HTMLTOCwriterProject – Automate Creating TOC HTML files for PDF Documents

Files:
Java Eclipse project folder: Eclipse202506-HTMLTOCwriter
Java Jar file: HTMLTOCWriter.jar
Properties file: config.properties (produced by the application)
Manually created: pdf filename.txt text file representing the TOC essential items for HTML anchor code creation
Java Source files (at end of this document)

The Config.Properties file is created with the SwingInterface's first four textbox variables. Here is the completed properties file contents with the four property elements:

#Tue Jul 08 11:45:22 PDT 2025

1. **toc.Composer=Bach**
2. **toc.bookTitle=Seven Toccaten - BWV 910 - BWV 916**
3. **toc.htmFilename=TOC-Bach-7-Toccaten-BWV-910-916-HIGHQuality.html**
4. **toc.pdfFilename=Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf**

The TOC-HTML file that is the final result of the process contains the properties in the heading of the html (shown in red):

</style>

</head>

<title>**TOC-Bach-7-Toccaten-BWV-910-916-HIGHQuality.html**</title><html> ← **{toc.htmFilename=TOC-Bach-7-Toccaten-BWV-910-916-HIGHQuality.html} (#3.)**

<body>

<table class="center">

<tr><th colspan='3'>Table of Contents</th></tr>

<tr><th colspan='3'>**Bach**</th></tr> ← **{toc.Composer=Bach} (#1.)**

<tr><th colspan='3'>**Seven Toccaten - BWV 910 - BWV 916**</th></tr> ← **{toc.bookTitle=Seven Toccaten - BWV 910 - BWV 916} (#2.)**

<tr><th colspan='3'> </th></tr>

<tr><th>Item No.</th><th>**Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf**
Sheet Music Contents</th><th>Page</th></tr> ←{ **toc.pdfFilename=Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf} (#4.)**

<tr><td>1</td><td style="font-weight:bold;">BWV 910</td><td>3</td></tr>

<tr><td>2</td><td style="font-weight:bold;">BWV 911</td><td>15</td></tr>

<tr><td>3</td><td style="font-weight:bold;">BWV 912a</td><td>28</td></tr>

<tr><td>4</td><td style="font-weight:bold;">BWV 912</td><td>40</td></tr>

<tr><td>5</td><td style="font-weight:bold;">BWV 913 early version</td><td>52</td></tr>

<tr><td>6</td><td style="font-weight:bold;">BWV 913 late version</td><td>65</td></tr>

<tr><td>7</td><td style="font-weight:bold;">BWV 914</td><td>80</td></tr>

<tr><td>8</td><td style="font-weight:bold;">BWV 915</td><td>89</td></tr>

<tr><td>9</td><td style="font-weight:bold;">BWV 916</td><td>102</td></tr>

</table></body></html>

The resulting TOC HTML page:

