

Jan 12, 15 9:04

AssEx3.java

Page 1/1

```

/**
 * 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);
    }
}

```

Jan 12, 15 9:04

FitnessClass.java

Page 1/4

```

import java.util.Arrays;

/** Defines an object representing a single fitness class
 */
public class FitnessClass implements Comparable<FitnessClass>
{
    //Class constant for number of weeks to record attendances
    private static final int NUM_WEEKS = 5;

    //More instance variables
    private String classId, className, tutorName;
    private int startTime;

    //array to store attendances for each class
    private int[] attendances;

    /**default constructor for FitnessProgram array instantiation
     *
     */
    public FitnessClass()
    {
        //initialise instance variables
        this(""); /*I saw this solution in tutorial 8 solutions
        number 2, for default constructor in bank account class*/
    }

    /**non-default Constructor
     *
     * @param classesInString parameter from FitnessProgram addClass method
     */
    public FitnessClass(String classesIn)
    {
        //Need to read data from String, use split (space-delimited) method and process each token
        String[] classesTokens = classesIn.split(" ");

        /*I used this classes 'set' methods in the constructor here, not sure if its bad practice....
        * I could have equally just initialised the instance variables using assignment operators
        * (.equals for strings and = for the integer), however, I think I saw this in one of the tutorial or lecture solutions
        */
        setClassId(classesTokens[0]);
        setClassName(classesTokens[1]);
        setTutorName(classesTokens[2]);
        setStartTime(Integer.parseInt(classesTokens[3])); //Is using 'magic numbers' acceptable here?

        //Instantiate the attendance array here, ready to store attendance
        * data
        */
        setAttendances(new int[NUM_WEEKS]);
    }

    /**Compare average (arithmetic mean) attendances for sorting
     * @param other is a fitnessclass object
     * @return an integer depending on the outcome of comparison
     */
    public int compareTo(FitnessClass other)
    {
        try
        {
            if (this.calculateMeanAttendance() < other.calculateMeanAttendance())

```

Jan 12, 15 9:04	<b>FitnessClass.java</b>	Page 2/4
-----------------	--------------------------	----------

```

ce()
{
    return 1;
}
else if (this.calculateMeanAttendance() == other.calculateMeanAt
tendance())
{
    return 0;
}
else
{
    return -1;
}
}
catch (NullPointerException ct)
{
    System.err.println("Not enough classes bud..." + ct);
    return 0;
}
}

/**accessor and modifier methods for each instance variable
 *
 * @return classId string
 */
public String getClassId()
{
    return classId;
}

/**Modifier for classId
 *
 * @param classIdIn string with classId
 */
public void setClassId(String classIdIn)
{
    this.classId = classIdIn;
}

/**Accessor for ClassName
 *
 * @return className string
 */
public String getClassName()
{
    return className;
}

/**Modifier for class name
 *
 * @param classNameIn string with class name
 */
public void setClassName(String classNameIn)
{
    this.className = classNameIn;
}

/**Accessor for tutor name
 *
 * @return string with tutor name
 */
public String getTutorName()
{
    return tutorName;
}

/**Modifier for tutor name
 *
 * @param tutorNameIn string with tutor name
 */
public void setTutorName(String tutorNameIn)

```

Jan 12, 15 9:04	<b>FitnessClass.java</b>	Page 3/4
-----------------	--------------------------	----------

```

{
    this.tutorName = tutorNameIn;
}

/**Accessor for class start time#
 *
 * @return startTime integer between 9 and 15 inclusive
 */
public int getStartTime()
{
    return startTime;
}

/**Modifier for class start time
 *
 * @param startTimeIn integer between 9 and 15 inclusive
 */
public void setStartTime(int startTimeIn)
{
    this.startTime = startTimeIn;
}

/**Return a string containing list of attendance integers
 *
 * @return attendanceList string with five integers separated by spaces
 */
public String getAttendances()
{
    String attendanceList = "";

    ///Loop through attendance array
    for (int i = 0; i < attendances.length; i++)
    {
        //Cast, format and concatenate attendance integers to string
        attendanceList += String.format("%-4s", "" + attendances
[i]);
    }

    //Return string
    return attendanceList;
}

/**Modifier to set attendance data
 *
 * @param attendanceArray array of integers representing attendances
 */
public void setAttendances(int[] attendanceArray)
{
    this.attendances = attendanceArray;
}

/**Accessor to return the average(mean) attendance for this object
 *
 * @return double representing mean attendance
 */
public double getMeanAttendance()
{
    double mean = calculateMeanAttendance();
    return mean;
}

/**helper method to calculate mean attendance
 * for comparisons of mean attendance
 *
 * @return a double representing the mean attendance
 */
private double calculateMeanAttendance()
{
    //Generalised for loop to sum integers from attendance array
    int attendanceTotal = 0;

