```
import java.io.*;
import java.util.Scanner;

/**
 * The main class
 */
public class AssEx3 {
        /**
         * The main method
         * @param args the arguments
         */
        public static void main(String[] args) {
                SportsCentreGUI display = new SportsCentreGUI();
                display.setVisible(true);
        }
}
```

```
/** Defines an object representing a single fitness class
 */
public class FitnessClass implements Comparable<FitnessClass> {
    private final int numOfWeeks = 5;
    private String classID;//class id
    private String className;//class name
    private String tutorName;// tutor name
    private int startTime;// start time
    private int[] atts;//attendances for the class

    /**constructor that takes id, name, tutor name, start time and an array of a
ttendances*/
    public FitnessClass(String id, String cName, String tName, int time, String[
] att)
    {
        //assigns values
        atts = new int[5];
        classID = id;
        className = cName;
        tutorName = tName;
        startTime = time;
        atts = new int[5];

        //the value att that is passed is 6 spaces as the first one is id
                for(int i = 0; i < numOfWeeks; i++)
                atts[i] = Integer.parseInt(att[i + 1]);//assigning starts with i
+1 to skip id spot
    }

    /**second constructor that takes one string for id, name , tutor, start time
     *  and an array of attendances*/
    public FitnessClass(String all, String[] att)
    {
        //assigns values
        atts = new int[5];
        String[] allSplit = all.split("[ ]+");
        classID = allSplit[0];
        className = allSplit[1];
        tutorName = allSplit[2];
        startTime =     Integer.parseInt(allSplit[3]);

        //the value att that is passed is 6 spaces as the first one is id
        for(int i = 0; i < 5; i++)
                atts[i] = Integer.parseInt(att[i + 1]);//assigning starts with i
+1 to skip id spot
    }

    //get and set for class ID
    public String getID(){return classID;}
    public void setID(String id){classID = id;}

    //get and set for class name
    public String getClassName(){return className;}
    public void setClassName(String name){className = name;}

  //get and set for tutor name
    public String getTutorName(){return tutorName;}
    public void setTutorName(String name){tutorName = name;}

  //get and set for start time
    public int getStartTime(){return startTime;}
    public void setStartTime(int time){startTime = time;}
  //get and set for attendances
    public int[] getAttendances(){return atts;}
    public void setAttendances(int[] att)
    {
        for(int i = 0; i < 5; i++)
                atts[i] = att[i];
    }


    /*returns the average of the class in question*/
    public double avg()
    {
        double ave = 0;
        double av = 0;

        for(int i = 0; i < numOfWeeks; i++)
                  av += atts[i];
        ave = av / numOfWeeks;
        return ave;
    }

    /*a method that returns a string for the attendance report*/

    public String toString()
    {
        String st;//to hold the string

        //add up all values with proper spacing
        st = String.format("%-15s \t %s  \t  %-10s  \t %02d %s %02d %s %02d %s %02d %s %02d \t
%05.2f", getID(), getClassName(),
        getTutorName(), getAttendances()[0], " ", getAttendances()[1], " ", getA
ttendances()[2],
        " ", getAttendances()[3]," ",  getAttendances()[4], avg());

        return st;
    }


    /**compares classes per averages*/
    public int compareTo(FitnessClass other) {

        if(this.avg() > other.avg())
                return -1; //return -1 if this object is greater

        else if(this.avg() == other.avg())
                return 0;//return 0 if equal

        else
                return 1;//return 1 if this object is less than other object
    }
}
```

```
import java.io.*;
import java.util.*;

/**
 * Maintains a list of Fitness Class objects
 * The list is initialized in order of start time
 * The methods allow objects to be added and deleted from the list
 * In addition an array can be returned in order of average attendance
 */
public class FitnessProgram {
