility.writeToFile(sportsClub.fileWriteBookings(),"Bookings.txt");//write the
bookings.txt
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
    }


}
```

UserInterface.java

```
import java.time.*;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeFormatterBuilder;
import java.time.temporal.ChronoUnit;
import java.util.*;

/**
 * Provides date utility methods for converting dates from string to
 * Java 8 date formats, and vice versa.
 *
 * @Lakshmi Saketh
 * @04262018
 */
public class UserInterface
{

    private Club sportsClub;//variable: sports Club
    private int duration;// variable int duration
    private Scanner sc = new Scanner(System.in);// scanner method

    /**
     * Parameterised constructor
     */
    public UserInterface(Club sportsClub)
    {
        this.sportsClub = sportsClub;
    }

    public void run()
    {
        while(true)
            switch (menu() )
            {
                    case 1:
                        showAvailableCourts();
                        break;
                    case 2:
                        makeBooking();
                        break;
                    case 3:
                        showMemberBookings();
                        break;
                     case 4:
                        showCourtBookings();
                        break;
                      case 5:
                        deleteBooking();
                        break;
```

```
              case 6:
                return;
              default:
                System.out.println ( "Invalid option" );
                break;
        }
    }

    private int menu()
    {
          System.out.println("|-----------------------------------------------|");
          System.out.println("| 1 - Show Available Courts                     |");
          System.out.println("| 2 - Make Booking for Member                   |");
          System.out.println("| 3 - Show Member Bookings                      |");
          System.out.println("| 4 - Show Court Bookings                       |");
          System.out.println("| 5 - Delete Booking                            |");
          System.out.println("| 6 - Exit                                      |" );
          System.out.println("|-----------------------------------------------|");
          System.out.println("Select your option (enter a selection number): ");
          int option = sc.nextInt();
          sc.nextLine();
          return  option;
     }

/**
 * This function used to show the avilable courts in the system
 * @return void
 */
    private void showAvailableCourts()
    {
       Scanner in=new Scanner(System.in);
       try
       {


          Scanner sc= new Scanner(System.in);
          System.out.println("Enter the Sport Name u want to Play :");
          System.out.println("Basketball\nBadminton" );
          String sportName = sc.nextLine();
          Sport s = sportsClub.findSport(sportName);
          if(s == null)
          {
             System.out.println("No Sport Found");
          }
          else
          {
             System.out.println("=========Available List=========");
          }
          for(Court co : s.getCourtList())
          {
             System.out.println("Court number :" + co.getCourtId());
          }
          System.out.println("=====================================");
       }

    catch(Exception e)
    {
       System.out.println("Exception"+e);
    }
}

    /**
     * This function helps to make booking for the court by taking the time user id as input
     *
     */
    private void makeBooking()
    {
       Scanner sc= new Scanner(System.in);
       Date dt = new Date();
       System.out.println("Enter Registered member ID: ");
       int memberID = sc.nextInt();
       if(sportsClub.memberIdentity(memberID))//check id exist or not
       {
          Member mem = sportsClub.searchMember(memberID); // get the obj of that member
based on ID
          if(mem != null)
          {
          if(mem.getFinancial())
          {


             System.out.println("Enter the Sport Name u want to Play :");
             System.out.println("Basketball\nBadminton" );
             Scanner sc1= new Scanner(System.in);
             String userSportName=sc1.nextLine();

             if(mem.statusSport(userSportName))
             {
                Scanner sc2= new Scanner(System.in);
                System.out.println("Enter the date u want to play in this format dd-MM-yyyy :
");
                String stringData=sc2.nextLine();
                LocalDate date = DateUtility.convertDate(stringData);
                System.out.println("Please enter the Start Time ");
                stringData=sc2.nextLine();
                LocalTime startTime = DateUtility.convertTime(stringData);
                 System.out.println("Please enter the End Time");
                 stringData=sc2.nextLine();
                LocalTime endTime= DateUtility.convertTime(stringData);
```

```
                DateUtility du=new DateUtility();
                duration=(int)(du.timeBetweenDateTimes(startTime,endTime));//calculats to get
the  duration
                Sport s = sportsClub.findSport(userSportName);

                if(s.timeCheck(mem.getTotalDuration(userSportName, date)+ duration))//total
duration checking (without exceeding limits
                {

                   if(sportsClub.validateTime(date, startTime, duration)==1)//check the pass
condition for the given statement
                   {

                      System.out.println("=========Available List=========");
                      for(Court courtObj : s.getAvailableCourts(date, startTime, duration))
                      {
                         System.out.println("\tCourt number " + courtObj.getCourtId() + " is
available");
                      }
                       System.out.println("======================================");
                       Scanner in= new Scanner(System.in);
                       System.out.println("Which Court would you like to play in?");
                      int courtNumber = in.nextInt();

                      Court courtObj = null;
                      for(Court c : s.getCourtList())
                      {
                         if(c.getCourtId() == courtNumber)
                         {
                            courtObj = c;
                         }
                      }

                      Booking book = new
Booking(dt.getTime(),date,startTime,endTime,mem,courtObj);
                      mem.addBooking(book);
                      courtObj.addBooking(book);
                      System.out.println("Successfully booked the court!");

                   }
                   else if(sportsClub.validateTime(date, startTime, duration)==5)
                      System.out.println("Booking done between 8-11pm");
                   else if(sportsClub.validateTime(date, startTime, duration)==4)
                      System.out.println("only 7 days Advance is available");
                   else if(sportsClub.validateTime(date, startTime, duration )==2)
                      System.out.println("Booking cannot be done");
                   else if(sportsClub.validateTime(date, startTime, duration)==1)
                      System.out.println("Bookings cant be advanved by years");
                   else if(sportsClub.validateTime(date, startTime, duration) < 0)
                      System.out.println("Bookings cant be done for previously!");
```

```
          }
          else
             System.out.println("Sorry  no slots available for sort"+userSportName);
        }
        else
        {
          System.out.println("Sorry! sport is not is not available.");

        }

      }
      else
        System.out.println("Finances are poor!");
    }
    }
    else
      System.out.println("Sorry you are not in any club!");

  }//endclass

  /**
   * This function helps to show the bookigs done by the member
   */
  public void showMemberBookings()
  {
     try
    {
      System.out.println("Please enter the memeber ID");
      int memberId =sc.nextInt();
      Member mem = sportsClub.searchMember(memberId);

      if(mem==null || mem.getBookings()==null)
      {
         System.out.println("Sorry! Member is not found.");
        }
      else
      {
        for(Booking bookingObj : mem.getBookings())
        {
           System.out.println("Booking made by "+mem.getMemberName() +" for " +
bookingObj.getBookingDate() + " at " + bookingObj.getBookingTime() + " for " +
bookingObj.getBookingEndTime() + " minutes on Court number " +
bookingObj.getCourt().getCourtId());

