**Changes made:**

* Replaced all PDFSharp code with QuestPDF.
* Added a header with the current date (left), "OBD report" (center), and "by Csaba Jozsi" (right).
* Added a footer with the page number and total pages.
* The PDF generation logic is now in QuestPDF's fluent API style.

using QuestPDF.Fluent;

using QuestPDF.Helpers;

using QuestPDF.Infrastructure;

using System;

using System.IO;

using System.IO.Ports;

using System.Linq;

class Program

{

static void Main()

{

string[] ports = SerialPort.GetPortNames();

if (ports.Length == 0)

{

Console.WriteLine("Nincs elérhető COM port a gépen.");

Console.ReadKey();

return;

}

bool sikeresKapcsolodas = false;

string responseFile = "Response.txt";

string pdfFile = "Response.pdf";

foreach (string port in ports)

{

using (SerialPort serialPort = new SerialPort(port))

{

serialPort.BaudRate = 115200;

serialPort.DataBits = 8;

serialPort.Parity = Parity.None;

serialPort.StopBits = StopBits.One;

serialPort.Handshake = Handshake.None;

serialPort.ReadTimeout = 2000;

serialPort.WriteTimeout = 2000;

serialPort.NewLine = "\r";

try

{

serialPort.Open();

Console.WriteLine($"Sikeresen kapcsolódtunk a {port} porthoz.");

Console.ReadKey();

sikeresKapcsolodas = true;

if (File.Exists("ELM327\_parancsok\_alkalmaz\_Dauntless.txt"))

{

string[] atParancsok = File.ReadAllLines("ELM327\_parancsok\_alkalmaz\_Dauntless.txt");

using (StreamWriter sw = new StreamWriter(responseFile, false))

{

foreach (string parancs in atParancsok)

{

if (string.IsNullOrWhiteSpace(parancs))

continue;

serialPort.Write(parancs + "\r");

Console.WriteLine($">>> Küldve: {parancs}");

try

{

string valasz = serialPort.ReadLine();

Console.WriteLine($"<<< Válasz: {valasz}");

sw.WriteLine($"{parancs} -> {valasz}");

}

catch (TimeoutException)

{

Console.WriteLine("<<< Válasz timeout.");

Console.ReadKey();

sw.WriteLine($"{parancs} -> (timeout)");

}

System.Threading.Thread.Sleep(10);

}

}

// PDF generálás a Response.txt tartalmából

CreatePdfFromTextFile(responseFile, pdfFile);

Console.WriteLine($"PDF riport létrehozva: {pdfFile}");

Console.ReadKey();

}

else

{

Console.WriteLine("Az 'ELM327\_parancsok\_alkalmaz.txt' fájl nem található.");

Console.ReadKey();

}

serialPort.Close();

break;

}

catch (Exception ex)

{

Console.WriteLine($"Nem sikerült kapcsolódni a {port} porthoz. Hiba: {ex.Message}");

Console.ReadKey();

}

}

}

if (!sikeresKapcsolodas)

{

Console.WriteLine("Nem sikerült egyetlen COM porthoz sem kapcsolódni.");

Console.ReadKey();

}

}

static void CreatePdfFromTextFile(string textFilePath, string pdfFilePath)

{

string[] lines = File.ReadAllLines(textFilePath);

var document = Document.Create(container =>

{

container.Page(page =>

{

page.Margin(40);

page.Header().Element(header =>

{

header.Row(row =>

{

row.RelativeColumn().AlignLeft().Text(DateTime.Now.ToString("yyyy-MM-dd")).FontSize(10);

row.RelativeColumn().AlignCenter().Text("OBD report").FontSize(16).Bold();

row.RelativeColumn().AlignRight().Text("by Csaba Jozsi").FontSize(10);

});

});

page.Content().Column(col =>

{

foreach (var line in lines)

{

col.Item().Text(line).FontSize(12);

}

});

page.Footer().AlignCenter().Text(x =>

{

x.CurrentPageNumber();

x.Span(" / ");

x.TotalPages();

}).FontSize(10);

});

});

document.GeneratePdf(pdfFilePath);

