SWE 316 - Homework #1 Report

King Fahd Univerity of Petroleum & Minerals

Information and Computer Science Department

SWE 316: Software Design and Construction

Term 251

Due Date: 25/10/2025

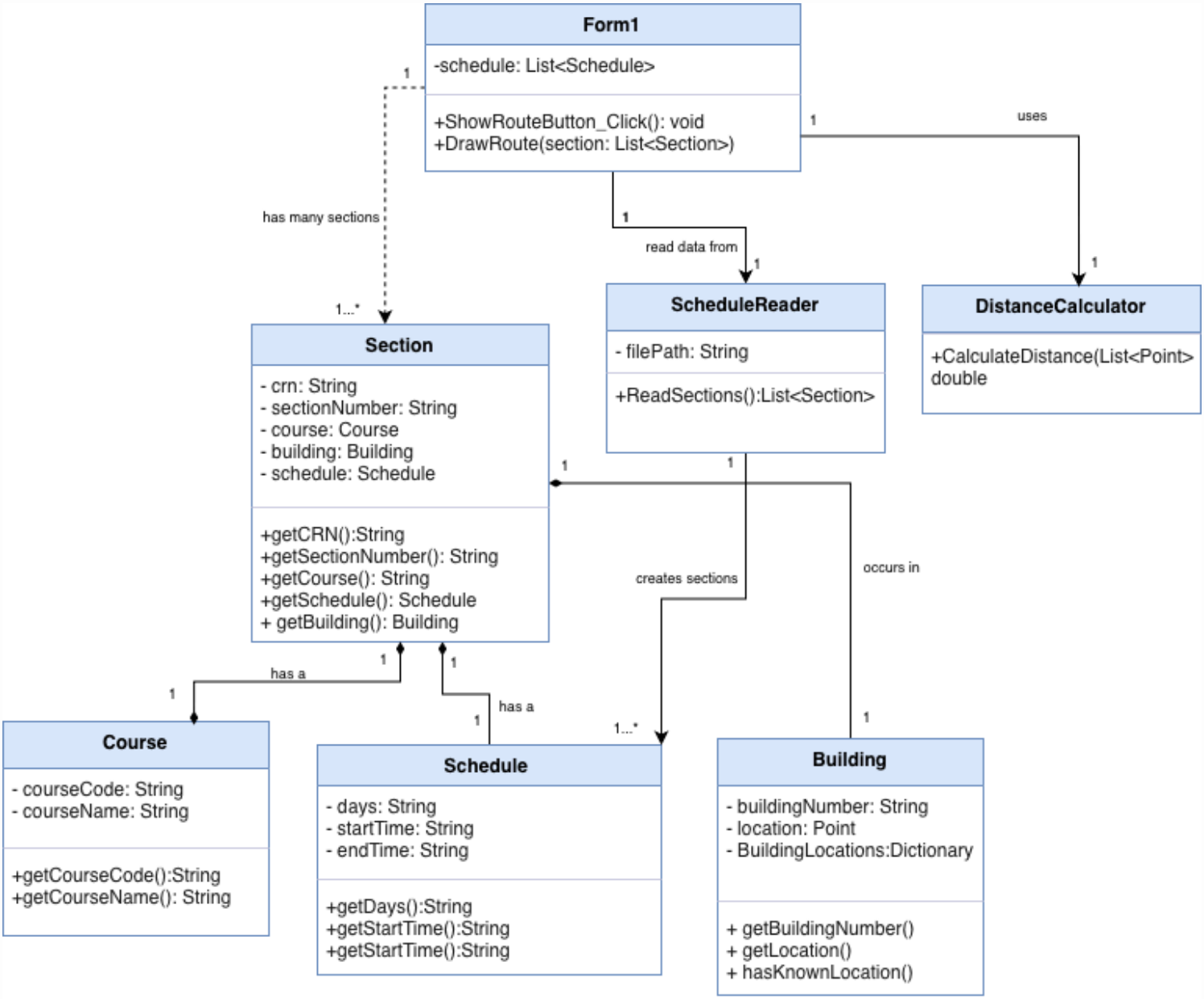
Student Name: Joud Aljabri

Student ID: 202277149

Instructor: Khalid Aljasser

Task	Grade	Your Grade	Comments
Task # 1: Class Diagram	15		
Task # 2: Application	55		
Check list and penalties No Cover page with grade table -5 File name (report) -5 Not in PDF format -5			
Total	70		

Class Diagram



Code Implementation

1- Course Class

Represents a university course (e.g., "SWE316 - Software Design")

```
namespace xAcademics
{
    internal class Course
    {
        private String _code;
        private String _name;

        public Course(string code, string name)
        {
            _code = code;
            _name = name;
        }

        public string getCode()
        {
            return _code;
        }
        public string getName()
        {
            return _name;
        }
    }
}
```

Code Implementation

2- Section Class

```
namespace xAcademics
{
    internal class Section
    {
        private string _crn;
        private string _sectionNumber;
        private Course _course;
        private Schedule _schedule;
        private Building _building;

        public Section(string crn, string sectionNumber,
        Course course, Schedule schedule, Building building)
        {
            _crn = crn;
            _sectionNumber = sectionNumber;
            _course = course;
            _schedule = schedule;
            _building = building;
        }

        public string getCRN()
        {
            return _crn;
        }

        public string getSectionNumber()
        {
            return _sectionNumber;
        }

        public Course getCourse()
        {
            return _course;
        }

        public Schedule getSchedule()
        {
            return _schedule;
        }

        public Building getBuilding()
        {
            return _building;
        }
```

Represents a course section that occurs in a specific building at a certain schedule.