        }
        if(mem.getBookings().size()==0)
          System.out.println("Sorry! Currebtly no bookings done by the member ");
      }
    }
    catch(Exception e)
```

```
        {
            System.out.println("Error"+e);
        }

    }//end class


    /**
     * It helps to see the bookings done by the court
     */
    private void showCourtBookings()
    {
        ArrayList<Court> courtList = new ArrayList<Court>();
        ArrayList<Booking> bookingList = new ArrayList<Booking>();
        for(Sport sObj : sportsClub.sportList)
        {
            System.out.println("Displaying Courts for : " + sObj.getSportName());
            courtList = sObj.getCourtList();
            for(Court cObj : courtList)
            {
                if(cObj.getCourtBookings().size()==0)
                    System.out.println("Booking are not yet started for sport :" +
sObj.getSportName() + " on Court : " + cObj.getCourtId());
                else
                {

                    Collections.sort(cObj.getCourtBookings());
                    System.out.println(cObj.getCourtBookings().toString());

                }
            }
        }
    }//End class




      /**
     * this function is used to delete a booking
     *
     *
     */
    private void deleteBooking()
    {
        Scanner memberID=new Scanner(System.in);
                     System.out.println("Please enter your member number: ");
                        int memberId  = memberID.nextInt();

        Member mem = sportsClub.searchMember(memberId);
        if(mem==null || mem.getBookings()==null)
    {
       System.out.println("Sorry! There are no members with the given ID. ");



    }
    else
    {
      for(Booking b : mem.getBookings())//Displays the member details n bookings
      {
        System.out.println("Booking Id : " + b.hashCode()+ " Booking made by
"+mem.getMemberName() +" for " + b.getBookingDate() + " at " + b.getBookingTime() + "
for " + b.getBookingEndTime() + " minutes on Court number " + b.getCourt().getCourtId());

      }

      if(mem.getBookings().size()!=0)
      {
            Scanner inputID=new Scanner(System.in);
             System.out.println("Enter Booking ID: ");
            int input = inputID.nextInt();
        Iterator<Booking> itr = mem.getBookings().iterator();


        while(itr.hasNext())//delettion starts based on the ID
        {
          if(itr.next().hashCode() == input)
          {
            itr.remove();
            for(String str : mem.getSportsPlayed())
            {
              Sport sportObj = sportsClub.findSport(str);

              ArrayList<Court> itrCourt = sportObj.getCourtList();
              for(Court c : itrCourt)//finds the court
              {

                Iterator<Booking> itrBooking = c.getCourtBookings().iterator();
                while(itrBooking.hasNext())//helps to get the booking object
                {
                  if(itrBooking.next().hashCode() == input)
                  {
                    itrBooking.remove();//removes the booking object
                    System.out.println("Deleted Successfully");
                  }
                }
              }
            }
          }
        }
```

```
          }
        }



    }    //endclass




} // end class
```

Utility

FileUtility.java

```
import java.util.*;
import java.io.*;
public class FileUtility
{
    public FileUtility()
    {
    }

    /**
     * Reads data from file returning the lines as a list, or null if error
     *
     */
    public static ArrayList<String> readFromFile(String fileName) throws IOException_
    {
       ArrayList<String> fileData = new ArrayList<>();
       //File filePath = new
File("F:/SwinBurne_Sem2/OOPS/Proj2/PageLoadingEffects/Project2_Start_Up_Code/"+file
Name);
        BufferedReader sc = new BufferedReader(new FileReader(fileName));

        String value = sc.readLine();
        while (value != null) {
            fileData.add(value);
            value = sc.readLine();
        }
        sc.close();
        return fileData;

    }

    /**
     * Write data to file
     *
     */
    public static void writeToFile(ArrayList<String> data,String fileName ) throws
IOException_
    {
        try (PrintWriter pw = new PrintWriter(new FileWriter(fileName, true))) {
            for (String s : data) {
                pw.println(s);
            }
        } catch (IOException e) {
            System.out.println("Error in write File:" + e);
        }
    }

}
```

DateUtility.java

```
/**
 * Class creates methods for date conversion
 * Java 8 date formats, and vice versa.
 *
 * @Lakshmi Saketh
 * @04262018
 */

public class DateUtility
{
    private static final String PATTERN_Date = "dd-MM-yyyy";
    private static final String PATTERN_Date2 = "dd-MMM-yyyy";


    public static LocalDate convertDate(String dateInput)
    {
        DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern(PATTERN_Date);
        return LocalDate.parse(dateInput, dateFormat);


    }


    public static LocalTime convertTime(String timeInput)
    {
        DateTimeFormatter parseFormat = new
DateTimeFormatterBuilder().appendPattern("H:m").toFormatter();
        return LocalTime.parse(timeInput, parseFormat);

    }
    /**
     * This gives the local date of system
     *
     * @return Local Sytem date or server date.
     */
    public static LocalDate getCurrentDate() {
        return LocalDate.now();
    }

    /**
     * This gives the local system time
     *
     * @return local system time
     */
    public static LocalTime getCurrentTime() {
        return LocalTime.now();
    }

    /**
     * This gives both the date and time
     *
     * @ retuns the date and time of local system.
     */

    public static LocalDateTime getCurrentDateTime() {
        return LocalDateTime.now();
    }

    /**
     * generates localised date from provided string Date
     *
     * @param inputDate provided string date
     * @return Localised date as per the requirements
     */
    public static String getLocalisedDate(LocalDate inputDate) {
        DateTimeFormatter df = DateTimeFormatter.ofPattern(PATTERN_Date2,
Locale.getDefault());
        return df.format(inputDate);
    }

    /**
     * Returns the days between provided dates
     *
     * @param date for comparison
     * @return difference between dates
     * <p><p>
     * See <a href
="https://stackoverflow.com/a/29812532/3796452">https://stackoverflow.com/a/29812532/3
796452</a> for reference
```

```
     */
    public static long daysBetweenDates(LocalDate firstDate, LocalDate secondDate) {
        return ChronoUnit/.DAYS.between(firstDate, secondDate);
    }

    /**
     * Returns the days between provided dates
     *
     * @param date for comparison
     * @return difference between times
     * <p><p>
     * See <a href
="https://stackoverflow.com/a/29812532/3796452">https://stackoverflow.com/a/29812532/3
796452</a> for reference
     */
    public static long timeBetweenDateTimes(LocalTime firstTime, LocalTime secondTime) {
        return ChronoUnit.MINUTES.between(firstTime, secondTime);
    }


    /**
     * Returns the days between provided dates
     *
     * @param date for comparison
     * @return difference between dates and times
     * <p><p>
     * See <a href
="https://stackoverflow.com/a/29812532/3796452">https://stackoverflow.com/a/29812532/3
796452</a> for reference
     */
    public static long hoursBetweenDateTimes(LocalDateTime firstTime, LocalDateTime
secondTime) {
        return ChronoUnit.HOURS.between(firstTime, secondTime);
    }
}




