SWE316

HW1

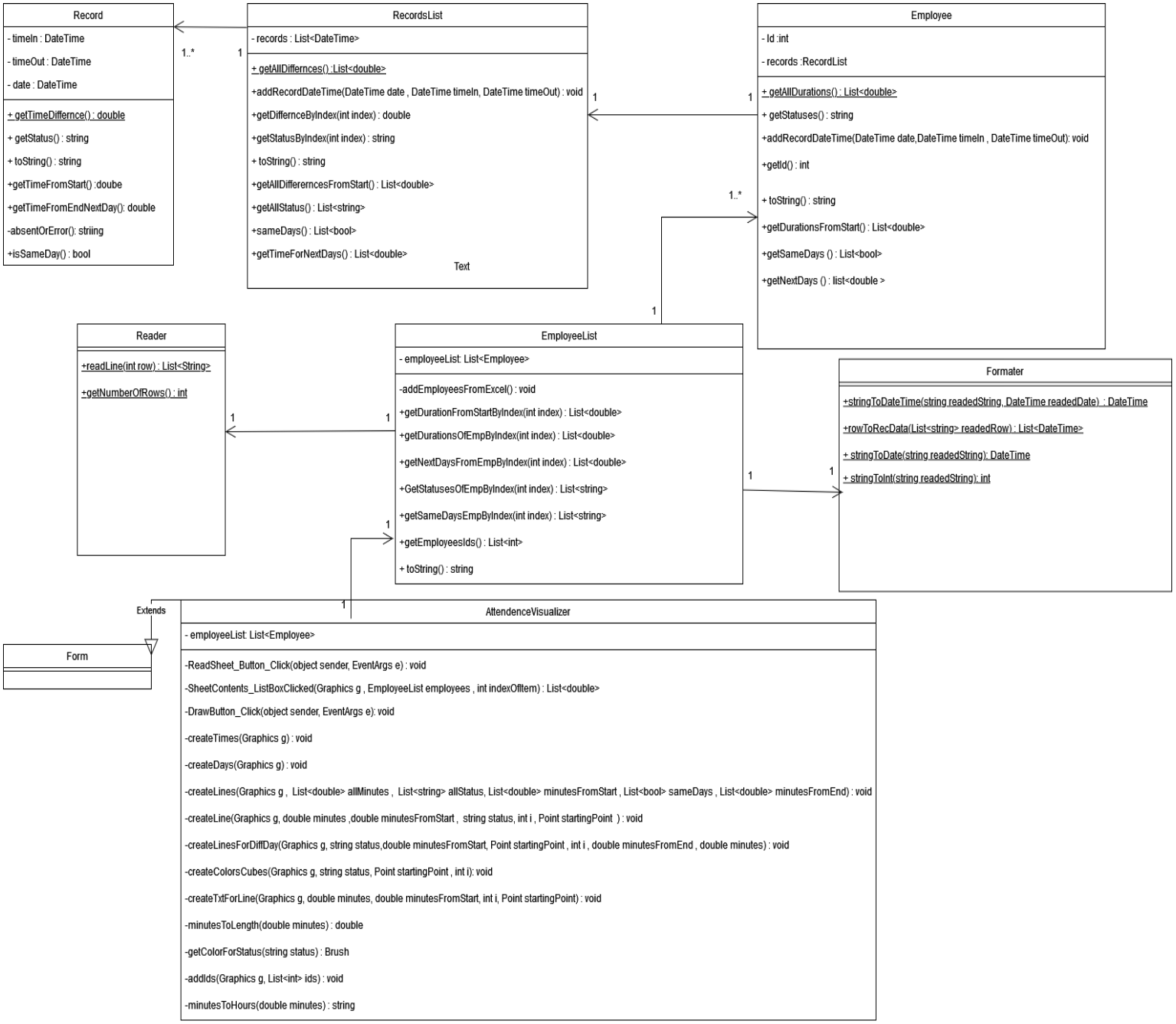
Date of Submission : 30/9/2023

Name : Hussain Asim Al Sayedali

Id : 202038340

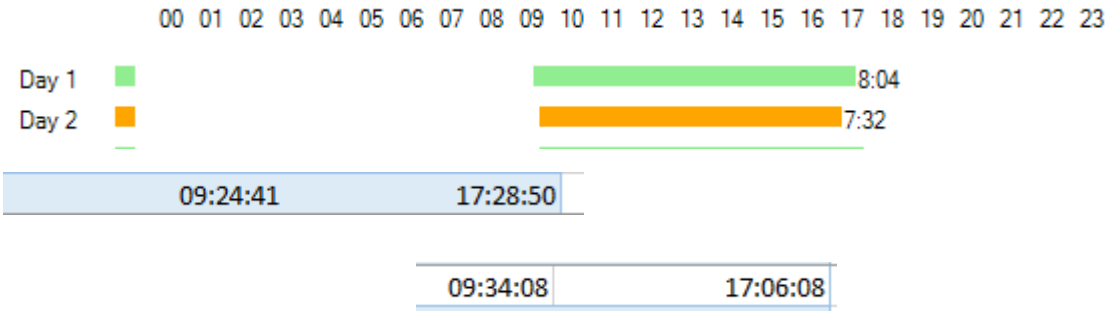
Task	Grade	Your Grade	Comments
Task # 1: Class Diagram	30		
Task # 2: Application	50		
Check list and penalties No Cover page with grade table File name (report) Not in PDF format			-10 ☐ -5 ☐ -10 ☐
Total	80		