Code Implementation

3- Building Class

Represents a building where every building is mapped to an approximate x-y cooridnates (approximated from the KFUPM Map Image)

```
namespace xAcademics
{
    internal class Building
    {

        private string _buildingNumber;
        private Point _location;

        private static readonly Dictionary<string, Point> BuildingLocations = new Dictionary<string, Point>
        {
            // Approximate (x, y) pixel coordinates on your KFUPM map image

            { "2", new Point(113, 71) },
            { "3", new Point(182, 118) },
            { "4", new Point(164, 99) },
            { "6",  new Point(224, 158) },
            { "7", new Point(222, 184) },
            { "8", new Point(261, 199) },
            { "9", new Point(265, 230 )},
            { "10", new Point(260, 257) },
            { "11", new Point(306, 283) },
            { "12", new Point(172, 177) },
            { "14", new Point(188, 236) },
            { "15", new Point(133, 194) },
            { "16", new Point(159, 122) },
            { "17", new Point(225, 216) },
            { "18", new Point(212, 259) },
            { "19", new Point(223, 282) },
            { "20", new Point(236, 293) },
            { "21", new Point(295, 241 )},
            { "22", new Point(326, 257) },
            { "23", new Point(353, 281 )},
            { "24", new Point(337, 301) },
            { "25", new Point(355, 329) },
            { "26", new Point(69, 54) },
            { "40", new Point(268, 38) },
            { "59", new Point(280, 146) },
```

Code Implementation

3- Building Class

Creating a builing object only if the cooridnates exist in the KFUPM map image since the image does not conatin all building apparing in the excel sheet

```
    public Building(string buildingNumber)
    {
        _buildingNumber = buildingNumber;

        if (BuildingLocations.ContainsKey(buildingNumber))
            _location = BuildingLocations[buildingNumber];
        else
            _location = new Point(-1, -1);
    }

    public string getBuildingNumber()
    {
        return _buildingNumber;
    }

    public Point getLocation ()
    {
        return _location;
    }
    public bool hasKnownLocation()
    {
        return _location.X != -1 && _location.Y != -1;
    }

    public override string ToString()
    {

        return "Building " + _buildingNumber;

    }
}
```

Code Implementation

4- Schedule Class

Represents the schedule of a section, including days and times, both start and end times.

```
namespace xAcademics
{
    internal class Schedule
    {
        private string _days;
        private string _startTime;
        private string _endTime;

        public Schedule(string days, string startTime, string endTime)
        {
            _days = days;
            _startTime = startTime;
            _endTime = endTime;
        }

        public string getDays()
        {
            return _days;
        }

        public string getStartTime()
        {
            return _startTime;
        }

        public string getEndTime()
        {
            return _endTime;
        }

        public override string ToString()
        {
            return "Schedule: " + _days + " from " + _startTime + " to " + _endTime;
        }
    }
```

Code Implementation

5- ScheduleReader Class

The ScheduleReader class is responsible for reading course schedule data from an Excel file
and converting it into a structured format using objects. It acts as a data loader for the scheduling system.

```
namespace xAcademics
{
    internal class ScheduleReader
    {
        private readonly string _filePath;

        public ScheduleReader(string filePath)
        {
            _filePath = filePath;
        }

        public List<Section> ReadSections()
        {
            List<Section> sections = new List<Section>();
            Excel.Application excelApp = new Excel.Application();
            Excel.Workbook workbook = null;
            Excel.Worksheet worksheet = null;

            try
            {
                workbook = excelApp.Workbooks.Open(_filePath);
                worksheet = (Excel.Worksheet)workbook.Worksheets[1];
                Excel.Range usedRange = worksheet.UsedRange;
                int rowCount = usedRange.Rows.Count;


                for (int row = 2; row <= rowCount; row++)
                {

                    string crn = worksheet.Cells[row, 2].Value?.ToString();
                    string courseCode = worksheet.Cells[row, 3].Value?.ToString();
                    string sectionNumber = worksheet.Cells[row, 5].Value?.ToString();
                    string courseName = worksheet.Cells[row, 6].Value?.ToString();
                    string days = worksheet.Cells[row, 8].Value?.ToString();
                    string startTime = worksheet.Cells[row, 9].Value?.ToString();
                    string endTime = worksheet.Cells[row, 10].Value?.ToString();
                    string buildingNumber = worksheet.Cells[row, 11].Value?.ToString();
```

Creates a section list, opens a new workbook and a new worksheet

Opens the righ excel file, selects the first worksheet and calculates how many rows we have ,

Iterate through all rows in the sheet, selects the values neeeded from each column of a row and turns them to strings

Code Implementation

5- ScheduleReader Class

```
{
    {
        {
            {


                if (string.IsNullOrWhiteSpace(crn) || string.IsNullOrWhiteSpace(buildingNumber))
                    continue;


                Course course = new Course(courseCode, courseName);
                Schedule schedule = new Schedule(days, startTime, endTime);
                Building building = new Building(buildingNumber);
                Section section = new Section(crn, sectionNumber, course, schedule, building);

                sections.Add(section);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error reading schedule file: " + ex.Message,
                "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            workbook?.Close(false);
            excelApp.Quit();
        }

        return sections;
    }
```

Creating the needed objects we need from each value

Code Implementation

6- DistanceCalculator Class

A class to calculate the distances needed for the buildings

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

namespace xAcademics
{
    internal class DistanceCalculator
    {
        public double CalculateDistance(List<Point> points)
        {
            if (points == null || points.Count < 2)
                return 0;

            double totalDistance = 0;

            for (int i = 0; i < points.Count - 1; i++)
            {
                int dx = points[i + 1].X - points[i].X;
                int dy = points[i + 1].Y - points[i].Y;

                double segment = Math.Sqrt(dx * dx + dy * dy);
                totalDistance += segment;
            }

            double meters = totalDistance * 0.8;

            return meters;
        }
    }
}
```