```

Jan 12, 15 9:04

## FitnessClass.java

Page 4/4

```

        //From beginning to end of array
        for (int meanIndex: attendances)
        {
            //Sum the integers in the attendance array
            attendanceTotal = attendanceTotal + meanIndex;//[meanInd
ex];
        }

        /*Calculate and return the mean of the elements in the attendanc
e array
        *we know these will always be filled so no error checking here
        */
        return (double) attendanceTotal/attendances.length;
    }
}

```

Jan 12, 15 9:04

## FitnessProgram.java

Page 1/6

```

import java.util.*;

/**
 * Maintains a list of Fitness Class objects
 * The list is initialised 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
{
    //Instance variables

    /*Class constants for maximum number of classes ie
    * max number of FitnessClass objects in array
    * to subtract 9 from class start time and
    * for maximum number of weeks for attendance data
    */
    private static final int MAX_CLASSES = 7;
    private static final int EARLIEST_TIME = 9;
    private static final int NUM_WEEKS = 5;

    //Arrays (list) of FitnessClass objects
    private FitnessClass [] classes;//primary array
    private FitnessClass[] sortedMeanAttendances = new FitnessClass[MAX_CLASSES];

    /*Integer to maintain count of FitnessClass objects in classes array*/
    private int numFitnessObjects = 0;

    /**Default constructor, i.e. no parameters as input
    * instantiates the array of FitnessClass objects
    */
    public FitnessProgram()
    {
        //Instantiate FitnessClass array to store list of classes
        classes = new FitnessClass[MAX_CLASSES];
    }

    /**Accessor method to find the first free timeslot
    * for a class to be added, or -1 if not found
    * @return integer freeTime
    */
    public int findAvailableTime()
    {
        int freeTime = 0;
        boolean slotFound = false;

        //Loop to search for null or empty entries in FitnessClass array
        while (!slotFound && freeTime <= classes.length)
        {
            if (classes[freeTime] == null)
            {
                slotFound = true;
            }
            else
            {
                freeTime++;
            }
        }

        if (slotFound)//there is an empty slot, return index plus 9 (this equals
the start time free)
        {
            freeTime += EARLIEST_TIME;
        }

        return freeTime;
    }
}

```

Jan 12, 15 9:04

## FitnessProgram.java

Page 2/6

```

/**Modifier method to add FitnessClass object to array, needs to
 * take FitnessClass string and start time as parameters (from GUI)
 * to pass to the fitnessclass object and to determine which position in the
 * FitnessClass array to store it in, ordered by start time
 * @param integer startTime between 9 and 15
 * @param classDescription string containing class data
 */
public void addClass(int startTime, String classDescription)
{
    //Instantiate FitnessClass object and store in array at index according
to start time
    classes[startTime-EARLIEST_TIME] = new FitnessClass(classDescription);

    //Increment object counter
    numFitnessObjects++;
}

/**Method to return the number of classes in list
 *
 * @return integer with count of number of classes
 */
public int getNumClasses()
{
    return numFitnessObjects;
}

/*Method to return a fitnessclass object with that start time
 * input parameter is an integer between 9 and 15
 */
public FitnessClass getClass(int classTime)
{
    return classes[classTime-EARLIEST_TIME];
}

/**This method sets array entry to null, for deletion of that class
 *
 * @param string with only a class Id
 */
public void setClassNull(String classId)
{
    //Search class list to find index of class with matching id
    int deleteClassIndex = searchClassId(classId);

    //set array entry to null
    classes[deleteClassIndex] = null;

    //Increment class counter by -1
    numFitnessObjects--;
}

/**Method to pass attendance data to a fitnessclass object
 * @param a string with a class Id and five integers
 * with a space delimiter between each one
 */
public void addAttendances(String data)
{
    try
    {
        //Get the array index of the matching class
        String classId = data.substring(0, 3);
        int classIndex = searchClassId(classId);

        //Create integer array from integers in string
        int [] attendances = new int[NUM_WEEKS];

        //Split string using alphabetic characters and spaces as delimit
ers (copied from lecture 11 by David Manlove)