Table of Contents

Bach

Seven Toccaten - BWV 910 - BWV 916

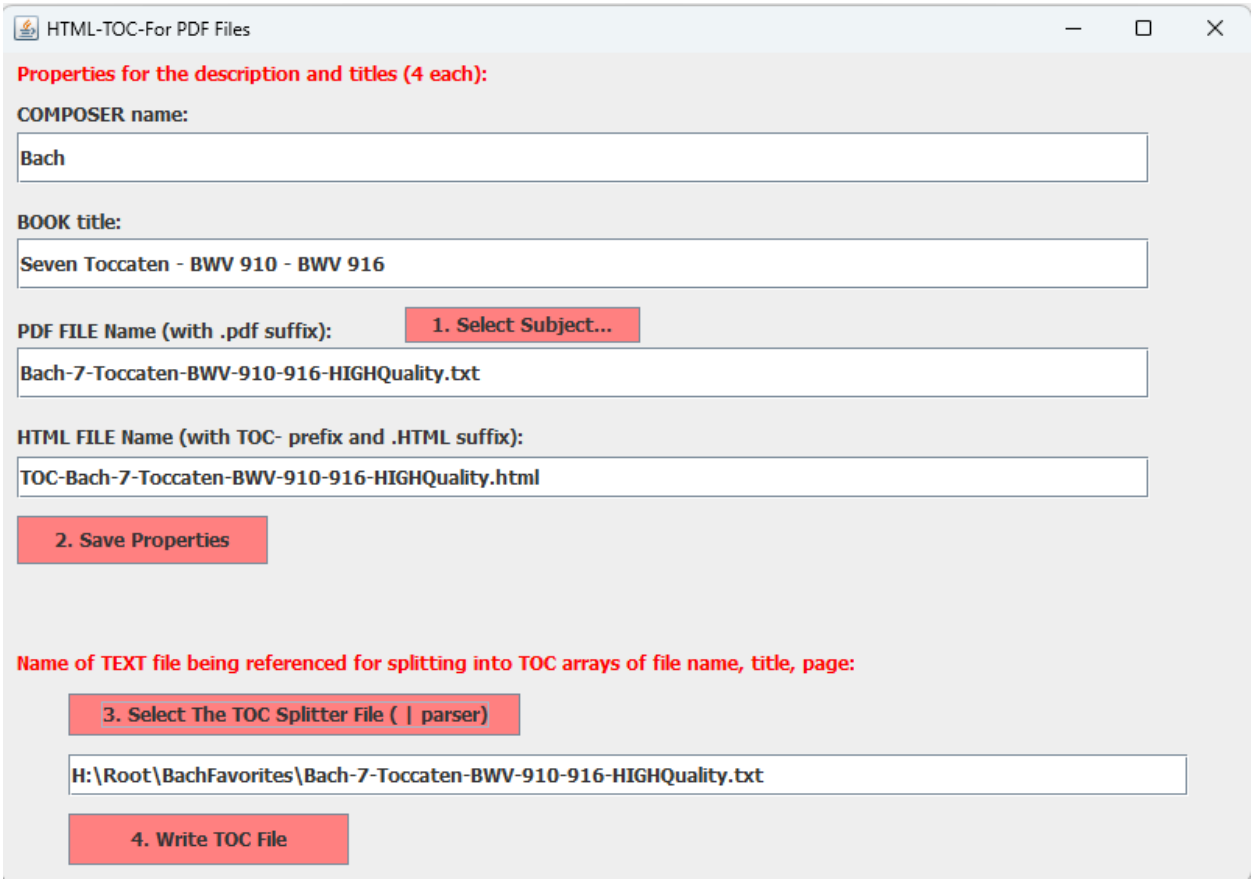
Item No.	Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf Sheet Music Contents	Page
1	BWV 910	3
2	BWV 911	15
3	BWV 912a	28
4	BWV 912	40
5	BWV 913 early version	52
6	BWV 913 late version	65
7	BWV 914	80
8	BWV 915	89
9	BWV 916	102

Note: The link to each page is produced from the splitter file. Each line of the splitter file produces each html anchor:

first line:
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 910,3

Each topic in the TOC is referenced by a line in the splitter file, which is then saved into the same directory as the source PDF file and the TOC-html file.

The Swing Interface:



The PDF File Name is selected with the '1. Select Subject pdf File' button and the splitter text file is chosen with the '3. Select The TOC Splitter File (| parser)' button. The splitter file is manually created and saved in the same directory (for a matter of convenience) as the source pdf file and the resulting TOC-html file.

Here are the contents of the Splitter File (named Splitter because the content lines are split into arrays for each read of the file based upon comma separators) :

Bach-7-Toccaten-BWV-910-916-HIGHQuality.txt file:

Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 910,3
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 911,15
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 912a,28
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 912,40
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 913 early version,52
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 913 late version,65
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 914,80
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 915,89
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 916,102

The text splitter file is the basis of the resulting HTML TOC file. Each line consists of (1) the pdf file name that is being referenced for the table of contents, (2) the title of the referenced content (in this case the contents are the 7 BWF numbers of the toccatas), and (3) the page number. Each line is split into a array of 3 elements that are used in the creation of the anchors that reference the title pages. The first reference is shown below:

This line in the splitter file:
Bach-7-Toccaten-BWV-910-916-HIGHQuality.pdf,BWV 910,3

This HTML anchor code is developed from the line above:
<tr><td>1</td><td style="font-weight:bold;">**BWV 910**</td><td>**3**</td></tr>

The process of creating a table of contents for very large numbers of topics is much simpler with this approach versus creating all the HTML by hand.