    /**
     * Returns the time between provided times in hours
     *
     * @param firstTime  First time with which comparison is to be done
     * @param secondTime second time with which comparison is to be done
     * @return number of hours between provided times
     */
    public static long hoursBetweenDateTimes(LocalDateTime firstTime, LocalDateTime
secondTime) {
        return ChronoUnit.HOURS.between(firstTime, secondTime);
    }
```

```
}
```

Business

Sport.java

```
/**
 * Write a description of class Lab here.
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.time.*;
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.text.ParseException;

public abstract class Sport
{
    private ArrayList<Court> courtList=new ArrayList<Court>();
    private int usageFee;
    private int insurance;
    private int slotTime;
    private String sportName;
    private int timeDuration;

    public Sport()
    {

    }

   /**
     * parameterized constructor for the class sport
     * @param sports data of string type and duration of int datatype
     */
    public Sport(String data, int duration) throws IOException, FileNotFoundException
    {
       try
       {

        String[] list = data.split(",");
        sportName=list[0].trim();
        usageFee=Integer.parseInt_(list[1].trim());
        insurance=(Integer.parseInt_(list[2].trim()));
        Court courtobj;
        for(int i=3;i<list.length;i++)
        {
        courtList.add(new Court(Integer.parseInt_(list[i].trim())));
        }
        timeDuration=duration;
    }
    catch(Exception E)
    {
        System.out.println("Exception Caught:"+ E);
    }

    }


    //abstarct Method we implement same in both the child classes
    public abstract boolean timeCheck(int timeCheck);

        /**
         * accessor for getusageFee variable
         *
         * @return usageFee in int format
         */
       public int getUsageFee()
        {
            return usageFee;
        }

        /**
         * mutator for setusageFee variable
         *
         * @param usageFee  provided in int format
         */
        public void setUsageFee(int usageFee)
        {
             this.usageFee=usageFee;
        }

        /**
         * accessor for getinsurance variable
         *
         * @return insurance in int format
         */
       public int getInsurance()
        {
            return insurance;
        }

        /**
         * mutator for setinsurance variable
         *
         * @param insurance  provided in int format
         */
        public void setInsurance(int insurance)
        {
```

```
     this.insurance=insurance;
  }

   /**
   * accessor for getslotTime variable
   *
   * @return slotTime in int format
   */
 public int getSlotTime()
  {
    return slotTime;
  }

  /**
   * mutator for setslotTime variable
   *
   * @param slotTime  provided in int format
   */
  public void setSlotTime(int slotTime)
  {
     this.slotTime=slotTime;
  }

 /**
   * accessor for generating court list
   */
  public ArrayList<Court> getCourtList()
  {
    return courtList;
  }

  /**
   * mutator to access courtList list
   *
   * @param courtList user provided data for courtList list
   */
  public void setCourtList(ArrayList<Court> courtList)
  {
    for(Court c : courtList )
  {
    this.courtList.add(c);
  }
  }

  /**
   * accesssor to getSportName
   *
   */

  public String getSportName()
```

```
        {
           return sportName;
        }

        /**
         * mutator to access set Sport Name
         *
         * @param set Sport Name user provided data for Sport Name
         */

        public void setSportName(String sportName)
        {
           this.sportName = sportName;
        }

        /**
         * this function gives the available courts based on the date, local time and duration
         * @param localdate,localTime,duration
         * @return court of arrayList
         **/
        public ArrayList<Court> getAvailableCourts(LocalDate date, LocalTime time, int
duration)
        {
           ArrayList<Court> courtsList=new ArrayList<Court>();

           for(Court courtObj : courtList)
           {
              if(courtObj.availabilityStatus(date, time, duration))
                    {

                       courtsList.add(courtObj);

                    }
            }

           return courtsList;
        }

    public String toString()
    {
        return "Sport{" +
                "Usage Fee='" + usageFee + '\'' +
                ", Insurance='" + insurance + '\'' +
                ", Courts List='" + courtList.toString() +
                '}';

    }
}
```

Basketball.java

```
/**
 * BasketBall class stores the details of the basketball Game.
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
public class Basketball extends Sport
{
    private static final int Slot_Time = 180;
    private ArrayList<Court> bList=new ArrayList<Court>();
    private double netHeight;
    /**
     * Constructor for objects of class BasketBall
     */
    public Basketball(String data) throws IOException, FileNotFoundException
    {



        super(data,Slot_Time);// Daily Max time for slot time is 3Hrs


    }


    /**
     * this function checks the limit of the slot
     * @param time as input to check
     * @return bool datatype
     */
    public boolean timeCheck(int timecheck)
    {
            if(Slot_Time >= timecheck)
            {
                        return true;
            }
            else
                        return false;
    }

    /**
     * get method for net height
     * @return net height
     */
    public double getNetHeight() {
        return netHeight;
    }

    /**
     * set method for net height
     * @param netHeight
     */
    public void setNetHeight(double netHeight) {
        this.netHeight = netHeight;
    }

    public String toString()
    {
        return  "BasketBall{" +
                "Slot Time Booking per Day =" + Slot_Time +
                '}';

    }

}
```

Badminton.java

```
/**
 * Badminton class stores the details of badminton game.
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
public class Badminton extends Sport
{

    private boolean isRacquetProvided = false;
    private static final int Slot_Time = 120;
    //private ArrayList<Court> bdList=new ArrayList<Court>();
    /**
     * Constructor for objects of class Badminton
     */
    public Badminton(String data) throws IOException, FileNotFoundException
    {
            super(data,Slot_Time); // Daily Max time for slot time is 3Hrs
    }

     /**
     * this function checks the limit of the slot
     * @param time as input to check
     * @return bool datatype
     */
    public boolean timeCheck(int timecheck)
    {
        if(Slot_Time >= timecheck)
        {
            return true;
        }
        else
            return false;
    }

    /**
     * get method for the racquet required
     * @return boolean status
     */
     public boolean isRacquetProvided() {
        return isRacquetProvided;
    }

    /**
     * set method for racquet required
     * @param boolean type racquet required.
     */
    public void setRacquetProvided(boolean racquetProvided) {
        isRacquetProvided = racquetProvided;
    }

    public String toString()
    {
         return  "Badminton{" +
                "Slot Time Booking per Day =" + Slot_Time +
                '}';
    }

}
```

Clubs&Members

club.java

```
/**
 * Club class stores the details of the club .
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.temporal.Temporal;
import java.time.*;
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
public class Club
{
   // Variables declared
   public static ArrayList<Member> memberList;
   public static ArrayList<Sport> sportList;
   String clubName;
   final static LocalTime courtOpenTime = LocalTime.of(8, 00, 00, 00);
   final static LocalTime courtEndTime = LocalTime.of(23, 00,00,00);
   /**
    * parameterized constructor with the string name
    */
   public Club(String name)
   {
      clubName = name;
      memberList = new ArrayList<Member>();
      sportList = new ArrayList<Sport>();