Task 1: Class Diagram

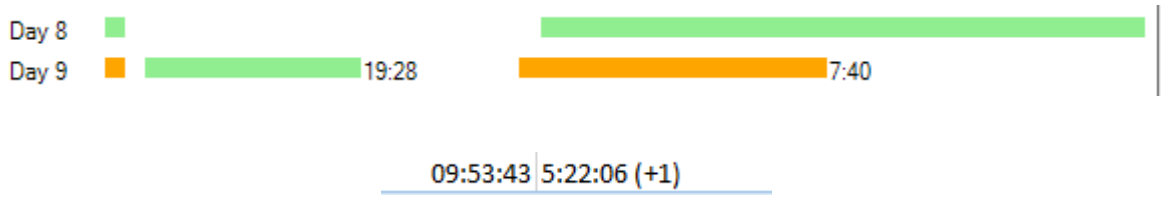


Task 2: Inputs / Outputs

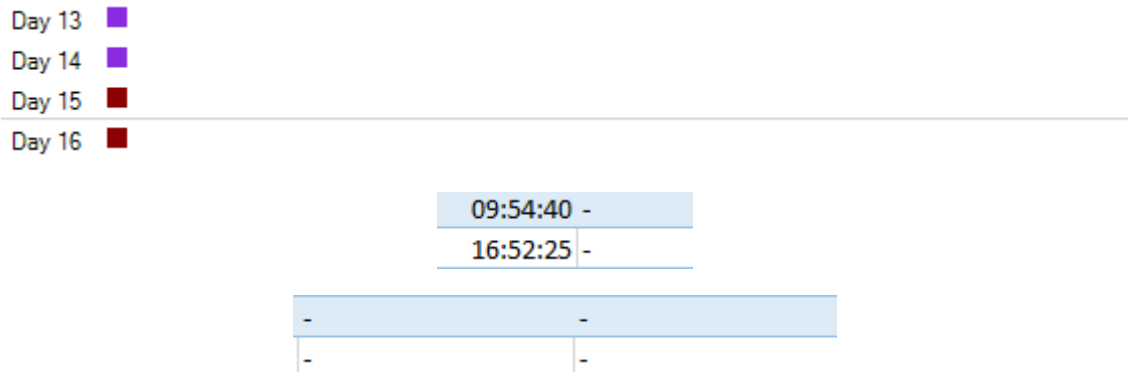
Case 1 and 2 : present and insufficient



Case 3 : working into the next day



Case 4 / 5 : data error and absent



Task 3: Develop the application

Record Class :

```
using System;
using System.Diagnostics;

namespace AttendanceVisualizer
{
    public class Record
    {
        private DateTime date;
        private DateTime timeIn;
        private DateTime timeOut;
        public Record(DateTime dateReaded,DateTime timeInReaded , DateTime
timeOutReaded) {
            this.date = dateReaded;
            this.timeIn = timeInReaded;
            this.timeOut = timeOutReaded;
        }
        public double getTimeDifference() {
            if (absentOrError().Equals("Absent") ||
absentOrError().Equals("Data error")) {
                return 0;
            }
            TimeSpan diff = (timeOut - timeIn).Duration();
            //Trace.WriteLine( "diff in minutes : "+diff + "total mins =" +
diff.TotalMinutes);
            return diff.TotalMinutes;
        }
        public double getTimeFromStart() {
            TimeSpan diff = (timeIn - date).Duration();
            //Trace.WriteLine( "diff in minutes : "+diff + "total mins =" +
diff.TotalMinutes);
            return diff.TotalMinutes;
        }
        public double getTimeFromEndNextDay()
        {
            //Trace.WriteLine(date.Day + ":" + timeOut.Day);
            if (date.Day == timeOut.Day){

                return 0;
            }

            DateTime nextDay = DateTime.Now;
            nextDay = date;
            nextDay = nextDay.AddDays(1);
            TimeSpan diff = (timeOut - nextDay).Duration();