   private FitnessClass[] classList;//to hold classes array sorted by start time
 and null when available
   private FitnessClass[] tmpC;//to hold classes as read from file
   private final int maxNum = 7;
   private int numOfClasses;//number of non null objects
   int numLines;//number of lines
   private Scanner scan;//to hold scanner object
   private String [][] attends;//to hold 2D array for all attendances values wit
h ID's

   /***constructor receives scanner with ClassesIn content and a 2D array with a
ll attendances values*/
   public FitnessProgram(Scanner sc, String[][] atts)
   {
           attends = atts.clone();//clone attendances parameter
           scan = sc; // assign scanner object parameter to our own scanner obje
ct

           tmpC = this.tempClassList();//fetch fitness class objects (unsorted)

           classList = new FitnessClass[maxNum];//create fitness class array wit
h 7 spaces

                        //loop through all spots from 9 to 15
                        for(int i = 0; i < maxNum; i ++)
                        {
                                /**bring value from method findClass, which take
s the unsorted array and a time slot*/
                                int found = findClass(tmpC, i + 9);

                                /**if found is greater than 0, assign to the sor
ted array*/
                                if(found >= 0)
                                        classList[i] = tmpC[found];

                                /**else, assign null*/
                                else
                                        classList[i] = null;
                        }
        }//end of constructor

   /**return sorted array of classes*/
   public FitnessClass[] getSchedule(){return classList;}

   /**returns max number of spots --> 7*/
   public int getMax(){return maxNum;}

   /**returns number of non null objects*/
   public int numOfClasses()
   {
            numOfClasses = 0;
            //loop through all spots
            for(int i = 0; i < classList.length; i++)
                    if(classList[i] != null)//when not null, increment variable
                            numOfClasses++;

            return numOfClasses;
    }


/**returns object at index x as required by assignment description
 * eventually, I didn’t have to use this method*/
    public FitnessClass classAt(int x)
    {
            return classList[x];
    }

  //return number of line in the scanner object passed from the GUI class
    public int numOfLines()
    {
            while(scan.hasNextLine())//while there’s a next line increment variab
le
                 {
                         numLines++;
                         scan.nextLine();
                 }
          return numLines;
    }

    /**creates unsorted list of classes as read from file*/
    public FitnessClass[] tempClassList()
    {
            FileReader red = null;
         try {
                 red = new FileReader("ClassesIn.txt");
         } catch (FileNotFoundException e) {
                 // TODO Auto−generated catch block
                 e.printStackTrace();
         }
                 Scanner in = new Scanner(red);//put it in a scanner object

                 tmpC = new FitnessClass[this.numOfLines()];//create a fitness cl
ass array as big the number of lines

                 //loop through all array
                 for(int i = 0; i < tmpC.length; i++)
                         {
                                 String record = in.nextLine();//read first line
                                 String[] id = new String[5];//to hold broken lin
e
                                 id = record.split("[ ]+");//break line
                                 String idx = id[0];//id is at first spot

                                 /**create a new fitness class by sending the lin
e from ClassesIn file
                                  * and the value returned from findID(attendance
s)*/
                                 tmpC[i] = new FitnessClass(record, findID(idx));
                         }

                 return tmpC;//return temporary array of unsorted classes
    }
    /**used by constructor to make sorted class array.
     * takes unsorted array and a time slot.