   }



      /**
       * accessor for generating court list
       */
      public ArrayList<Sport> getSportList()
      {
         return sportList;
      }

      /**
       * mutator to access courtList list
       *
       * @param courtList user provided data for courtList list
       */
      public void setSportList(ArrayList<Sport> SportList)
      {
         this.sportList=sportList;
      }
      /**
       * accessor for generating court list
       */
      public ArrayList<Member> getMemberList()
      {
         return memberList;
      }

      /**
       * mutator to access courtList list
```

```
     *
     * @param courtList user provided data for courtList list
     */
    public void setMemberList(ArrayList<Member> memberList)
    {
        this.memberList=memberList;
    }
  /**
   * This helps to identify the member with their ID
   * @param member ID of int
   * @return Member object.
   *
   */
  public Member searchMember(int id)
  {
      Member var = null;
      try
      {
      for(Member m : memberList)
      {
          if(m.getMemberID() == id)
          {
              var = m;
          }
      }
      if(var==null)
      {
          String str = "Could not find the member with the ID";

      }
      }
      catch(Exception e)
      {
          System.out.println("Exception:" + e);
      }
      return var;
  }

  /**
   * this function helps to find the identity of member
   * @param id of int type it is member id
   * @return boolean whether the memeber with that id exist or not
   *
   */
  public boolean memberIdentity(int id)
  {
      boolean bool = false;
      for(Member mem : memberList)
      {
          if(mem.getMemberID() == id)
```

```
            bool = true;
        }

        return bool;

    }

    /**
     * this function helps to identify the court based on ID
     * @param ID of int data type
     * @return object of Court
     *
     */
    public Court searchCourt(int courtID)
    {
        for(Sport sportObj : sportList)
        {
            ArrayList<Court> CourtList = sportObj.getCourtList();
            for(Court courtObj : CourtList)
            {
                if(courtObj.getCourtId()==courtID)
                {
                    return courtObj;
                }
            }
        }
        return null;
    }

    /**
     * this function used for validation of the time of office
     * @param date,time,duration
     * @return int returns the availability as an Integer
     */
    public int validateTime(LocalDate checkDate, LocalTime checkTime, int checkDuration)
    {
        Period difference = LocalDate.now().until(checkDate);
        int checkYears = difference.getYears();
        int checkMonths = difference.getMonths();
        int checkDays = difference.getDays();
        if(checkYears < 0 || checkMonths < 0 || checkDays < 0)
            return -1;
        else if(checkYears == 0)
        {
            if(checkMonths == 0)
            {
                if(checkDays <=7 && checkDays !=0)
                {
                    if((checkTime.isAfter(courtOpenTime)||checkTime.equals(courtOpenTime))&&
checkTime.plusMinutes(checkDuration).isBefore(courtEndTime))
```

```
                {
                    return 1;
                }
                else
                    return 2;
            }
            else if(checkDays==0)
            {
                if(checkTime.isAfter(LocalTime.now()))
                    return 1;
                else
                    return -1;
            }
            else
                return 3;
        }
        else
            return 5;
    }
    else
        return 1;
}


/**
 * this function used for getting the sportObj based on the string sportname
 * @param sport Name fromuser
 * @return sport object
 *
 */
public Sport findSport(String sport)
{
    Sport sportObj = null;
    //System.out.println(sportList.toString());
    for(Sport s : sportList)
    {
        if(s.getSportName().equals(sport))
        {
            System.out.println(s.getSportName());
            sportObj= s;
        }
    }
    return sportObj;
}

    /**
 * This functions used to read file of Sports
 *
 */
```

```
public void fileReadSports() throws IOException_,FileNotFoundException
{
   Sport sp=null;
   try
   {
   ArrayList<String> sportData = FileUtility.readFromFile("sports.txt");
   for(String sportString : sportData)
   {
      String[] data = sportString.split(",");
      if(data[0].equalsIgnoreCase("Basketball"))
      {
         sp = new Basketball(sportString);

      }
      else if(data[0].equalsIgnoreCase("Badminton"))
      {
         sp = new Badminton(sportString);

      }

      else
      {
         System.out.println("Please send proper file Sports.txt");
      }
      sportList.add(sp);

   }
   }
   catch(Exception e)
   {
      System.out.println("Exception:"+e);
   }

}

 /**
 * this function used to read the file members from members.txt
 * @param club object
 *
 */
public void fileReadMembers(Club clubObj) throws IOException
{
   Member b;

   ArrayList<String> memberData = FileUtility.readFromFile("members.txt");
   for(String memObj : memberData)
   {
      b= new Member(memObj, clubObj);
      memberList.add(b);
   }
```

```
    }



    /**
     * this function is used to write the bookings in bookings.txt
     * @return ArrayList of string data type.
     *
     */
    public ArrayList<String> fileWriteBookings()
    {
        ArrayList <String> bookingList = new ArrayList<String>();
        try
        {
            for(Member mem : memberList)
            {

                ArrayList<Booking> bookingsList = mem.getBookings();
                if(bookingsList!=null)
                {
                for(Booking b : bookingsList)
                {
                    if(b!=null)
                    {
                        String Data = b.storeDetails();
                        bookingList.add(Data);
                    }
                    else
                        System.out.println("IT should hav some data to write");
                }
            }

           }

        }
        catch(Exception e)
        {
            System.out.println(e);
        }

        return bookingList;
    }

    /**
     *  This helps to read bookings from bookings.txt
     *  @param ClubOBJ of club datatype.
     */
     public void readBookings(Club clubObj) throws IOException
    {
        ArrayList<String> bookingsList = FileUtility.readFromFile("Bookings.txt");
        for(String data : bookingsList)
        {
            try
            {
            Booking booking = new Booking(data,this);
            if(booking.getBookingDate()==null)
              System.out.println("Object is Null");
            else
            {
            Member mem = booking.getMember();
            Court court = booking.getCourt();
            court.addBooking(booking);
            mem.addBooking(booking);
        }
            }
            catch(Exception e)
            {
                String str = "Issue :"+e;
            }
        }
    }

}
```

Member.java

```
/**
 * Club class stores the details of the club .
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.util.*;
import java.time.LocalDate;
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
public class Member
{
    private int memberID;
    private String memberName;
    private boolean financial;
    private ArrayList<String> sportList;
    private ArrayList<Booking> bookingList;
    /**
     * Constructor the class Member objects
     */
    public Member()
    {
    }
    /**
     * parameterized concructor
     * @param name of string datatype
     * @thows IO exception, File found exception
     */
    public Member(String name) throws IOException, FileNotFoundException
    {
       bookingList = new ArrayList<Booking>();
        String[] list = name.split(",");
        this.memberID = Integer.parseInt(list[0].trim());
        this.memberName = list[1].trim();
        this.financial=Boolean.parseBoolean(list[2].trim());
        if(list[2].equalsIgnoreCase("true") && list[3]!=null && (list[3].equals("Badminton") ||
list[3].equals("Basketball")))
        {
            for(int i=3;i<list.length;i++)
            {
        sportList.add(list[i]);