In summary, three files are used in this process:

1. Splitter file (text file explained above) – pdfFile name.txt which contains a comma-separated list of contents
2. The properties file which is created from entering the four text items onto the Swing Interface
3. The TOC-html TOC file which is the final product
4. The pdf source file which is referenced for the table of contents

Java Source Files:

1. Director.java
2. FileChooser.java
3. PropertiesManagement.java
4. SwingInterface.java
5. TOCLine.java
6. WriteFile.java

==========
Director.java code

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class Director {
	private ArrayList<TOCLine> arrTOCList = new ArrayList<TOCLine>();

	public void loadTOCLines(String pSplitterFilePath) throws IOException {
```

```
		String path = pSplitterFilePath;
		try (BufferedReader reader = new BufferedReader(new FileReader(path))) {
			String line;
			while ((line = reader.readLine()) != null) {

				String[] arrLine = line.split(",");
				TOCLine toc = new TOCLine();
				toc.addLine(arrLine[0],  arrLine[1],  arrLine[2]);
				arrTOCList.add(toc);

				  for (int i = 0; i < arrLine.length; i++) {
				      System.out.println("Element at index " + i + ": " + arrLine[i]);

				  }
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
		System.out.println(" arrTOCList is this long: " + arrTOCList.size() );
		System.out.println("arrTOCList 4 is this long: " + arrTOCList.get(4).getPage() +
"  " + arrTOCList.get(4).getTitle());

		// == pass to a file writer ===
		WriteFile writeFile = new WriteFile();
		PropertiesManagement properties = new PropertiesManagement();
		writeFile.writePreamble(properties.getTOCHTMLFileName());

		writeFile.writeBody(properties.getTOCHTMLFileName(), arrTOCList);

		for(int i = 0; i < arrTOCList.size(); i++) {
			System.out.println(arrTOCList.get(i));

		};
	}
}
```

==========

FileChooser.java code

```
import java.awt.Dimension;
import java.io.File;

import javax.swing.JFileChooser;
```

```
public class FileChooser {

        public File chooseDirectory(String pDlgTitle) {

        JFileChooser chooser = new JFileChooser();
  File fSelectedFile = null;

        chooser.setDialogTitle(pDlgTitle);
        chooser.setPreferredSize(new Dimension(800, 1000)); // Set preferred size to
800x1000

  chooser.setCurrentDirectory(new java.io.File("."));

  chooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
  chooser.setAcceptAllFileFilterUsed(true);

  if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
         fSelectedFile = chooser.getSelectedFile();
           if (fSelectedFile != null) {
             // Proceed with file operations using selectedFile
             System.out.println("Selected file: " + fSelectedFile.getAbsolutePath());
           }
        }
  return fSelectedFile;
        }
}
=======================================
PropertiesManagement.java code

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Properties;

public class PropertiesManagement {

public void saveProperties(String ptocComposer, String ptocBookTitle, String
ptocPdfFilename, String ptocHtmFilename) {

    try (OutputStream output = new FileOutputStream("config.properties")) {

        Properties prop = new Properties();
```

```
        // set the properties value
        prop.setProperty("toc.Composer", ptocComposer);
        prop.setProperty("toc.bookTitle", ptocBookTitle);
        prop.setProperty("toc.pdfFilename", ptocPdfFilename); //including .pdf
        prop.setProperty("toc.htmFilename", ptocHtmFilename);

        // save properties to project root folder
        prop.store(output, null);

        System.out.println(prop);

    } catch (IOException io) {
        io.printStackTrace();
    }

}

        public String getTOCComposer() {
         Properties prop = new Properties();

         try (InputStream input = new FileInputStream("config.properties")) {
    prop.load(input);
    System.out.println(prop.getProperty("toc.Composer"));
        } catch (IOException ex) {
                ex.printStackTrace();
        }
        return(prop.getProperty("toc.Composer").toString());
        }

        public String getTOCBookTitle() {
         Properties prop = new Properties();

         try (InputStream input = new FileInputStream("config.properties")) {
   prop.load(input);
   System.out.println(prop.getProperty("toc.bookTitle"));
                } catch (IOException ex) {
                ex.printStackTrace();
                }
        return(prop.getProperty("toc.bookTitle").toString());
        }

        public String getTOCPDFFileName() {
                 Properties prop = new Properties();
```