```

Jan 12, 15 9:04

## FitnessProgram.java

Page 3/6

```

        String [] attendanceIntegers = data.split("[a-zA-Z]+");

        //Loop through array and set attendance integers
        for (int attendanceIndex = 0; attendanceIndex < NUM_WEEKS; atten
danceIndex++)
        {
            //Discard first integer in string array as this is from
the classID.
            attendances[attendanceIndex] = Integer.parseInt(attendan
ceIntegers[(attendanceIndex + 2)]); //+2 to skip first token
        }

        //Pass integer array to the appropriate fitnessclass in the fitnessclass
array
        classes[classIndex].setAttendances(attendances);
    }

    catch (ArrayIndexOutOfBoundsException ai)
    {
        System.err.println("Something was wrong with the input: " + ai);
    }
}

/**Search through the array to find the index of the fitness class
 *
 * @param a string with a class id
 * @return either the class array index or -1 if not found
 */
public int searchClassId(String id)
{
    //Search the class ids
    boolean idFound = false;
    int classIndex = 0;
    while (!idFound && classIndex < MAX_CLASSES)
    {
        //Order of nested if statements important here to avoid nullpoin
terexceptions.
        if (classes[classIndex] == null)
        {
            //Increment array index counter if array entry i
s null
            classIndex++;
        }
        else if (classes[classIndex].getClassId().equals(id)) //Test for
matching class id
        {
            //Set boolean to break out of while loop
            idFound = true;
        }
        else
        {
            //Increment if the classId doesnt match
            classIndex++;
        }
    }

    //Test to see if id was found
    if (idFound)
    {
        //Return the FitnessClass with the matching id
        return classIndex;
    }
    else
    {
        return -1;
    }
}

```

Jan 12, 15 9:04

FitnessProgram.java

Page 4/6

```

/**Method to sort class list by mean attendance
 *
 */
private void sortClasses()
{
    //Instantiate/reset sorting array
    sortedMeanAttendances = new FitnessClass[MAX_CLASSES];

    //Integer to count null entries (i.e. missing classes in 'classes' array
    of fitnessclass objects
    int nullClassCount = 0;

    //Integer to keep track of index position in array to be sorted on atten
dance means
    int sortArrayCounter = 0;

    //Loop to go through each array index in the 'classes' array
    for (int i = 0; i < classes.length; i++)
    {
        //Test if the array index is empty
        if (classes[i] == null)
        {
            nullClassCount++; //increment the null entry counter
        }

        else
        {
            sortedMeanAttendances[sortArrayCounter] = classes[i]; //
otherwise copy the first non-null entry to the array for sorting
            /*Increment the counter that is keeping track of where t
he next empty position is in the sorting array,
            * this ensures null entries always end up at the end of
this array
            */
            sortArrayCounter++;
        }
    }

    //Create a new array with a length shorter by the number of null entries
(nullClassCount) than the maximum (7)
    FitnessClass[] smallArray = new FitnessClass[MAX_CLASSES - nullClassCoun
t];

    //Copy the array entries for sorting across to the new shorter array, st
arting at the beginning, ensuring null entries are removed
    System.arraycopy(sortedMeanAttendances, 0, smallArray, 0, MAX_CLASSES-nu
llClassCount);

    //shallow copy the old array to the smaller one, smaller array disappear
s (pointer points to nothing)
    sortedMeanAttendances = smallArray;

    //Sort attendance array by non-increasing order of mean attendance (uses
compareTo method in fitnessclass)
    Arrays.sort(sortedMeanAttendances);
}

/**Method to return formatted string with attendance report
 *
 * @return String with attendance report
 */
public String getMeanAttendanceReport()
{
    //Create sorted attendance array
    sortClasses();