            }
        }

    }


    /**
     * this function is used to get the total duration of the booking
     * @param sport and date
     * @return int returns the total duration
     */

    public int getTotalDuration(String sportName, LocalDate date)
    {
        int duration = 0;

        for(String s : sportList)
        {
            if(s.equalsIgnoreCase(sportName) )
            {
                for(Booking b : bookingList)
                {

                    if(b.getBookingDate().equals(date))
                    {
                            DateUtility du=new DateUtility();
                        duration = duration +
(int)(du.timeBetweenDateTimes(b.getBookingTime(),b.getBookingEndTime()));
                    }
                }

            }
        }
        return duration;

    }



    /**
     * This constructor takes in a string and initialises the variables of the member class
     * @param memberData and club Obj
     */
    public Member(String memberData, Club clubObj)
    {
        String[] splitData = memberData.split(",");

        try
        {

            memberID = Integer.parseInt(splitData[0]);
            memberName = splitData[1];
            sportList = new ArrayList<String>();
            if(splitData[2].equals("true")||splitData[2].equals("false"))
            {
                financial = Boolean.parseBoolean(splitData[2]);
            }
            else
            {
                financial=Boolean.parseBoolean(splitData[2]);
                for(int i = 2;i<splitData.length;i++)
                {
                    for(Sport s : clubObj.sportList)
                    {
                        if(s.getSportName().equalsIgnoreCase(splitData[i]))
                            sportList.add(splitData[i]);
                    }

                }
            }
            for(int i = 3;i<splitData.length;i++)
            {
                for(Sport s : clubObj.sportList)
                {
                    if(s.getSportName().equalsIgnoreCase(splitData[i]))
                        sportList.add(splitData[i]);

                }
            }
            bookingList = new ArrayList<Booking>();
        }
        catch(Exception e)
        {
            String str = "Error  :"+e;
        }
    }
    /**
     * Get method for the member ID
     * @return int of MemberID
     */
    public int getMemberID()
    {
        return memberID;
    }

     /**
     * Get method for the member ID
     * @return String of memberName
     */
    public String getMemberName()
    {
        return memberName;
    }

     /**
     * Get method for bookings
     * @return list of bookingList
     */
    public ArrayList<Booking> getBookings()
    {
        return bookingList;
    }

    /**
     * Get method for booking OBj
     * @return list of bookingList
     */
    public void addBooking(Booking bookingObj)
    {
        bookingList.add(bookingObj);
    }

    /**
     * This function removes the booking object
     * @param booking obj
     *
     */

    public void removeBooking(Booking bookingObj)
    {
        bookingList.remove(bookingObj);
    }

    /**
```

```
     * Get method for sports played
     * @return list of Sports List
     */
    public ArrayList<String> getSportsPlayed()
    {
        return sportList;
    }

    /**
     * get method for financial
     * @return boolean of financial
     */
    public boolean getFinancial()
    {
        return financial;
    }


    /**
     * this function checks if a member is playing a sport
     * @param sportName of string type
     * @return boolean returns true if the sport is played by a member
     */

    public boolean statusSport(String name)
    {
        boolean bool = false;
        for(String s : sportList)
        {
            if(s.equalsIgnoreCase(name))
            {
                bool = true;
            }
        }
        return bool;
    }









    public String toString()
    {
        return "Member Details{" +
                "Member ID='" + memberID + '\'' +
                ", Member Name='" + memberName + '\'' +
                ", Financial ='" + financial +
                ", Sports Played ='" + sportList.toString() +
                '}';
    }

}
```

Court.Java

```
import java.time.*;
import java.util.*;

/**
 * Provides date utility methods for converting dates from string to
 * Java 8 date formats, and vice versa.
 *
 * @Lakshmi Saketh
 * @04262018
 */
public class Court
{
    private ArrayList<Booking> courtBookings;
    private int courtId;

    /**
     * parameterized constructor
     * @param int of CourtID
     */
    public Court(int courtId)
    {
        this.courtId=courtId;
        courtBookings=new ArrayList<Booking>();

    }

    /**
     * get method used for getting court ID
     * @return courtID ofint
    */
    public int getCourtId()
    {
        return courtId;
    }
    /**
     * this function is used to add a booking
     * @param BOOKING OBJ
     *
     */
    public void addBooking(Booking b)
    {
        courtBookings.add(b);


    }

    /**
     * this gives us array list of courtbookings
     * @return the court bookings
    */
    public ArrayList<Booking> getCourtBookings()
    {
        return courtBookings;
    }
    /**
     * this function is used for setting court Bookings
     * @param Array List of court Bookings
     *
     */
    public void setCourtBookings(ArrayList<Booking> courtBookings)
    {
        this.courtBookings=courtBookings;


    }



        /**
         * this function is used to check the availability status of the court
         * @param date date,time,duration
         * @return boolean
         */
        public boolean availabilityStatus(LocalDate date, LocalTime time, int duration)
        {

            for(Booking b : courtBookings)
            {
                if(b.getBookingDate().isEqual(date))
                {