            //Trace.WriteLine( "diff in minutes : "+diff + "total mins =" +
diff.TotalMinutes);
            return diff.TotalMinutes;
        }
        public string getStatus() {
            if (timeIn.Equals(new DateTime(1980, 8, 8, 8, 8, 8)) &&
timeOut.Equals(new DateTime(1980, 8, 8, 8, 8, 8)))
            {
```

```
                return "Absent";
            }
            else if (timeIn.Day != date.Day || timeIn.Equals(new DateTime(1980,
8, 8, 8, 8, 8)) || timeOut.Equals(new DateTime(1980, 8, 8, 8, 8, 8))) {
                return "Data error";
            }
            double duration = getTimeDifference();

            if (duration >= 480)
            {
                return "present";
            }
            else
            {
                return "insuffecient";
            }
        }
        public string toString() {
            string formatedRecord = date.ToString() + " " + timeIn.ToString() +
" " + timeOut.ToString() + " diff : " + getTimeDifference() +"  " + getStatus()
+ "\n";
            return formatedRecord;
        }
        private string absentOrError() {
            if (timeIn.Equals(new DateTime(1980, 8, 8, 8, 8, 8)) &&
timeOut.Equals(new DateTime(1980, 8, 8, 8, 8, 8)))
            {
                return "Absent";
            }
            else if (timeIn.Day != date.Day || timeIn.Equals(new DateTime(1980,
8, 8, 8, 8, 8)) || timeOut.Equals(new DateTime(1980, 8, 8, 8, 8, 8)))
            {
                return "Data error";
            }
            else return "no";
        }
        public bool isSameDay() {
            if (timeIn.Day == timeOut.Day) {
                return true;
            }
            return false;
        }
    }
}
```