```

```
     * returns index of required class in the unsorted array if found
     * returns −1 if there’s no class in a particular spot so that the constructo
r can fill null in the place*/

    public int findClass(FitnessClass[] fc, int x)
    {
            int len = fc.length;//length of unsorted array
            for(int i = 0; i < len; i++)
            {
                    if(x == fc[i].getStartTime())//if found return index
                            return i;
            }
            return −1;//return −1 when not found
    }

    /**used by tempClassList method to make unsorted list of classes
     * takes a string (ID) and it returns the matching set of attendance values*/
    public String[] findID(String x)
    {
            int len = numOfLines();//number of line
            int i;
            for(i = 0; i < len; i++)
                    if(x.equals(attends[i][0]))//if id matches one of the lines
                            return attends[i]; //return the entire line with atte
ndance values

         return attends[i];//if not found, return anything as the program descrip
tion assumes it will find the id
    }

    /**used by atString method that makes the attendance report content
     * returns array of classes sorted by averages*/
    public FitnessClass[] sortClasses()
    {
          //array to hold non−null objects before sorting
            FitnessClass[] temp = new FitnessClass[numOfClasses()];

            //go through all program and fill non−null objects in temp array
            int t = 0;//to increment temp array
                 for(int c = 0; c < maxNum; c++)
                  if(classList[c] != null)
                         {
                         temp[t] = classList[c];//when not null fill temp and inc
rement
                         t++;
                         }

            Arrays.sort(temp);//sort temp array per averages
            return temp;
    }

    /**return string for attendance report*/
    public String atString()
    {
            String s = "";//to hold entire string

            String header = "id \t class \t tutor \t      attendances "
            + "\t  avg \n ==============================================================
====\n";//header string

            /**loop through all sorted classes and bring their strings*/
            for(int i = 0; i < numOfClasses(); i++)
            s += sortClasses()[i].toString() + "\n";

            //add up header string + main string + overall value in s string and
return it
            s = String.format("%s %s \n\t\t\t\t overall avg: %4.2f", header, s, avgAll());
            return s;
    }

    /**used by atString
     * returns overall average
     * */
    public double avgAll()
    {
            double d = 0.0;
            /**when not null, add up averages from all classes*/
            for(int i = 0; i < maxNum; i++)
                    if(classList[i] != null)
                            d += classList[i].avg();

            d /= numOfClasses();//calculate average

            return d;
    }

    /*returns index for first free (null) position in the program*/
    public int findFirstFree()
    {
            for(int i = 0; i < maxNum; i++)
                    if(classList[i] == null)
                            return i;//return first free spot

            return −1;//when program is full

    }

    /*checks if an ID already exists before adding or deleting a class*/
    public boolean ifExist(String id)
    {
            for(int i = 0; i < maxNum; i++)//loops through the list of all classe
s
            {
                  //if it finds a class with the same id it returns true
                    if(classList[i] != null)
                            if(id.equals(classList[i].getID()))
                                    return true;
            }
           return false;//returns false if ID is not found
    }

    /*adds new class with values from text fields at the first free spot in the p
rogram*/
    public void addClass(String id, String name, String tut)
    {
            String[] atts = {"ignored value", "0","0","0","0","0"};//initialize atten
dances of new class as 0's
            int time;
            int index = findFirstFree();//pulls first free spot index

            if(index == 0)//if first free spot is the first one set time at 9
                    time = 9;
            else//otherwise, start time of the new class if the start time of pre
vious class + 1
```

```
                    time = classList[index − 1].getStartTime() + 1;

            classList[index] = new FitnessClass(id, name, tut, time, atts);//adds
 new class to the first free spot on the program
  }

        /*deletes a class based on the ID user enters*/
        public void deleteClass(String id)
        {
                for(int i = 0; i < maxNum; i++)//loops through the list of all
classes
                  {
                        //if it finds a class with the same id it sets it to nu
ll
                          if(classList[i] != null)
                                  if(id.equals(classList[i].getID()))
                                  {
                                       classList[i] = null;
                                       break;
                                       }
                  }
        }//end of delete Class method

 }//end of class
```

```
import java.awt.*;
import javax.swing.*;

/**
 * Class to define window in which attendance report is displayed.
 */
public class ReportFrame extends JFrame {

        private JTextArea display;

        public ReportFrame()
        {
                this.setSize(510, 230);
                this.setTitle("Attendance Report");
                this.setLocation(100, 100);

                /*this is not to close the entire program upon closing this wind
ow*/
                this.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

                display = new JTextArea();//create JTextArea field

                add(display, BorderLayout.CENTER);//adds it to Center
        }

        public JTextArea getDisplay(){return display;}//returns display to write
 to − used by GUI class
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;

/**
 * Defines a GUI that displays details of a FitnessProgram object
 * and contains buttons enabling access to the required functionality.