}

}

**2025.08.19. Final QuestPDF version:**

using QuestPDF.Fluent;

using QuestPDF.Helpers;

using QuestPDF.Infrastructure;

using System;

using System.IO;

using System.IO.Ports;

using System.Linq;

using System.Reflection.Metadata;

// Explicitly alias the QuestPDF Document to avoid ambiguity

using QuestPdfDocument = QuestPDF.Fluent.Document;

class Program

{

static void Main()

{

string[] ports = SerialPort.GetPortNames();

if (ports.Length == 0)

{

Console.WriteLine("Nincs elérhető COM port a gépen.");

Console.ReadKey();

return;

}

bool sikeresKapcsolodas = false;

string responseFile = "Response.txt";

string pdfFile = "Response.pdf";

foreach (string port in ports)

{

using (SerialPort serialPort = new SerialPort(port))

{

serialPort.BaudRate = 115200;

serialPort.DataBits = 8;

serialPort.Parity = Parity.None;

serialPort.StopBits = StopBits.One;

serialPort.Handshake = Handshake.None;

serialPort.ReadTimeout = 2000;

serialPort.WriteTimeout = 2000;

serialPort.NewLine = "\r";

try

{

serialPort.Open();

Console.WriteLine($"Sikeresen kapcsolódtunk a {port} porthoz.");

Console.ReadKey();

sikeresKapcsolodas = true;

if (File.Exists("ELM327\_parancsok\_alkalmaz\_Dauntless.txt"))

{

string[] atParancsok = File.ReadAllLines("ELM327\_parancsok\_alkalmaz\_Dauntless.txt");

using (StreamWriter sw = new StreamWriter(responseFile, false))

{

foreach (string parancs in atParancsok)

{

if (string.IsNullOrWhiteSpace(parancs))

continue;

serialPort.Write(parancs + "\r");

Console.WriteLine($">>> Küldve: {parancs}");

try

{

string valasz = serialPort.ReadLine();

Console.WriteLine($"<<< Válasz: {valasz}");

sw.WriteLine($"{parancs} -> {valasz}");

}

catch (TimeoutException)

{

Console.WriteLine("<<< Válasz timeout.");

Console.ReadKey();

sw.WriteLine($"{parancs} -> (timeout)");

}

System.Threading.Thread.Sleep(10);

}

}

// PDF generálás a Response.txt tartalmából

CreatePdfFromTextFile(responseFile, pdfFile);

Console.WriteLine($"PDF riport létrehozva: {pdfFile}");

Console.ReadKey();

}

else

{

Console.WriteLine("Az 'ELM327\_parancsok\_alkalmaz.txt' fájl nem található.");

Console.ReadKey();

}

serialPort.Close();

break;

}

catch (Exception ex)

{

Console.WriteLine($"Nem sikerült kapcsolódni a {port} porthoz. Hiba: {ex.Message}");

Console.ReadKey();

}

}

}

if (!sikeresKapcsolodas)

{

Console.WriteLine("Nem sikerült egyetlen COM porthoz sem kapcsolódni.");

Console.ReadKey();

}

}

static void CreatePdfFromTextFile(string textFilePath, string pdfFilePath)

{

string[] lines = File.ReadAllLines(textFilePath);

var document = QuestPdfDocument.Create(container =>

{

container.Page(page =>

{

page.Margin(40);

page.Header().Element(header =>

{

header.Row(row =>

{

row.RelativeItem().AlignLeft().Text(DateTime.Now.ToString("yyyy-MM-dd")).FontSize(10);

row.RelativeItem().AlignCenter().Text("OBD report").FontSize(16).Bold();

row.RelativeItem().AlignRight().Text("by Csaba Jozsi").FontSize(10);

});

});

page.Content().Column(col =>

{

foreach (var line in lines)

{

col.Item().Text(line).FontSize(12);

}

});

page.Footer().AlignCenter().Text(x =>

{

x.CurrentPageNumber().FontSize(10);

x.Span(" / ").FontSize(10);

x.TotalPages().FontSize(10);

});

});

});

document.GeneratePdf(pdfFilePath);

}

}