RecordList Class :

```
using System;
using System.Collections.Generic;

namespace AttendanceVisualizer
{
    public class RecordList
    {
        private List<Record> records = new List<Record>();
        public RecordList(List<Record> readedRecords)
        {
            records = readedRecords;
        }
        public RecordList(Record readedRecord)
        {
            records.Add(readedRecord);
        }
        public RecordList()
        {
        }
        //public void addRecord(Record record)
        //{
        //    records.Add(record);
        //}
        public void addRecordDateTime(DateTime date, DateTime timeIn, DateTime
timeOut) {
            records.Add(new Record(date, timeIn, timeOut));
        }
        public List<double> getAllDifferences() {
            List<double> differnces = new List<double>();
            for(int i = 0 ; i < records.Count; i++)
            {
                differnces.Add(records[i].getTimeDifference());
            }
            return differnces;
        }
        public List<double> getAllDiffererncesFromStart()
        {
            List<double> differnces = new List<double>();
            for (int i = 0; i < records.Count; i++)
            {
                differnces.Add(records[i].getTimeFromStart());
            }
            return differnces;
        }
        public List<string> getAllStatus() {
            List<string> statuses = new List<string>();
            for (int i = 0; i < records.Count; i++) {
                statuses.Add(records[i].getStatus());
            }
            return statuses;
        }
        public double getDifferencebyIndex(int index) {
            return records[index].getTimeDifference();
        }
        public double getDifferenceFromStartbyIndex(int index)
```

```
        {
            return records[index].getTimeFromStart();
        }
        public string getStatusByIndex(int index)
        {
            return records[index].getStatus();
        }

        public string toString() {
            string formatedAllRecords = "";
            for(int i = 0 ; i < records.Count; i++)
            {
                formatedAllRecords  += records[i].toString();
            }
            return formatedAllRecords;
        }
        public List<bool> sameDays() {
            List<bool> sameDays = new List<bool>();
            for (int i = 0; i < records.Count; i++)
            {
                sameDays.Add(records[i].isSameDay());
            }
            return sameDays;
        }
        public List<double> getTimeForNextDays() {
            List<double> nextDays = new List<double>();
            for (int i = 0; i < records.Count; i++)
            {
                nextDays.Add(records[i].getTimeFromEndNextDay());
            }
            return nextDays;
        }
    }
}
```

Employee Class :

```
using System;
using System.Collections.Generic;

namespace AttendanceVisualizer
{
    public class Employee
    {
        private int id;
        private RecordList recordList = new RecordList();
        public Employee(int idReaded , RecordList recordListReaded) {

            this.id = idReaded;
            this.recordList = recordListReaded;
        }
        public Employee(int idReaded)
        {
            this.id = idReaded;
        }
        public Employee()
        {
        }

        public List<double> getDurations() {

            return recordList.getAllDifferences();
        }
        public List<double> getDurationsFromStart()
        {

            return recordList.getAllDiffererncesFromStart();
        }
        public int getId()
        {
            return id;
        }
        //public void addRecord(Record record) {
        //    recordList.addRecord(record);
        //}
        public void addRecordDateTime(DateTime date,DateTime timeIn , DateTime
timeOut)
        {
            recordList.addRecordDateTime(date, timeIn, timeOut);
        }
        public List<string> getStatuses() {
            return recordList.getAllStatus();
        }
        public string toString() {

            return id + "\n" + recordList.toString();
        }
        public List<bool> getSameDays() {

            return recordList.sameDays();
```

```
        }
        public List<double> getNextDays()
        {

            return recordList.getTimeForNextDays();
        }
    }
}
```