```

Jan 12, 15 9:04

FitnessProgram.java

Page 5/6

```

/*create formatted string with fitnessclass objects
 * and mean attendances for reportframe window
 */
String attendanceReport = " " + String.format("%-5s %-16s %-16s %14s %29s %n
",
        "Id", "Class", "Tutor", "Attendances", "Average Attendance");
attendanceReport += "=====
+ "===== " + String.format("%n")
; //First part of report, this will not change

//Local variables for creating attendance string
//mean for each fitnessclass
double classMean = 0.0;
//Store sum of means
double meanSum = 0.0;
//store overall mean
double overallMean = 0.0;
//Strings to store fitnessclass instance variables
String idString = "";
String classNameString = "";
String tutorNameString = "";
String attendanceList = "";
String attendanceAverage = "";

//Loop to create attendance report string
for (int i = 0; i < sortedMeanAttendances.length; i++)
{
    //Get and store values from fitnessclass objects
    idString = sortedMeanAttendances[i].getClassId();
    classNameString = sortedMeanAttendances[i].getClassName();
    tutorNameString = sortedMeanAttendances[i].getTutorName();
    attendanceList = sortedMeanAttendances[i].getAttendances();
    classMean = sortedMeanAttendances[i].getMeanAttendance();
    attendanceAverage = String.format("%.2f", classMean);

    //Store sum of attendances
    meanSum += classMean;

    //Concatenate local variables to report string
    attendanceReport += " " + String.format("%-5s %-16s %-16s %-10s %15s
%n", idString, classNameString,
        tutorNameString, attendanceList, attendanceAvera
ge);
}

//calculate overall mean attendance
overallMean = meanSum/numFitnessObjects;

//Add overall mean to report string
attendanceReport += String.format("%n %65s %4.2f", "Overall Average: ", overa
llMean);
return attendanceReport;
}

/**Helper method to return formatted string with each class instance variabl
e
 *
 * @return string for writing to classesout text file
 */
public String getClassesOutFile()
{
    //Instantiate local string to store formatted report

```

Jan 12, 15 9:04

**FitnessProgram.java**

Page 6/6

```

String classVariables = "";

//Loop to build string for report
for (int outIndex = 0; outIndex < classes.length; outIndex++)
{
    if (classes[outIndex] != null)//obtain variables only from exist
ing classes
    {
        classVariables += " " + classes[outIndex].getClassId() +
" " + classes[outIndex].getClassName() + " " +
        classes[outIndex].getTutorName() + " " + classes
[outIndex].getStartTime() + String.format("%n");
    }
}

return classVariables;
}
}

```

Jan 12, 15 9:04

**ReportFrame.java**

Page 1/2

```

import java.awt.*;

import javax.swing.*;

/**
 * Class to define window in which attendance report is displayed.
 */
public class ReportFrame extends JFrame
{
    //Instance variables
    private FitnessProgram attendanceList;
    private JTextArea attendanceReport;
    private JFrame reportFrame;

    //Constants for JTextArea and JFrame sizes
    private final int FRAME_WIDTH = 750;
    private final int FRAME_HEIGHT = 350;
    private final int ROWS = 100;
    private final int COLUMNS = 300;

    /**Constructor for attendance report
     *
     * @param classList is a fitnessclass object
     */
    public ReportFrame(FitnessProgram classList)
    {
        //Initialise fitnessprogram object
        attendanceList = classList;

        //Layout JTextArea
        layoutAttendanceWindow();
        formatAttendanceData();
    }

    /**Method to layout JTextArea and display attendance report
     *
     */
    private void layoutAttendanceWindow()
    {
        //Instantiate JFrame to contain JTextArea
        reportFrame = new JFrame();
        reportFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);

        //Set location to prevent obscuring GUI
        reportFrame.setLocation(450, 300);
        reportFrame.setVisible(true);
        reportFrame.setTitle("Attendance Report – all arithmetic means");

        //Set close behaviour (just remove window, dont terminate progra
m)
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        //Instantiate JTextArea for attendance report
        attendanceReport = new JTextArea(ROWS, COLUMNS);

        //Use this font for consistency and correct formatting (Monospac
ed font)
        attendanceReport.setFont(new Font("Courier", Font.PLAIN, 14));

        //Prevent user from editing attendance window
        attendanceReport.setEditable(false);

        //Add JTextArea to JFrame
        reportFrame.add(attendanceReport, BorderLayout.CENTER);
    }
}