 */
public class SportsCentreGUI extends JFrame implements ActionListener {

        /** GUI JButtons */
        private JButton closeButton, attendanceButton;
        private JButton addButton, deleteButton;

        /** GUI JTextFields */
        private JTextField idIn, classIn, tutorIn;

        /** Display of class timetable */
        private JTextArea display;

        /** Display of attendance information */
        private ReportFrame report;

        /** Names of input text files */
        private final String classesInFile = "ClassesIn.txt";
        private final String classesOutFile = "ClassesOut.txt";
        private final String attendancesFile = "AttendancesIn.txt";

        /**an object to hold the fitness program object we’ll create*/
        private FitnessProgram fp;



        /**
         * Constructor for AssEx3GUI class
         */
        public SportsCentreGUI() {
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setTitle("Boyd−Orr Sports Centre");
                setSize(730, 300);
                display = new JTextArea();
                display.setFont(new Font("Courier", Font.PLAIN, 14));
                add(display, BorderLayout.CENTER);
                layoutTop();
                layoutBottom();
                // more code needed here

                /**this is to read ClassesIn, feed it to FitnessProgram object*/
                FileReader reader = null;
                try {
                        reader = new FileReader(classesInFile);
                } catch (FileNotFoundException e) {
                        // TODO Auto−generated catch block
                        e.printStackTrace();
                }
                /**put file content in scanner object*/
                Scanner scan = new Scanner(reader);

                /**create fitness program object by passing the scanner object a
nd the attendances values*/
```

```
                fp = new FitnessProgram(scan, initAttendances());


                //closing reader object
                try {
                        reader.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                /**fire up display*/
                updateDisplay();

        }//end of constructor

        /**
         * Creates the FitnessProgram list ordered by start time
         * using data from the file ClassesIn.txt
         */
        public void initLadiesDay() {

                /**i eventually didn't have to use this method
                 *
                 * but it sets a fitness class array to the value of the fitness
 program time table*/

                FitnessClass[] sched = fp.getSchedule();
        }

        /**
         * Initializes the attendances using data
         * from the file AttendancesIn.txt
         * returns 2D string of all the value and passes them to the constructor
         * of fitness program object
         */
        public String[][] initAttendances() {

                /**reads the content of attendances file*/
            FileReader reader = null;
            try {
                         reader = new FileReader(attendancesFile);
                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }


            int num = lineCounter();//number of lines in the file
            Scanner scan = new Scanner(reader);//feeds reader content to a scann
er object
            String[] ats = new String[num];//holds each line before it is broken
 into string array
            String[][] atData = new String[num][6];//2D array with rows = number
 of lines && columns = number of values in each row

            /**loops through all lines in the file, and broken in to string arra
ys*/
            for (int i = 0; i < num; i++) {
                {
                        ats[i] = scan.nextLine();//go to next line
                        atData[i] = ats[i].split("[ ]+");//break line into 6 stri
```

```
ngs and put them in the rows of the 2D array
                }
                //now we have a 2D string array filled with the ID's and attenda
nce values for all classes/
                }

            /**this value is ready to be fed into the fitness program object con
structor*/
            return atData;
        }

        /**
         * Instantiates timetable display and adds it to GUI
         */
        public void updateDisplay() {

                FitnessClass[] fc = fp.getSchedule();//fetch time table from fit
ness program object
                String times = "";//to hold time slot for each class
                String classes = "";//to hold class names
                String tuts = "";//to hold tutors names
                for(int i = 0; i < 7; i++)//loop through all time slots(from 9 t
o 15)
                    {
                        int s = i + 9;//start time
                        int t = i + 9 + 1;//end time for each 1 hour long class
                        String ti = s + "-" + t;//to hold time slot, for example
 : 9-10

                        String time = String.format("%-13s", ti);//to make suffi
cient space between classes