EmployeeList class :

```
using AttendanceVisualizer.proClasses;
using System;
using System.Collections.Generic;
using Excel = Microsoft.Office.Interop.Excel;
namespace AttendanceVisualizer
{
    public  class EmployeeList
    {
        private List<Employee> employeesList = new List<Employee>();
        public EmployeeList(List<Employee> employeesList) {

            this.employeesList = employeesList;
        }
        public EmployeeList(Employee employee)
        {
            employeesList.Add(employee);
        }
        public EmployeeList()
        {
            AddEmployeesFromExcels();
        }

        private void AddEmployeesFromExcels()
        {
            int numberOfRows = Reader.getNumberOfRows();
            //Formater formater = new Formater();
            for (int i = 2; i < numberOfRows; i++) {
                List<string> rowReaded = Reader.readLine(i);

                int idFormated = Formater.stringToInt(rowReaded[0]);

                List<DateTime> rowFormated = Formater.rowToRecData(rowReaded);
                DateTime dateFormated = rowFormated[0];
                DateTime timeInFormated = rowFormated[1];
                DateTime timeOutFormated = rowFormated[2];
                Employee currentEmpReaded;
                if (employeesList.Count == 0 || idFormated !=
employeesList[employeesList.Count - 1].getId())
                {
                    currentEmpReaded = new Employee(idFormated);
                    employeesList.Add(currentEmpReaded);
                }
                else {
                    currentEmpReaded = employeesList[employeesList.Count - 1];
                }
                currentEmpReaded.addRecordDateTime(dateFormated,
timeInFormated, timeOutFormated);

            }
        }
        public List<double> getDurationsOfEmpByIndex(int index)
        {
            return employeesList[index].getDurations();
```

```
        }
        public List<double> getDurationFromStartByIndex(int index)
        {
            return employeesList[index].getDurationsFromStart();
        }
        public List<double> getNextDaysFromEmpByIndex(int index)
        {
            return employeesList[index].getNextDays();
        }


        public List<string> GetStatusesOfEmpByIndex(int index) {
            return employeesList[index].getStatuses();
        }
        public List<bool> getSameDaysEmpByIndex(int index)
        {
            return employeesList[index].getSameDays();
        }
        public List<int> getEmployeesIds() {
            List<int> ids = new List<int>();
            for (int i = 0; i < employeesList.Count; i++) {
                ids.Add(employeesList[i].getId());
            }
            return ids;
        }
        public string toString() {
            string formatedString = "";
            for(int i = 0; i < employeesList.Count ; i++)
            {
                formatedString += employeesList[i].toString();

            }
            return formatedString;
        }
    }
}
```

Formater class :

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AttendanceVisualizer.proClasses
{
    public class Formater
    {
        public Formater() { }
        public static  DateTime stringToDate(string readedString) {
            String[] spearator = { "/" };
            Int32 count = 3;
            string[] time = readedString.Split(spearator, count,
               StringSplitOptions.RemoveEmptyEntries);



            int year = int.Parse(time[2].ToString());
            int month = int.Parse(time[1].ToString());
            int day = int.Parse(time[0].ToString());

            return new DateTime(year, month, day,0,0,0);
        }
        public static DateTime stringToDateTime(string readedString, DateTime
readedDate)
        {
            if (readedString.Contains("-"))
            {
                DateTime errorDate = new DateTime(1980, 8, 8, 8, 8, 8);
                return errorDate;
            }
            else {
                // Trace.WriteLine(readedString);
                if (readedString.Contains("(+1)"))
                {
                    //Trace.WriteLine(readedDate.ToString());
                    readedDate = readedDate.AddDays(1);
                    //Trace.WriteLine(readedDate.ToString());

                    readedString = readedString.Replace("(+1)", "");
                };
                String[] spearator = { ":" };
                Int32 count = 3;
                string[] time = readedString.Split(spearator, count,
                   StringSplitOptions.RemoveEmptyEntries);

                //for (int i = 0; i < time.Length; i++)
                //{
                //    Trace.WriteLine(time[i]);
```

```
                //}

                int hours = int.Parse(time[0].ToString());
                int minutes = int.Parse(time[1].ToString());
                int seconds = int.Parse(time[2].ToString());
                //Trace.WriteLine("hours" + hours + "minutes: " + minutes +
"seconds " + seconds);

                return new DateTime(readedDate.Year, readedDate.Month,
readedDate.Day, hours, minutes, seconds);
            }
        }
        public static List<DateTime> rowToRecData(List<string> readedRow)
        {
            string dateReaded = readedRow[1];
            string timeInReaded = readedRow[2];
            string timeOutReaded = readedRow[3];

            DateTime dateFormated = stringToDate(dateReaded);
            DateTime timeInFormated = stringToDateTime(timeInReaded,
dateFormated);
            DateTime timeOutFormated = stringToDateTime(timeOutReaded,
dateFormated);
            List<DateTime> results = new List<DateTime>();

            results.Add(dateFormated);
            results.Add(timeInFormated);
            results.Add(timeOutFormated);
            return results;
        }
        public static int stringToInt(string num) {
            return Convert.ToInt32(num);
        }
    }
}
```

Reader class :