```

Jan 12, 15 9:04

**ReportFrame.java**

Page 2/2

```

    }

    /**method to get attendance report
     *
     */
    private void formatAttendanceData()
    {
        //FitnessProgram object contains method to return formatted attendance string
        attendanceReport.setText(attendanceList.getMeanAttendanceReport());
    }
}

```

Jan 12, 15 9:04

**SportsCentreGUI.java**

Page 1/8

```

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; //domt know why this warning is here...

    //Declare FitnessProgram object
    private FitnessProgram classList;

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

    //Maximum number of timeslots
    private final static int MAX_TIME_SLOTS = 7;
    //Earliest possible start time
    private final static int CLASSES_OPEN = 9;
    //Number of rows in GUI timetable display
    private final static int TIMETABLEROWS = 3;

    /**
     * Constructor for AssEx3GUI class
     */
    public SportsCentreGUI() {
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setTitle("Boyd-Orr Sports Centre");
        setSize(700, 300);
        display = new JTextArea(TIMETABLEROWS, MAX_TIME_SLOTS);
        display.setFont(new Font("Courier", Font.PLAIN, 14));
        add(display, BorderLayout.CENTER);
        layoutTop();
        layoutBottom();

        //Call method to create list of classes (FitnessProgram object)
        initLadiesDay();

        //Call method to add in attendances
        initAttendances();

        //Call method to format and display class timetable information
    }
}

```

Jan 12, 15 9:04

**SportsCentreGUI.java**

Page 2/8

```

        updateDisplay();
    }

    /**
     * Creates the FitnessProgram list ordered by start time
     * using data from the file ClassesIn.txt
     */
    public void initLadiesDay()
    {
        //Instantiate FitnessProgram object
        classList = new FitnessProgram();

        //Declare filereader to open classesIn file
        FileReader readClassesData = null;

        //Declare scanner to read strings from file
        Scanner classesInStrings;

        //This try-catch solution taken from tutorial 10 solutions and lectures
        try
        {
            try
            {
                //Instantiate filereader for classesIn file
                readClassesData = new FileReader(classesInFile);

                //Instantiate scanner for classesIn file
                classesInStrings = new Scanner(readClassesData);

                //While loop to read in all lines
                while (classesInStrings.hasNextLine())
                {
                    /*read in the id, class name, tutor name
                    s and start time as a single string,
                    and the start time as an integer, and then
                    en pass these as parameters to the addClass method in classList*/

                    //String to store addClass method parameters.
                    String classDesc = classesInStrings.nextLine();

                    //Find and store the class start time, as this
                    s this will always be the last integer on each line
                    int startTime = Integer.parseInt(classDesc.substring((classDesc.length()-2), classDesc.length()).trim());

                    //Call addClass method from fitnessprogram object to add a new fitnessclass object to the list (array of FitnessClass objects)
                    classList.addClass(startTime, classDesc);
                }
            }
            finally
            {
                //Test to see if filereader was initialised.
                if (readClassesData != null)
                {
                    //Close file
                    readClassesData.close();
                }
            }
        }
    }

```

Jan 12, 15 9:04

**SportsCentreGUI.java**

Page 3/8

```

        //Catch file missing exception
        catch (FileNotFoundException i)
        {
            System.err.println("File not found....well done.." + i);
            System.exit(1);
        }

        //Exception for closing file errors
        catch (IOException o)
        {
            System.err.println("Somehow, could not read file, this is in GUI class: "
+ o);
        }

    }

    /**
     * Initialises the attendances using data
     * from the file AttendancesIn.txt
     */
    public void initAttendances()
    {
        //Declare local filereader and scanner for AttendancesIn file
        FileReader readAttendanceFile = null;
        Scanner attendanceInString;

        try
        {
            try
            {
                //Instantiate filereader and scanner objects
                readAttendanceFile = new FileReader(attendancesFile);

                attendanceInString = new Scanner(readAttendanceFile);

                //Loop through each line
                while (attendanceInString.hasNextLine())
                {
                    //Read in line from attendances file and
                    pass id to fitness program
                    classList.addAttendances(attendanceInString.nextLine());
                }
            }
            finally
            {
                //Test to see if filereader was initialised.
                if (readAttendanceFile != null)
                {
                    //Close attendancesIn file
                    readAttendanceFile.close();
                }
            }
        }
        catch (FileNotFoundException g)
        {
            System.err.println("Attendance file not found...where is it?: " + g);
            System.exit(1);
        }
        catch (IOException p)
        {
            System.err.println("Eclipse made me catch this exception, as it wouldnt let me put in "
+ "a 'finally' block without loads of jiggery pokery, nonsense: " + p);
            System.exit(1);
        }
    }