                    if(!b.overTime(time, duration))
                    {

                        return true;
                    }
                    return false;
                }
                else
                    return true;
            }
            return true;
        }
}
```

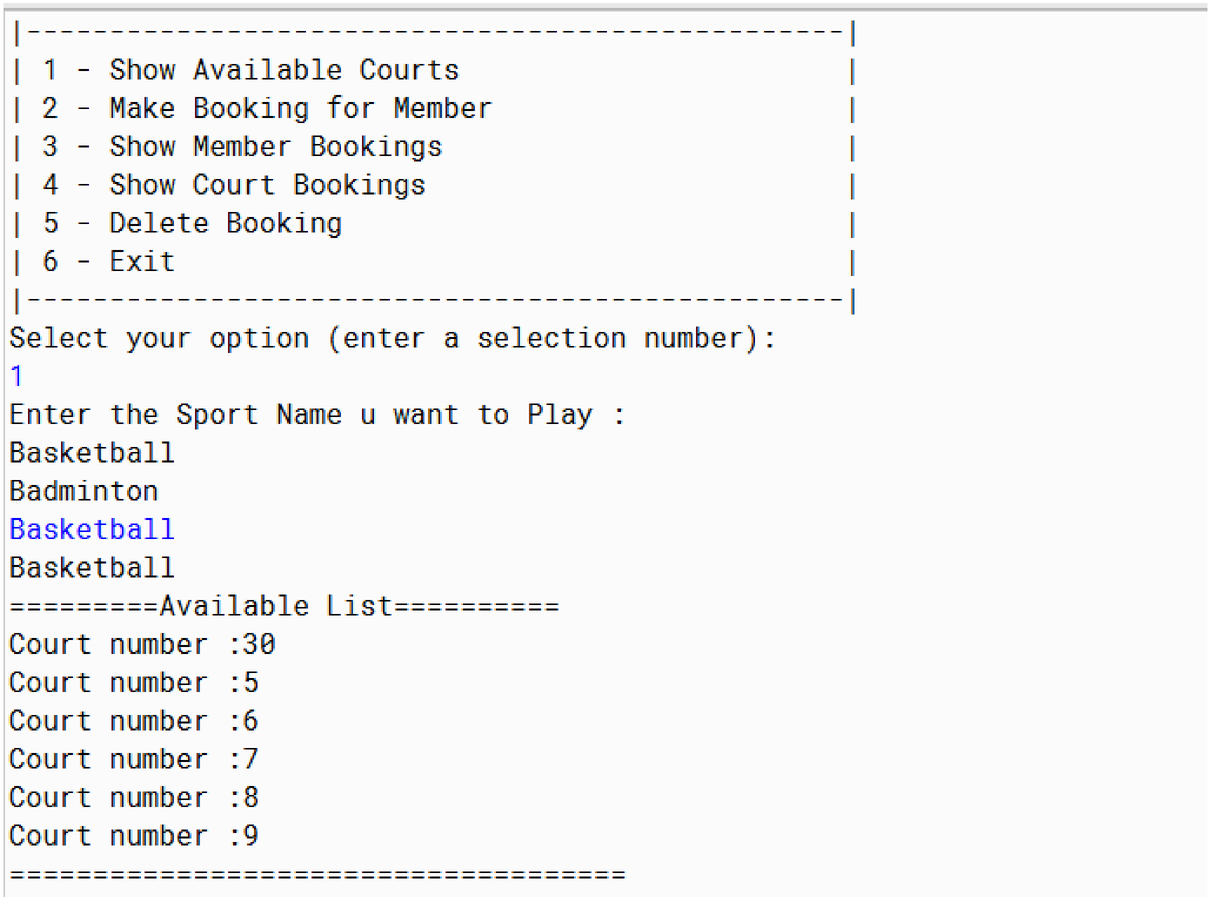
Booking.Java

```
/**
 * Club class stores the details of the club .
 * Stores all the Lab Class Data get and set methods
 * @author LakshmiSaketh
 * @version 04262018
 */
import java.util.*;
import java.time.LocalDate;
import java.util.*;
import java.io.IOException;
import java.io.FileNotFoundException;
public class Member
{
    private int memberID;
    private String memberName;
    private boolean financial;
    private ArrayList<String> sportList;
    private ArrayList<Booking> bookingList;
    /**
     * Constructor the class Member objects
     */
    public Member()
    {
    }
    /**
     * parameterized concructor
     * @param name of string datatype
     * @thows IO exception, File found exception
     */
    public Member(String name) throws IOException, FileNotFoundException
    {
       bookingList = new ArrayList<Booking>();
        String[] list = name.split(",");
        this.memberID = Integer.parseInt(list[0].trim());
        this.memberName = list[1].trim();
        this.financial=Boolean.parseBoolean(list[2].trim());
        if(list[2].equalsIgnoreCase("true") && list[3]!=null && (list[3].equals("Badminton") ||
list[3].equals("Basketball")))
        {
            for(int i=3;i<list.length;i++)
            {
        sportList.add(list[i]);
```

```
            }
        }

    }



    /**
     * this function is used to get the total duration of the booking
     * @param sport and date
     * @return int returns the total duration
     */

    public int getTotalDuration(String sportName, LocalDate date)
    {
        int duration = 0;

        for(String s : sportList)
        {
            if(s.equalsIgnoreCase(sportName) )
            {
                for(Booking b : bookingList)
                {

                    if(b.getBookingDate().equals(date))
                    {
                            DateUtility du=new DateUtility();
                        duration = duration +
(int)(du.timeBetweenDateTimes(b.getBookingTime(),b.getBookingEndTime()));
                    }
                }

            }
        }
        return duration;

    }



    /**
     * This constructor takes in a string and initialises the variables of the member class
     * @param memberData and club Obj
     */
    public Member(String memberData, Club clubObj)
    {
        String[] splitData = memberData.split(",");

        try
        {

            memberID = Integer.parseInt(splitData[0]);
            memberName = splitData[1];
            sportList = new ArrayList<String>();
            if(splitData[2].equals("true")||splitData[2].equals("false"))
            {
                financial = Boolean.parseBoolean(splitData[2]);
            }
            else
            {
                financial=Boolean.parseBoolean(splitData[2]);
                for(int i = 2;i<splitData.length;i++)
                {
                    for(Sport s : clubObj.sportList)
                    {
                        if(s.getSportName().equalsIgnoreCase(splitData[i]))
                            sportList.add(splitData[i]);
                    }

                }
            }
            for(int i = 3;i<splitData.length;i++)
            {
                for(Sport s : clubObj.sportList)
                {
                    if(s.getSportName().equalsIgnoreCase(splitData[i]))
                        sportList.add(splitData[i]);

                }
            }
            bookingList = new ArrayList<Booking>();
        }
        catch(Exception e)
        {
            String str = "Error  :"+e;
        }
    }
    /**
     * Get method for the member ID
     * @return int of MemberID
     */
    public int getMemberID()
    {
        return memberID;
    }

     /**
     * Get method for the member ID
     * @return String of memberName
     */
    public String getMemberName()
    {
        return memberName;
    }

     /**
     * Get method for bookings
     * @return list of bookingList
     */
    public ArrayList<Booking> getBookings()
    {
        return bookingList;
    }

    /**
     * Get method for booking OBj
     * @return list of bookingList
     */
    public void addBooking(Booking bookingObj)
    {
        bookingList.add(bookingObj);
    }

    /**
     * This function removes the booking object
     * @param booking obj
     *
     */

    public void removeBooking(Booking bookingObj)
    {
        bookingList.remove(bookingObj);
    }

    /**
     * Get method for sports played
     * @return list of Sports List
     */
    public ArrayList<String> getSportsPlayed()
    {
        return sportList;
    }

    /**
     * get method for financial
     * @return boolean of financial
     */
    public boolean getFinancial()
    {
        return financial;
    }
    /**
     * this function checks if a member is playing a sport
     * @param sportName of string type
     * @return boolean returns true if the sport is played by a member
     */

    public boolean statusSport(String name)
    {
        boolean bool = false;
        for(String s : sportList)
        {
            if(s.equalsIgnoreCase(name))
            {
                bool = true;
            }
        }
        return bool;
    }

    public String toString()
    {
        return "Member Details{" +
                "Member ID='" + memberID + '\'' +
                ", Member Name='" + memberName + '\'' +
                ", Financial ='" + financial +
                ", Sports Played ='" + sportList.toString() +
                '}';
    }