```
using System.Collections.Generic;
using Excel = Microsoft.Office.Interop.Excel;
namespace AttendanceVisualizer
{
    public class Reader
    {
        public Reader()
        {
        }
        public static List<string> readLine(int row)
        {
            Excel.Workbook xWorkBook =
Globals.ThisAddIn.Application.ActiveWorkbook;

            Excel.Worksheet xWorksheet = xWorkBook.Worksheets.Item[1];
            Excel.Range usedRng;
            usedRng = xWorksheet.UsedRange;
            int numberOfRows = usedRng.Rows.Count;
            //int row = 1;
            Excel.Range idRead = xWorksheet.Cells[row, 1];
            Excel.Range dateRead = xWorksheet.Cells[row, 2];
            Excel.Range inRead = xWorksheet.Cells[row, 3];
            Excel.Range outRead = xWorksheet.Cells[row, 4];

            List<string> rowReaded = new List<string>();
            rowReaded.Add(idRead.Text);
            rowReaded.Add(dateRead.Text);

            rowReaded.Add(inRead.Text);
            rowReaded.Add(outRead.Text);

            return rowReaded;
        }
        public static int getNumberOfRows() {
            Excel.Workbook xWorkBook =
Globals.ThisAddIn.Application.ActiveWorkbook;
            Excel.Worksheet xWorksheet = xWorkBook.Worksheets.Item[1];
            Excel.Range usedRng;
            usedRng = xWorksheet.UsedRange;
            int numberOfRows = usedRng.Rows.Count;
            return numberOfRows;
        }
    }
}
```

AttendenceVisualizer Class :

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using  Excel = Microsoft.Office.Interop.Excel;
using static AttendanceVisualizer.Record;
using AttendanceVisualizer.proClasses;
using System.Reflection;
using Microsoft.Office.Interop.Excel;
using Point = System.Drawing.Point;
using ScrollBars = System.Windows.Forms.ScrollBars;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace AttendanceVisualizer
{
    public partial class AttendanceVisulizerForm : Form
    {

        public AttendanceVisulizerForm()
        {

            InitializeComponent();

            //EmployeeList employees = new EmployeeList();
            //Graphics g = TimeLineDrawingPanel.CreateGraphics();


            //addIds(g, employees.getEmployeesIds());

            //SheetContents_ListBox.Items.Clear();
            //SheetContents_ListBox.Click += new EventHandler(delegate (Object
o, EventArgs a)
            //{
            //    g.Clear(Color.White);
            //    int indexOfItem = SheetContents_ListBox.SelectedIndex;
            //    SheetContents_ListBoxClicked(g, employees, indexOfItem);

            //});


        }

        private void ReadSheet_Button_Click(object sender, EventArgs e )
        {
            {
```

```
                EmployeeList employees = new EmployeeList();
                Graphics g = TimeLineDrawingPanel.CreateGraphics();



                SheetContents_ListBox.Items.Clear();
                addIds(g, employees.getEmployeesIds());


                SheetContents_ListBox.Click += new EventHandler(delegate
(Object o, EventArgs a)
                {
                    g.Clear(Color.White);
                    int indexOfItem = SheetContents_ListBox.SelectedIndex;
                    SheetContents_ListBoxClicked(g , employees ,indexOfItem);

                });


            }

        }
        private void SheetContents_ListBoxClicked(Graphics g , EmployeeList
employees , int indexOfItem) {

            createTimes(g);
            createDays(g);
            drawGrayLines(g);
            createLines(g , employees.getDurationsOfEmpByIndex(indexOfItem) ,
                employees.GetStatusesOfEmpByIndex(indexOfItem) ,
employees.getDurationFromStartByIndex(indexOfItem) ,
                employees.getSameDaysEmpByIndex(indexOfItem) ,
employees.getNextDaysFromEmpByIndex(indexOfItem));
        }


        private void DrawButton_Click(object sender, EventArgs e)
        {
            //g.DrawString("This is a string", DrawButton.Font, txtBrush, p2,
sf); //[redacted]