```
            try (InputStream input = new FileInputStream("config.properties")) {
            prop.load(input);
            System.out.println(prop.getProperty("toc.pdfFilename"));
                        } catch (IOException ex) {
                        ex.printStackTrace();
                        }
                return(prop.getProperty("toc.pdfFilename").toString());
                }

        public String getTOCHTMLFileName() {
                Properties prop = new Properties();

                try (InputStream input = new FileInputStream("config.properties")) {
            prop.load(input);
            System.out.println(prop.getProperty("toc.htmFilename"));
                        } catch (IOException ex) {
                        ex.printStackTrace();
                        }
                return(prop.getProperty("toc.htmFilename").toString());
                }
        }
```

==========

SwingInterface.java code

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.awt.event.ActionEvent;
import java.awt.Color;
import java.awt.Component;
import javax.swing.Box;

public class SwingInterface {

        private JFrame frmHtmltocforPdfFiles;
        private JTextField textSplitFilePath;
```

```
	private JTextField textComposerName;
	private JTextField textBookTitle;
	private JTextField textPDFFileName;
	private JTextField texTOChtmlFileName;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					SwingInterface window = new SwingInterface();
					window.frmHtmltocforPdfFiles.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public SwingInterface() {
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frmHtmltocforPdfFiles = new JFrame();
		frmHtmltocforPdfFiles.setTitle("HTML-TOC-For PDF Files");
		frmHtmltocforPdfFiles.setBounds(100, 100, 818, 571);
		frmHtmltocforPdfFiles.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frmHtmltocforPdfFiles.getContentPane().setLayout(null);

		JLabel lblNewLabel = new JLabel("Name of TEXT file being referenced for
splitting into TOC arrays of file name, title, page:\r\n");
		lblNewLabel.setForeground(new Color(255, 0, 0));
		lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblNewLabel.setBounds(10, 378, 555, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblNewLabel);
```

```
		textSplitFilePath = new JTextField();
		textSplitFilePath.setFont(new Font("Tahoma", Font.BOLD, 12));
		textSplitFilePath.setBounds(43, 450, 719, 27);
		frmHtmltocforPdfFiles.getContentPane().add(textSplitFilePath);
		textSplitFilePath.setColumns(10);

		JButton btnFileChooser = new JButton("3. Select The TOC Splitter File ( |
parser)");
		btnFileChooser.setBackground(new Color(255, 128, 128));
		btnFileChooser.setFont(new Font("Tahoma", Font.BOLD, 12));
		btnFileChooser.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				// btnFileChooser clicked
				//Bach-7-Toccaten-BWV-910-916-HIGHQuality.txt
				FileChooser fileChooser = new FileChooser();
				File fSplitterFile = fileChooser.chooseDirectory(null);
				String splitterFile = fSplitterFile.toString();
				textSplitFilePath.setText(splitterFile);
				textPDFFileName.setText(fSplitterFile.getName());
			}
		});
		btnFileChooser.setBounds(43, 411, 290, 27);
		frmHtmltocforPdfFiles.getContentPane().add(btnFileChooser);

		JButton btnWriteTOCFile = new JButton("4. Write TOC File");
		btnWriteTOCFile.setBackground(new Color(255, 128, 128));
		btnWriteTOCFile.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				// btnWriteTOCFile clicked
				Director director = new Director();
				try {
					director.loadTOCLines(textSplitFilePath.getText());
				} catch (IOException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
			}
		});
		btnWriteTOCFile.setFont(new Font("Tahoma", Font.BOLD, 12));
		btnWriteTOCFile.setBounds(43, 488, 180, 33);