}
```

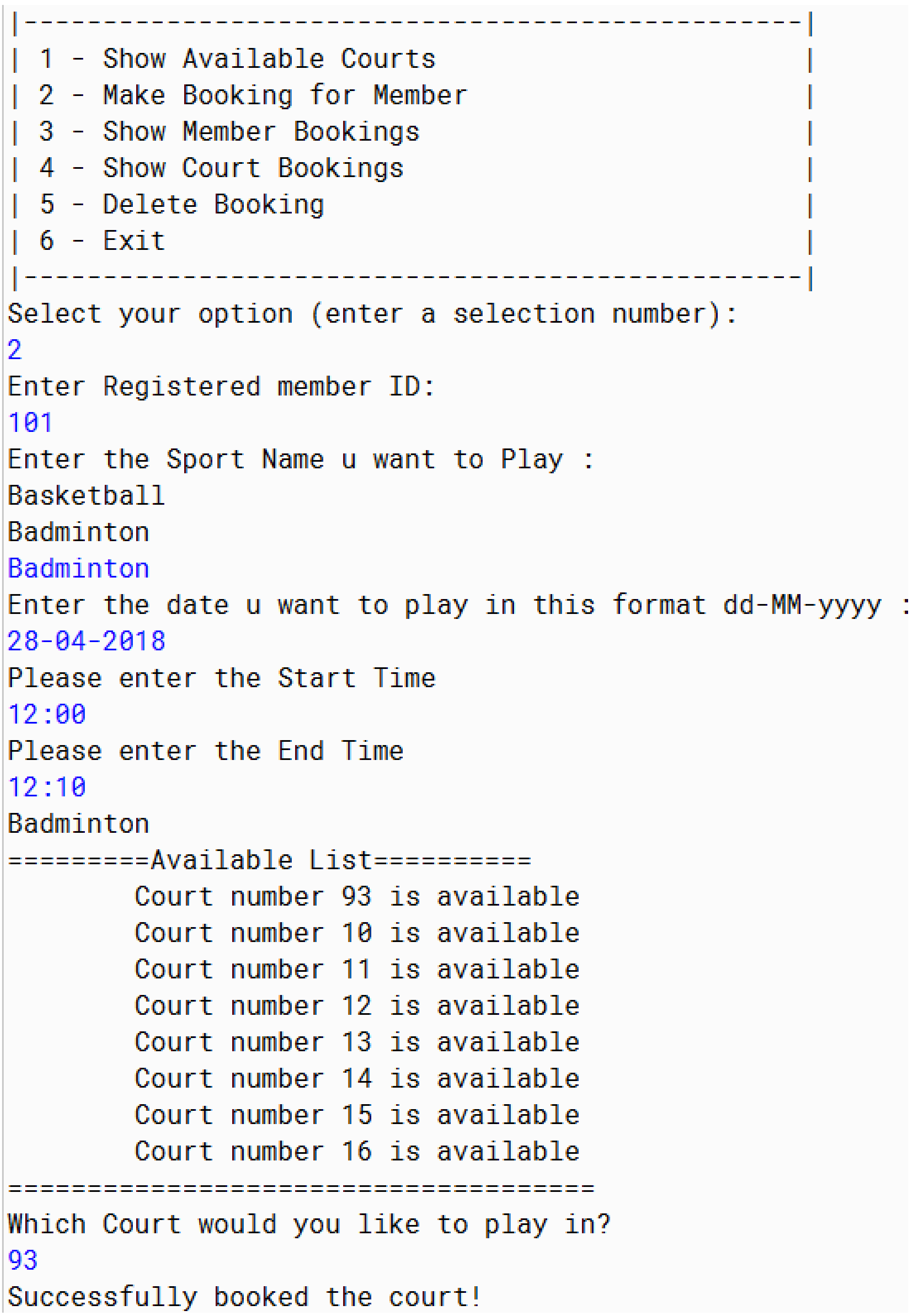
Output:

Output1:



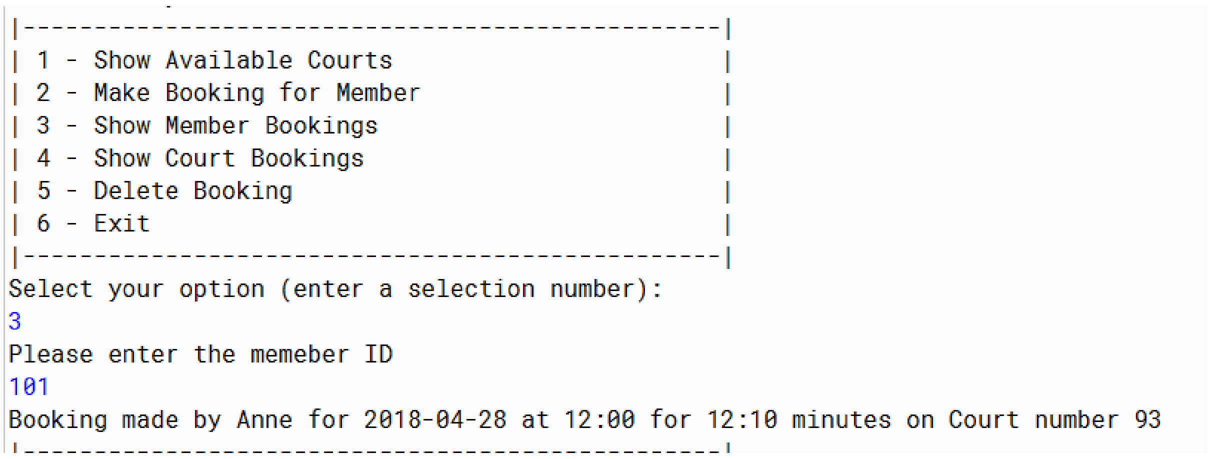
```
|-------------------------------------------------|
| 1 - Show Available Courts                       |
| 2 - Make Booking for Member                     |
| 3 - Show Member Bookings                        |
| 4 - Show Court Bookings                         |
| 5 - Delete Booking                              |
| 6 - Exit                                        |
|-------------------------------------------------|
Select your option (enter a selection number):
1
Enter the Sport Name u want to Play :
Basketball
Badminton
Basketball
Basketball
=========Available List==========
Court number :30
Court number :5
Court number :6
Court number :7
Court number :8
Court number :9
=====================================
```

Output2:



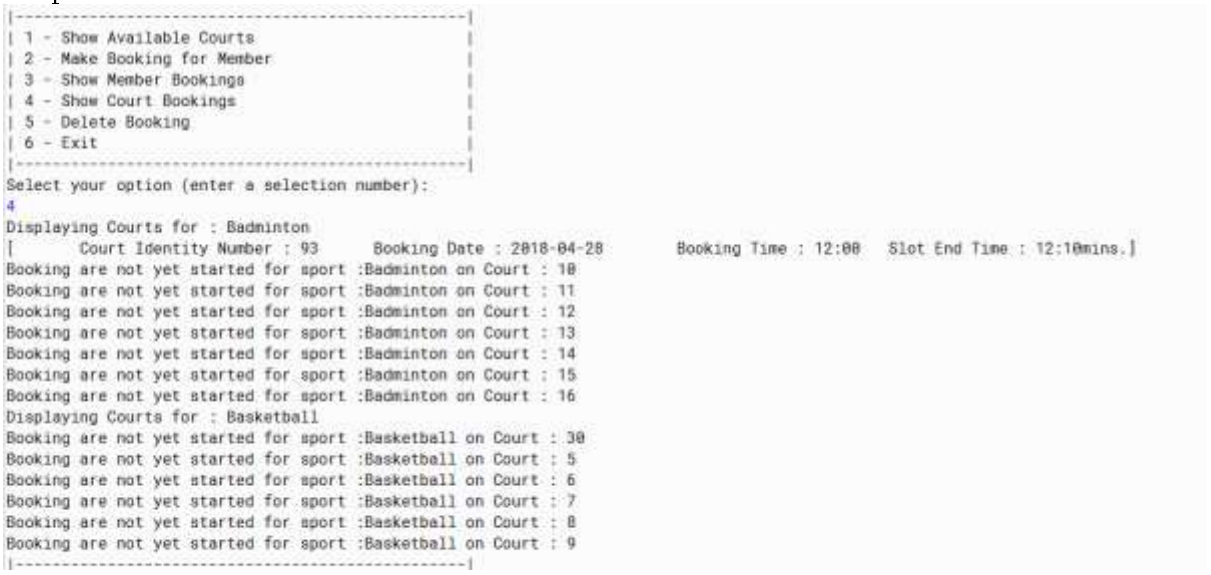
```
|-------------------------------------------------|
| 1 - Show Available Courts                       |
| 2 - Make Booking for Member                     |
| 3 - Show Member Bookings                        |
| 4 - Show Court Bookings                         |
| 5 - Delete Booking                              |
| 6 - Exit                                        |
|-------------------------------------------------|
Select your option (enter a selection number):
2
Enter Registered member ID:
101
Enter the Sport Name u want to Play :
Basketball
Badminton
Badminton
Enter the date u want to play in this format dd-MM-yyyy :
28-04-2018
Please enter the Start Time
12:00
Please enter the End Time
12:10
Badminton
=========Available List==========
        Court number 93 is available
        Court number 10 is available
        Court number 11 is available
        Court number 12 is available
        Court number 13 is available
        Court number 14 is available
        Court number 15 is available
        Court number 16 is available
=====================================
Which Court would you like to play in?
93
Successfully booked the court!
```

Output3:



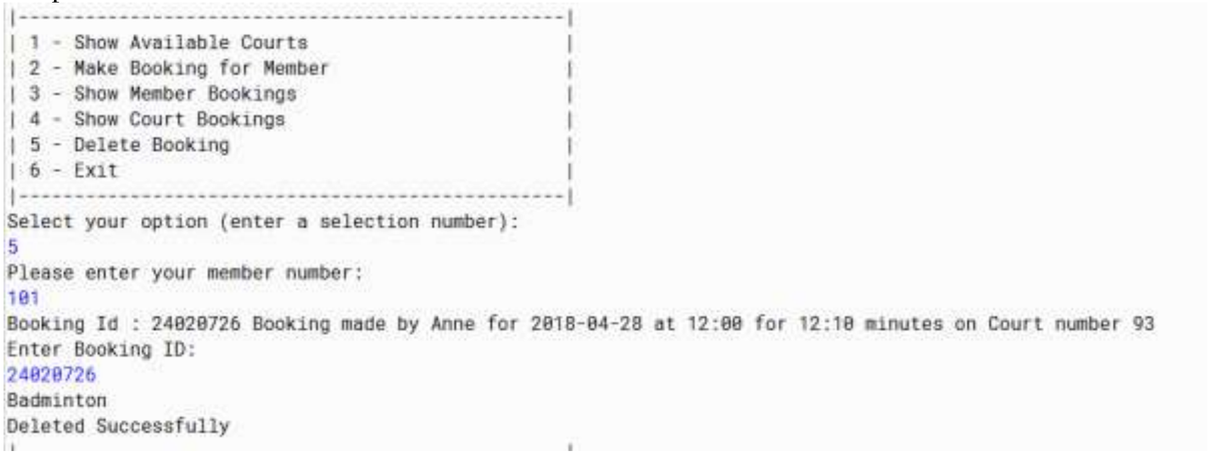
```
|--------------------------------------------------|
| 1 - Show Available Courts                        |
| 2 - Make Booking for Member                      |
| 3 - Show Member Bookings                         |
| 4 - Show Court Bookings                          |
| 5 - Delete Booking                               |
| 6 - Exit                                         |
|--------------------------------------------------|
Select your option (enter a selection number):
3
Please enter the memeber ID
101
Booking made by Anne for 2018-04-28 at 12:00 for 12:10 minutes on Court number 93
```

Output4:



```
|--------------------------------------------------|
| 1 - Show Available Courts                        |
| 2 - Make Booking for Member                      |
| 3 - Show Member Bookings                         |
| 4 - Show Court Bookings                          |
| 5 - Delete Booking                               |
| 6 - Exit                                         |
|--------------------------------------------------|
Select your option (enter a selection number):
4
Displaying Courts for : Badminton
[       Court Identity Number : 93      Booking Date : 2018-04-28       Booking Time : 12:00   Slot End Time : 12:10mins.]
Booking are not yet started for sport :Badminton on Court : 10
Booking are not yet started for sport :Badminton on Court : 11
Booking are not yet started for sport :Badminton on Court : 12
Booking are not yet started for sport :Badminton on Court : 13
Booking are not yet started for sport :Badminton on Court : 14
Booking are not yet started for sport :Badminton on Court : 15
Booking are not yet started for sport :Badminton on Court : 16
Displaying Courts for : Basketball
Booking are not yet started for sport :Basketball on Court : 30
Booking are not yet started for sport :Basketball on Court : 5
Booking are not yet started for sport :Basketball on Court : 6
Booking are not yet started for sport :Basketball on Court : 7
Booking are not yet started for sport :Basketball on Court : 8
Booking are not yet started for sport :Basketball on Court : 9
|--------------------------------------------------|
```

Output5:



```
|--------------------------------------------------|
| 1 - Show Available Courts                        |
| 2 - Make Booking for Member                      |
| 3 - Show Member Bookings                         |
| 4 - Show Court Bookings                          |
| 5 - Delete Booking                               |
| 6 - Exit                                         |
|--------------------------------------------------|
Select your option (enter a selection number):
5
Please enter your member number:
101
Booking Id : 24020726 Booking made by Anne for 2018-04-28 at 12:00 for 12:10 minutes on Court number 93
Enter Booking ID:
24020726
Badminton
Deleted Successfully
```