            //g.DrawLine(GreenPen, p1, p2); // [redacted]
            //Graphics g = TimeLineDrawingPanel.CreateGraphics();

            //Color timeLineColor = Color.Black;
            //Brush timeLineBrush = new SolidBrush(timeLineColor);
            //Pen timeLinePen = new Pen(timeLineColor);


            //Pen GreenPen = new Pen(Color.Green);
            //Brush txtBrush = new SolidBrush(Color.Black);
            //StringFormat sf = new StringFormat();
            //sf.LineAlignment = StringAlignment.Center;
            //sf.Alignment = StringAlignment.Center;


            //Point changingPoint = new Point(80, 10);
```

```
            //Point startingPoint = new Point(80, 10);
            //Point endingPoint = new Point(560, 10);

            ////createTimes(g);
            ////createDays(g);
            ////createLines(g);




            //DateTime startingDate = new DateTime(2016, 6, 5);
            //RectangleF myRectangleF = new RectangleF(30, 30, 40, 10);

        }
        private void createTimes(Graphics g)
        {
            Brush txtBrush = new SolidBrush(Color.Black);
            StringFormat sf = new StringFormat();
            Point changingPoint = new Point(80, 10);
            for (int i = 0; i < 24; i++)
            {
                string drawnString = "0" + i;
                if (i >= 10)
                {
                    drawnString = i + "";
                }
                g.DrawString(drawnString, DrawButton.Font, txtBrush,
changingPoint, sf);
                changingPoint.Offset(20, 0);
            }
        }
        private void createDays(Graphics g)
        {
            Point startingPoint = new Point(80, 10);
            Brush txtBrush = new SolidBrush(Color.Black);
            StringFormat sf = new StringFormat();
            for (int i = 1; i <= 90; i++)
            {
                RectangleF currentRec = new RectangleF(30, 20 + i * 10, 40, 20);
                g.DrawString("Day " + i, DrawButton.Font, txtBrush, 10 , 20 + 20*i
, sf); //████████████████████████████
            }
        }

        private void createLines(Graphics g ,  List<double> allMinutes ,
List<string> allStatus
            , List<double> minutesFromStart , List<bool> sameDays , List<double>
minutesFromEnd)
        {
            //double lengthRec = minutesToLength(minutes);
            Point startingPoint = new Point(80, 10);
            Brush txtBrush = new SolidBrush(Color.Black);
            Brush colordBrush = new SolidBrush(Color.BlueViolet);
            StringFormat sf = new StringFormat();
            for (int i = 0; i < allMinutes.Count; i++)
```

```
            {
                //Trace.WriteLine("length for end" + minutesFromEnd[i]);
                createColorsCubes(g, allStatus[i], startingPoint, i);
                if (sameDays[i])
                {
                    createLine(g, allMinutes[i], minutesFromStart[i],
allStatus[i], i, startingPoint);
                    createTxtForLine(g, allMinutes[i], minutesFromStart[i], i,
startingPoint);
                }
                else {
                    createLinesForDiffDay(g, allStatus[i],
minutesFromStart[i], startingPoint, i , minutesFromEnd[i], allMinutes[i]);
                }

            }
        }
        private void createLine(Graphics g, double minutes ,double
minutesFromStart ,  string status, int i , Point startingPoint  ) {
            Brush txtBrush = new SolidBrush(Color.Black);
            Brush colordBrush = getColorForStatus(status);
            float length = (float)minutesToLength(minutes);
            float lengthFromStart = (float)minutesToLength(minutesFromStart);

            //g.FillRectangle(colordBrush, (startingPoint.X - 20), 20 + (i+1) *
20, 10, 10);

            if(length != 0 && !status.Equals("Absent") && !status.Equals("Data
error"))
                g.FillRectangle(colordBrush, lengthFromStart + startingPoint.X,
+20 + (i+1) * 20, length, 10);
        }
        private void createLinesForDiffDay(Graphics g, string status,double
