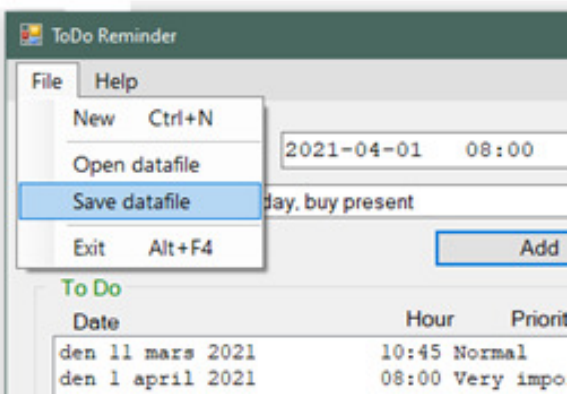
Assignment 6: Text Files

1. Overview

Saving data to files permanently is an essential part of all applications working with data that need to be referred to again. Text files are very practical to use not only for saving data but also for exporting and importing data in textual form. Different programs can communicate this way. As an example you can save data to a text file in a way that can be loaded in Excel to perform other calculations or print out using the latter's printing features.

You can add a "Save As…" submenu if you are using the SaveFileDialog control



2. Writing and reading Files

2.1 To save data to a text file and load previously saved data from a text file, you can use the following algorithm:

- Create a class **FileManager** for handling reading and writing logics using text files.
- Write a method that opens a file and saves data from your application to a text file.
- Write another method that opens a file which contains data saved previously by your application, for reading and transferring the data back to your application.
- Data that is saved should be read exactly in the same order.
- In our current application, we are going to save all the tasks that are registered in the **TaskList** in the **TaskManager** object. Begin saving some extra information:
    - Save a token or a version number so you detect that the file is one saved by your application the next time you open the file for reading.
    - Save the current number of tasks stored in the taskList (taskList.Count)
    - For every task in the list
        - Save the values in the instance variables of the task object (description, priority, date)

- When reading data:
    - Clear the contents of the **taskList** (if already created) or re-created it to get a new empty list. Alternatively, you can create or recreate the **TaskManager** object in the **MainForm**.
    - Read the first line and compare the text to what you had saved as control token (as described above).
        - If it matches continue reading the file.
        - Else – cancel reading.
    - Read the number of tasks to read from the file
    - For each of the tasks
        - Create a Task object, read data from the file
        - Save the object in the taskList.

2.2 Draw the File menu as illustrated in the figure above. When the user clicks the **Save datafile,** save all tasks that are registered in the **tasklist** in the **taskManager** class to a file in the directory where the applications **exe**-file is located.(bin\Debug or obj\Debug).

2.3 Declare a string variable in the **MainForm** to store the complete path plus the file name used for storing data.

```
public partial class MainForm : Form
{
    //private Task task;
    private TaskManager taskManager;

    //Working with only one file located in the same directory
    //as the application's EXE- file
    string fileName = Application.StartupPath + "\\Tasks.txt";
```

2.4 Double-click on the menu item **Save datafile** in Visual studio and complete the method with the following code (MainForm).

```
private void menuFileSave_Click(object sender, EventArgs e)
{
    string errMessage = "Something went wrong while saving data to file";

    bool ok = taskManager.WriteDataToFile ( fileName );
    if (!ok)
        MessageBox.Show ( errMessage );
    else
        MessageBox.Show ( "Data saved to file:" + Environment.NewLine +fileName);
}
```

2.5 Double-click on the menu item **Open datafile** in Visual studio and complete the method with the following code (MainForm).

```
private void menuFileOpen_Click(object sender, EventArgs e)
{
    string errMessage = "Something went wrong when opening the file";

    bool ok = taskManager.ReadDataFromFile(fileName );
    if (!ok)
        MessageBox.Show ( errMessage );
    else
        UpdateGUI ( );
}
```