                        times+=time;//add up all time slots


                        String clas;//to hold each individual class
                        String tut;//to hold each individual tutor name

                        //if object is null then class name is available and tut
or name is blank
                        if(fc[i] == null)
                                {clas = "Available";
                                tut = "";}

                        //else, fetch class name and tutor name
                        else{
                                clas = "" + fc[i].getClassName();
                                tut = "" + fc[i].getTutorName();
                                }

                        //add up fetched info into the strings for classes and t
utors
                        classes+=String.format("%-13s", clas);
                        tuts+=String.format("%-13s", tut);
                        }

                //set display text to the strings for all classes
                display.setText("" + times + "\n" + classes + "\n" + tuts);
        }

        /**
         * adds buttons to top of GUI
         */
        public void layoutTop() {
                JPanel top = new JPanel();
                closeButton = new JButton("Save and Exit");
                closeButton.addActionListener(this);
                top.add(closeButton);
                attendanceButton = new JButton("View Attendances");
                attendanceButton.addActionListener(this);
                top.add(attendanceButton);
                add(top, BorderLayout.NORTH);
        }

        /**
         * adds labels, text fields and buttons to bottom of GUI
         */
        public void layoutBottom() {
                // instantiate panel for bottom of display
                JPanel bottom = new JPanel(new GridLayout(3, 3));

                // add upper label, text field and button
                JLabel idLabel = new JLabel("Enter Class Id");
                bottom.add(idLabel);
                idIn = new JTextField();
                bottom.add(idIn);
                JPanel panel1 = new JPanel();
                addButton = new JButton("Add");
                addButton.addActionListener(this);
                panel1.add(addButton);
                bottom.add(panel1);

                // add middle label, text field and button
                JLabel nmeLabel = new JLabel("Enter Class Name");
                bottom.add(nmeLabel);
                classIn = new JTextField();
                bottom.add(classIn);
                JPanel panel2 = new JPanel();
                deleteButton = new JButton("Delete");
                deleteButton.addActionListener(this);
                panel2.add(deleteButton);
                bottom.add(panel2);

                // add lower label text field and button
                JLabel tutLabel = new JLabel("Enter Tutor Name");
                bottom.add(tutLabel);
                tutorIn = new JTextField();
                bottom.add(tutorIn);

                add(bottom, BorderLayout.SOUTH);
        }

        /**
         * Processes adding a class
         */
        public void processAdding() {

                String id = idIn.getText();//get text from id field
                String name = classIn.getText();//get text from class name
                String tut = tutorIn.getText();//get text from tutor name
                int firstFree = fp.findFirstFree();//fetch first available time
slot


                /**if program is full, inform user and clear fields*/
```

```
                if(firstFree == -1)
                {
                        JOptionPane.showMessageDialog(null, "Sorry program is full", "
Error message",
                        JOptionPane.ERROR_MESSAGE);

                        //clear fields
                        idIn.setText("");
                        classIn.setText("");
                        tutorIn.setText("");
                }

                /**check if class already exist*/
                else if(fp.ifExist(id) == true)
                {
                        JOptionPane.showMessageDialog(null, "Class already exist", "Err
or message",
                        JOptionPane.ERROR_MESSAGE);

                        //clear fields
                        idIn.setText("");
                        classIn.setText("");
                        tutorIn.setText("");
                }

                /**if one of the fields is empty*/
                else if(id.isEmpty() || name.isEmpty() || tut.isEmpty())

                        JOptionPane.showMessageDialog(null, "You must fill all fiels", "
Error message",
                        JOptionPane.ERROR_MESSAGE);

                /**add class to time table*/
                else
                {
                        fp.addClass(id, name, tut);//add new class

                        JOptionPane.showMessageDialog(null, "New class has been added"
, "Confirmed",
                        JOptionPane.OK_OPTION);//show confirmation message

                        //clears fields