```



Jan 12, 15 9:04

## SportsCentreGUI.java

Page 4/8

```

    }

    /**
     * Instantiates timetable display and adds it to GUI
     */
    public void updateDisplay()
    {
        /*add timetable to display JTextArea
         * For loop to cycle through class and tutor names and start time
s*/
        String className = "";
        String tutorName = "";
        String timeSlot = "";

        //Loop seven times
        for (int classIndex = 0; classIndex < MAX_TIME_SLOTS; classIndex
++)
        {
            /*Test to determine whether timeslot has a class(i.e. th
e array contains an object with that start time)
            * we need to add 9 to the index as the getClass method i
n fitnessclass subtracts 9 in order to
            * implement other functionality*/

            //Iteratively build the timeslot string (easier to forma
t this way)
            timeSlot += String.format("%-12s", (classIndex + CLASSES
_OPEN) + "-" + (classIndex + 1 + CLASSES_OPEN));

            //Iteratively concatenate fitnessclass instance variable
values, formatted to be left justified with a fieldwidth of 12
            if (classList.getClass(classIndex + CLASSES_OPEN) == nul
l)
            {
                className += String.format("%-12s", "Available");
                tutorName += String.format("%-12s", "");
            }
            else //obtain and store instance variable values from Fi
tnessClass objects in array
            {
                className += String.format("%-12s", classList.ge
tClass(classIndex + CLASSES_OPEN).getClassName());
                tutorName += String.format("%-12s", classList.get
Class(classIndex + CLASSES_OPEN).getTutorName());
            }

            //Add platform-independent new line operators to each string
            className += String.format("%n");
            tutorName += String.format("%n");
            timeSlot += String.format("%n");

            //Add strings to the GUI
            display.setText(timeSlot + className + tutorName);
        }

        /**
         * 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);

```

Jan 12, 15 9:04

## SportsCentreGUI.java

Page 5/8

```

        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 panell = new JPanel();
        addButton = new JButton("Add");
        addButton.addActionListener(this);
        panell.add(addButton);
        bottom.add(panell);

        // 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()
    {
        //Check that all textfields have data (no validation of input fo
rmat as per specification)
        if (idIn.getText().isEmpty() || classIn.getText().isEmpty() || t
utorIn.getText().isEmpty() || idIn.getText().length() > 3)
        {
            JOptionPane.showMessageDialog(null, "One or more of the required
inputs is missing or the classId is incorrect, try again", "Error", JOptionPane.ERROR_MESSAGE);
            clearTextFields();
        }
        //Use helper methods to check if class id already exists in arra
y
        else if (checkId())//boolean method
        {
            //if method returns true, then id already exists
            , display a warning
            JOptionPane.showMessageDialog(null, "A class with tha
t ID already exists.\n"
            + "Please check carefully and try again, if you