minutesFromStart, Point startingPoint , int i , double minutesFromEnd , double
minutes) {
            Brush colordBrush = getColorForStatus(status);
            float lengthFromStart = (float)minutesToLength(minutesFromStart);
            float lengthForEnd = (float)minutesToLength(minutesFromEnd);
            StringFormat sf = new StringFormat();
            Brush txtBrush = new SolidBrush(Color.Black);
            //Trace.WriteLine("length for end" + lengthForEnd);
            if (!status.Equals("Absent") && !status.Equals("Data error")) {
                g.FillRectangle(colordBrush, lengthFromStart + startingPoint.X,
+20 + (i + 1) * 20, 500 - lengthFromStart, 10);
                g.FillRectangle(colordBrush, startingPoint.X, +20 + (i + 2) * 20,
lengthForEnd, 10);
                g.DrawString(minutesToHours(minutes) + "", DrawButton.Font,
txtBrush, lengthForEnd + startingPoint.X, 20 + 20 * (i + 2), sf);
            }
        }
        private void createColorsCubes(Graphics g, string status, Point
startingPoint , int i) {
            Brush colordBrush = getColorForStatus(status);
            g.FillRectangle(colordBrush, (startingPoint.X - 20), 20 + (i + 1) *
20, 10, 10);
        }
```

```
        private void createTxtForLine(Graphics g, double minutes, double
minutesFromStart, int i, Point startingPoint)
        {
            StringFormat sf = new StringFormat();
            Brush txtBrush = new SolidBrush(Color.Black);

            float length = (float)minutesToLength(minutes);
            float lengthFromStart = (float)minutesToLength(minutesFromStart);
            float wholeLength = lengthFromStart + length;
            //g.FillRectangle(colordBrush, (startingPoint.X - 20), 20 + (i + 1)
* 20, 10, 10);
            if (length != 0)
                g.DrawString(minutesToHours(minutes) + "", DrawButton.Font,
txtBrush, wholeLength + startingPoint.X, 20 + 20 * (i+1), sf);
        }

        private double minutesToLength(double minutes) {
            return ((minutes / 60)) * 20;
        }
        private Brush getColorForStatus(string status) {
            Brush colordBrush = new SolidBrush(Color.BlueViolet);
            if (status.Equals("Absent"))
            {
                colordBrush = new SolidBrush(Color.DarkRed);
            }
            else if (status.Equals("Data error"))
            {
                colordBrush = new SolidBrush(Color.BlueViolet);

            }
            else if (status.Equals("present"))
            {
                colordBrush = new SolidBrush(Color.LightGreen);

            }
            else if (status.Equals("insuffecient"))
            {
                colordBrush = new SolidBrush(Color.Orange);
            }
            else {
                colordBrush = new SolidBrush(Color.Black);
            }
            return colordBrush;
        }
        private void addIds(Graphics g, List<int> ids)
        {
            for (int i = 0; i < ids.Count; i++)
            {
                SheetContents_ListBox.Items.Add(ids[i]);
            }
        }
        private string minutesToHours(double minutes)
        {
            int hours = (int)minutes / 60;
            int minutesRemaining = (int)minutes % 60;
            string strMinRem = minutesRemaining.ToString();
            if (strMinRem.Length == 1) {
```

```
                strMinRem = "0" + strMinRem;
            }
            else if(strMinRem.Length == 0) {
                strMinRem = "00";
            }

            return hours + ":" + strMinRem;
        }
        private void drawGrayLines(Graphics g) {
            Brush colordBrush = new SolidBrush(Color.LightGray);
            for (int i = 1; i <= 18; i++) {
                g.DrawLine(new Pen(colordBrush), new Point(0, 100 * i + 35), new
Point(700, 100 * i + 35));
            }
        }

        //private void Scroller_scroll(object sender, ScrollEventArgs e)
        //{
        //    TimeLineDrawingPanel.VerticalScroll = -Scroller.Value;
        //}
    }
}
```