                        idIn.setText("");
                        classIn.setText("");
                        tutorIn.setText("");

                        updateDisplay();//update time table
                }

        }

        /**
         * Processes deleting a class
         */
        public void processDeletion() {
                String id = idIn.getText();//get text from id field

                /**checks if empty*/
                if(id.isEmpty())
                        JOptionPane.showMessageDialog(null, "Please enter the ID of the cl
ass you want deleted", "Error message",
                        JOptionPane.ERROR_MESSAGE);
                /**check if class exist*/
                else if(!fp.ifExist(id))
                        {
                        JOptionPane.showMessageDialog(null, "Class does not exist", "Er
ror message",
                        JOptionPane.ERROR_MESSAGE);

                        //clears fields
                        idIn.setText("");
                        classIn.setText("");
                        tutorIn.setText("");
                        }
                /**delete class*/
                else
                        {
                        fp.deleteClass(id);
                        JOptionPane.showMessageDialog(null, "Class deleted", "Confirm
ed",
                        JOptionPane.OK_OPTION);//confirmation message

                        //clears fields
                        idIn.setText("");
                        classIn.setText("");
                        tutorIn.setText("");

                        updateDisplay();//update display
                        }
        }

        /**
         * Instantiates a new window and displays the attendance report
         */
        public void displayReport() {
            report = new ReportFrame();//create Report Frame object
            JTextArea dis;//object to hold JTextArea
            dis = report.getDisplay();//assign display to our object

            /**set display to data brought from method in fitness program object
*/
                dis.setText(fp.atString());

            /**make report visible*/
            report.setVisible(true);
        }

        /**
         * Writes lines to file representing class name,
         * tutor and start time and then exits from the program
         */
        public void processSaveAndClose() {
                /**open a file to save data*/
                PrintWriter writer = null;
            try {
                         writer = new PrintWriter(classesOutFile);
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }


            String s = "";//to hold data
```

```
            FitnessClass[] f = fp.getSchedule();//fetch schedule from fitness pr
ogram object

            /**loop through all time slots and bring data from non null objects*
/
            for(int i = 0; i < fp.getMax(); i++)
                if(f[i] != null)//while not null add up data to s string
                        s += f[i].getID() + " " + f[i].getClassName() + " " + f[i
].getTutorName()
                        + " " + f[i].getStartTime() + "\n";

            /**write data to file*/
            writer.println(s);

            //show confirmation message
            JOptionPane.showMessageDialog(null, "Data saved to file", "Confirmed", JOpti
onPane.OK_OPTION);

            //close print writer object
            writer.close();

            //exit system
            System.exit(0);
                }

        /**counts lines in the classesIn file*/
        public int lineCounter()
        {
                int numLines = 0;
                /**read file*/
                FileReader read = null;
                try {
                        read = new FileReader(classesInFile);
                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                /**put file in a scanner object*/
                Scanner scan = new Scanner(read);

                //while there's a next line
                while(scan.hasNextLine())
                        {
                        numLines++;//increment
                        scan.nextLine();//go to next line
                        }
                return numLines;//return number of lines
        }

        /**
         * Process button clicks.
         * @param ae the ActionEvent
         */
        public void actionPerformed(ActionEvent ae) {

                /**if user clicks attendance button, display report*/
                if(ae.getSource() == attendanceButton)
                        displayReport();

                /**if user clicks add button, process adding*/
                else if(ae.getSource() == addButton)
```

```
                        processAdding();

                /**if user clicks delete button, process deletion*/
                else if(ae.getSource() == deleteButton)
                        processDeletion();

                /**if user clicks close button, process save and close*/
                else if(ae.getSource() == closeButton)
                        processSaveAndClose();

        }
}
```