```

```

Jan 12, 15 9:04      SportsCentreGUI.java      Page 6/8

wish", "Error", JOptionPane.ERROR_MESSAGE);
        clearTextFields();
    }

    /**if class id doesnt already exist we can proceed to add
ing class, read in texfields,
    * create string (class time using findAvailableTime met
hod from fitnessprogram,classId,
    * name and tutor name from textfields) and pass as para
meter to add class method, plus
    * update all necessary arrays and displays
    */
    else
    {
        //Get first available timeslot
        int addStartTime = classList.findAvailableTime();

;
        //Create string in format required for addClass
method
        String classNames = "" + idIn.getText().trim() +
            + "" + tutorIn.getText().trim()

        //Invoke addClass method passing parameters from
textfield inputs
        classList.addClass(addStartTime, classNames);

        //Create attendance string for adding in attenda
nces of this class (all set to 0)
        String newClassAttendance = "" + idIn.getText().
trim() + "00000";

        //Set new class attendances passing attendance s
tring as a paramater
        classList.addAttendances(newClassAttendance);
        clearTextFields();

        //Display new timetable on GUI
        updateDisplay();
    }

}

/**
 * Processes deleting a class
 */
public void processDeletion()
{
    //Test if id matches an id in our list
    if (checkId())//id matches the list
    {
        //Delete the FitnessClass object from the array
        classList.setClassNull(idIn.getText().trim());

        clearTextFields();

        //Refresh the timetable display with updated classes
        updateDisplay();
    }
    else//id doesnt exist, display warning and clear textfields
    {
        JOptionPane.showMessageDialog(null, "Sorry, there is no class with
that ID, please check" + String.format("%n")
            + " and try again", "Error", JOptionPane.ERROR_MESSAGE
);
    }
}

```

```

Jan 12, 15 9:04      SportsCentreGUI.java      Page 7/8

        clearTextFields();
    }

}

/**
 * Instantiates a new window and displays the attendance report
 */
public void displayReport()
{
    //Instantiate the reportframe object and pass classList object t
o it*/
    report = new ReportFrame(classList);
}

/**
 * Writes lines to file representing class id, name,
 * tutor and start time and then exits from the program
 */
public void processSaveAndClose()
{
    try
    {
        //Declare local Printwriter
        PrintWriter classListFile = null;
        try
        {
            //Instantiate local printwriter and pass file nam
e to it need a method in fitness program to pass a formatted report to it
            classListFile = new PrintWriter(classesOutFile);
            classListFile.print(classList.getClassesOutFile(
));
        }
        finally
        {
            //Test whether file was successfully initialised
            if (classListFile != null)
            {
                //Close file
                classListFile.close();
            }
        }
    }
    catch (IOException cf)
    {
        System.err.println("Something went wrong: " + cf);
    }
}

//Exit from the program
System.exit(0);
}

/**
 * Process button clicks.
 * @param ae the ActionEvent
 */
public void actionPerformed(ActionEvent ae)
{
    //Test to determine which button has been activated
    if (ae.getSource() == addButton)//add class button has been clic
ked
    {
        //Check if there is an available start time to add a cla
ss to
        if (classList.getNumClasses() < MAX_TIME_SLOTS)
        {
            processAdding();//start time is available, proce
ed
        }
        else //no start time available, no further processing
    }
}

```

Jan 12, 15 9:04

**SportsCentreGUI.java**

Page 8/8

```

        JOptionPane.showMessageDialog(null, "There are no available time slots for adding a class.\n"
        + "Please carefully check the timetable and delete a class before trying this again", "Error", JOptionPane.ERROR_MESSAGE);
        clearTextFields();
    }
}
else if (ae.getSource() == deleteButton) //delete class button has been clicked
{
    //Check if id field is empty, if so, display a warning
    if (idIn.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(null, "ID field is empty..... you figure out the next step:", "Error", JOptionPane.ERROR_MESSAGE);
        clearTextFields();
    }
    else
    {
        processDeletion();
    }
}
else if (ae.getSource() == attendanceButton) //view attendance button has been clicked
{
    //Call method to display attendance report window
    displayReport();
}
else if (ae.getSource() == closeButton) //save and exit button has been clicked
{
    //Call method to create output file and exit program
    processSaveAndClose();
}

}

/**Method to check if id exists
 *
 * @return whether a matching classId was found
 */
private boolean checkId()
{
    //Store user input in a local variable
    String addClassId = idIn.getText().trim();

    //Search class ids
    if (classList.searchClassId(addClassId) >= 0)
    {
        return true; //Classid already exists
    }
    else
    {
        return false; //Classid doesn't exist
    }
}

/**Method to clear all textfields
 */
private void clearTextFields()
{
    idIn.setText(""); //classId textfield
    classIn.setText(""); //class name textfield
    tutorIn.setText(""); //tutor name textfield
}
}

```