		frmHtmltocforPdfFiles.getContentPane().add(btnWriteTOCFile);

		JLabel lblNameOfComposer = new JLabel("COMPOSER name:");
```

```
		lblNameOfComposer.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblNameOfComposer.setBounds(10, 26, 180, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblNameOfComposer);

		textComposerName = new JTextField();
		textComposerName.setFont(new Font("Tahoma", Font.BOLD, 12));
		textComposerName.setBounds(10, 51, 727, 33);
		frmHtmltocforPdfFiles.getContentPane().add(textComposerName);
		textComposerName.setColumns(10);

		JLabel lblNameOfBook = new JLabel("BOOK title:");
		lblNameOfBook.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblNameOfBook.setBounds(10, 95, 180, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblNameOfBook);

		textBookTitle = new JTextField();
		textBookTitle.setFont(new Font("Tahoma", Font.BOLD, 12));
		textBookTitle.setBounds(10, 119, 727, 33);
		frmHtmltocforPdfFiles.getContentPane().add(textBookTitle);
		textBookTitle.setColumns(10);

		JLabel lblPdfFileName = new JLabel("PDF FILE Name (with .pdf suffix):");
		lblPdfFileName.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblPdfFileName.setBounds(10, 165, 226, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblPdfFileName);

		JLabel lblPropertiesForThe = new JLabel("Properties for the description and
titles (4 each):\r\n");
		lblPropertiesForThe.setForeground(new Color(255, 0, 0));
		lblPropertiesForThe.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblPropertiesForThe.setBounds(10, 0, 555, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblPropertiesForThe);

		textPDFFileName = new JTextField();
		textPDFFileName.setFont(new Font("Tahoma", Font.BOLD, 12));
		textPDFFileName.setBounds(10, 189, 727, 33);
		frmHtmltocforPdfFiles.getContentPane().add(textPDFFileName);
		textPDFFileName.setColumns(10);

		JLabel lblHtmlFileName = new JLabel("HTML FILE Name (with TOC- prefix
and .HTML suffix):");
		lblHtmlFileName.setFont(new Font("Tahoma", Font.BOLD, 12));
		lblHtmlFileName.setBounds(10, 233, 369, 27);
		frmHtmltocforPdfFiles.getContentPane().add(lblHtmlFileName);
```

```
		texTOChtmlFileName = new JTextField();
		texTOChtmlFileName.setFont(new Font("Tahoma", Font.BOLD, 12));
		texTOChtmlFileName.setBounds(10, 259, 727, 27);
		frmHtmltocforPdfFiles.getContentPane().add(texTOChtmlFileName);
		texTOChtmlFileName.setColumns(10);

		JButton btnSelectPDFFile = new JButton("1. Select Subject PDF File");
		btnSelectPDFFile.setFont(new Font("Tahoma", Font.BOLD, 12));
		btnSelectPDFFile.setBackground(new Color(255, 128, 128));
		btnSelectPDFFile.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				// btnSelectPDFFile clicked
				FileChooser fileChooser = new FileChooser();
				File fPDFfile = fileChooser.chooseDirectory(null);
				textPDFFileName.setText(fPDFfile.getName());
			}
		});
		btnSelectPDFFile.setBounds(259, 163, 206, 23);
		frmHtmltocforPdfFiles.getContentPane().add(btnSelectPDFFile);

		JButton btnSaveProperties = new JButton("2. Save Properties");
		btnSaveProperties.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				// btnSaveProperties clicked
				PropertiesManagement properties = new
PropertiesManagement();
				properties.saveProperties(textComposerName.getText(),
textBookTitle.getText(), textPDFFileName.getText(), texTOChtmlFileName.getText());
			}
		});
		btnSaveProperties.setBackground(new Color(255, 128, 128));
		btnSaveProperties.setFont(new Font("Tahoma", Font.BOLD, 12));
		btnSaveProperties.setBounds(10, 297, 161, 31);
		frmHtmltocforPdfFiles.getContentPane().add(btnSaveProperties);

		Component horizontalStrut = Box.createHorizontalStrut(20);
		horizontalStrut.setBounds(10, 339, 782, 15);
		frmHtmltocforPdfFiles.getContentPane().add(horizontalStrut);
	}
}
```