2.6 The above two methods make calls to methods in the **TaskManager** class. So you need to copy the following two methods into the **TaskManager** class:

```
//Send the taskList object declared in above to which
//data from the fileName is saved.  As taskList is an object
//it will have the changes made in the FileManager when
//the method ReadTaskLisstFrFile returns.
public bool ReadDataFromFile(string fileName)
{
    FileManager fileManager = new FileManager();

    //objects are passed by ref so taskList will be updated.
    return fileManager.ReadTaskLisstFrFile(taskList, fileName);
}
```

```
//FileManager is a class that handles saving and reading data
//to and from a text file. Send the taskList from which
//data is read and saved to the file fileName
public bool WriteDataToFile(string fileName)
{
    FileManager fileManager = new FileManager();
    return fileManager.SaveTaskListToFile(taskList, fileName);
}
```

2.7 The TaskManager class in turn makes calls to methods in the FileManager class.

2.8 The contents of the FileManager is given in the next two pages as images and not as code. The main reason is that copying code does not often make you pay attention to each statement. By re-writing, you will get a chance to follow each step, understand and think about why a certain step is taken.

2.9 Write each line of code and make sure that you understand the purpose of the code. **Write comments to note your thoughts and understanding of the code**.

Good Luck.

Farid Naisan
Course Coordinator and teacher

```
//Don't forget to include using System.IO; in above
class FileManager
{
    //Write this token as the first line in the textfile as a stample
    //to mark that the file is saved by this application
    //
    private const string fileVersionToken = "ToDoRe_21";
    //The file version can help you to make old files compatible
    //The fileVersion does not have to be the same as the app's version nr.
    private const double fileVersionNr = 1.0;

    public bool SaveTaskListToFile(List<Task> taskList, string fileName)
    {
        bool ok = true;
        StreamWriter writer = null;
        try
        {
            writer = new StreamWriter(fileName);
            writer.WriteLine(fileVersionToken);
            writer.WriteLine(fileVersionNr);
            writer.WriteLine(taskList.Count);

            for (int i = 0; i < taskList.Count; i++)
            {
                writer.WriteLine(taskList[i].Description);
                writer.WriteLine(taskList[i].Priority.ToString());
                writer.WriteLine(taskList[i].TaskDate.Year);
                writer.WriteLine(taskList[i].TaskDate.Month);
                writer.WriteLine(taskList[i].TaskDate.Day);
                writer.WriteLine(taskList[i].TaskDate.Hour);
                writer.WriteLine(taskList[i].TaskDate.Minute);
                writer.WriteLine(taskList[i].TaskDate.Second);
            }
        }
        catch
        {
            //if any error occurs in above try-block, the execution will
            //jump to this block (avoiding program crash)
            ok = false;
        }
        finally
        {
            //This block is always executed no matter if an error occurs
            //or not
            if (writer != null)
                writer.Close();
        }
        return ok;
    }
```

```
public bool ReadTaskLisstFrFile(List<Task> taskList, string fileName)
{
    bool ok = true;
    StreamReader reader = null;

    try
    {
        //Clear the contents of taskList
        if (taskList != null)
            taskList.Clear();
        else
            taskList = new List<Task>();

        reader = new StreamReader(fileName);

        //Ensure that it is the correct file
        string versionTest = reader.ReadLine();
        //Ensure that it has the correct version nr.
        double version = double.Parse(reader.ReadLine());

        if ((versionTest == fileVersionToken)  && (version == fileVersionNr) )//correct file
        {
            //read num of items (tasks) to read
            int count = int.Parse(reader.ReadLine());
            for (int i = 0; i < count; i++)
            {
                Task task = new Task();
                task.Description = reader.ReadLine();
                task.Priority = (PriorityType)Enum.Parse(typeof(PriorityType), reader.ReadLine());

                //DateTime.Year,Month, etc. are readonly!
                int year = 0, month = 0, day = 0;
                int hour = 0, minute = 0, second = 0;

                year = int.Parse(reader.ReadLine());
                month = int.Parse(reader.ReadLine());
                day = int.Parse(reader.ReadLine());
                hour = int.Parse(reader.ReadLine());
                minute = int.Parse(reader.ReadLine());
                second = int.Parse(reader.ReadLine());

                task.TaskDate = new DateTime(year, month, day, hour, minute, second);

                taskList.Add(task);
            }
        }
        else
            ok = false;
    }
    catch
    {
        ok = false;
    }
    finally
    {
        if (reader != null)
            reader.Close();
    }
    return ok;
}
```