==========
TOCLine.java code

```
public class TOCLine {

	private String pdfFile;
	private String title;
	private String page;

	public void addLine (String pPDFfile, String pTitle, String pPage) {
		// constructor returns object
		this.pdfFile = pPDFfile;
		this.title = pTitle;
		this.page = pPage;
	}

	public String getPDFfile() {
		return this.pdfFile;
	}

	public String getTitle() {
		return this.title;
	}

	public String getPage() {
		return this.page;
	}


	@Override
  public String toString() {
    return "KeyValuePair [pdfFile=" + pdfFile + ", title=" + title + " page= " + page + "]";
  }
}
```

==========
WriteFile.java code

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
```

```
public class WriteFile {

   public void writePreamble(String pFileName) throws IOException {
         PropertiesManagement properties = new PropertiesManagement();
      FileWriter fileWriter = new FileWriter(pFileName, false);
      BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

      bufferedWriter.write("<!doctype html><html>\n");
      bufferedWriter.write("<head>\n");
      bufferedWriter.write("<meta charset=\"UTF-8\">\n");
      bufferedWriter.write("<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\">\n");
      bufferedWriter.write("\n");
      bufferedWriter.write("<style type=\"text/css\">\n");
      bufferedWriter.write(".container p { margin-left:25%;}\n");
      bufferedWriter.write("table, th, td {\n");
      bufferedWriter.write("font-size: 20px;\n");
      bufferedWriter.write("border: 1px solid white;\n");
      bufferedWriter.write("border-collapse: collapse; }\n");
      bufferedWriter.write("th, td {  background-color: lightgrey;}\n");
      bufferedWriter.write("table.center {\n");
      bufferedWriter.write("margin-left: auto;\n");
      bufferedWriter.write("margin-right: auto;\n");
      bufferedWriter.write("}\n");
      bufferedWriter.write(" \n");
      bufferedWriter.write(".overlay {\n");
      bufferedWriter.write("height: 100%;\n");
      bufferedWriter.write("width: 0;\n");
      bufferedWriter.write("position: fixed;\n");
      bufferedWriter.write("z-index: 1;\n");
      bufferedWriter.write("top: 0;\n");
      bufferedWriter.write("left: 0;\n");
      bufferedWriter.write("background-color: rgb(0,0,0);\n");
      bufferedWriter.write("background-color: rgba(0,0,0, 0.9);\n");
      bufferedWriter.write("overflow-x: hidden;\n");
      bufferedWriter.write("transition: 0.5s;\n");
      bufferedWriter.write("}\n");
      bufferedWriter.write(" \n");
      bufferedWriter.write(".overlay-content {\n");
      bufferedWriter.write("position: relative;\n");
      bufferedWriter.write("top: 10%;\n");
      bufferedWriter.write("width: 100%;\n");
      bufferedWriter.write("text-align: left;\n");
```

```
    bufferedWriter.write("margin-top: 5px;\n");
    bufferedWriter.write(" \n");
        bufferedWriter.write(" \n");
        bufferedWriter.write(".overlay a {\n");
        bufferedWriter.write("padding: 8px;\n");
        bufferedWriter.write("text-decoration: none;\n");
        bufferedWriter.write("font-size: 20px;\n");
        bufferedWriter.write("color: #818181;\n");
        bufferedWriter.write("display: block;\n");
        bufferedWriter.write("transition: 0.3s;\n");
        bufferedWriter.write("}\n");
        bufferedWriter.write("\n");
        bufferedWriter.write(".overlay a:hover, .overlay a:focus {\n");
        bufferedWriter.write("color: #f1f1f1;\n");
        bufferedWriter.write("}\n");
        bufferedWriter.write("\n");
        bufferedWriter.write(".overlay .closebtn {\n");
        bufferedWriter.write("position: absolute;\n");
        bufferedWriter.write("top: 10px;\n");
        bufferedWriter.write("right: 45px;\n");
        bufferedWriter.write("font-size: 60px;\n");
        bufferedWriter.write("}\n");
        bufferedWriter.write("\n");
        bufferedWriter.write("@media screen and (max-height: 450px) {\n");
        bufferedWriter.write(".overlay a {font-size: 20px}\n");
        bufferedWriter.write(".overlay .closebtn {\n");
        bufferedWriter.write("font-size: 20px;\n");
        bufferedWriter.write("top: 10px;\n");
        bufferedWriter.write("right: 35px;\n");
        bufferedWriter.write("}\n");
        bufferedWriter.write("}\n");
        bufferedWriter.write("</style>\n");
        bufferedWriter.write("\n");
        bufferedWriter.write("</head>\n");
        bufferedWriter.write("<title>" + properties.getTOCHTMLFileName() + "</title>");
        bufferedWriter.write("<html>\n");
        bufferedWriter.write("<body>\n");
        bufferedWriter.write("<table class=\"center\">\n");
        bufferedWriter.write("<tr><th colspan='3'>Table of Contents</th></tr>\n");
        bufferedWriter.write("<tr><th colspan='3'>" + properties.getTOCComposer() +
"</th></tr>\n");
        bufferedWriter.write("<tr><th colspan='3'>" + properties.getTOCBookTitle() +
"</th></tr>\n");
        bufferedWriter.write("<tr><th colspan='3'> </th></tr>\n");
```

```
        bufferedWriter.write("<tr><th>Item No.</th><th>" +
properties.getTOCPDFFileName() + "<br>Sheet Music
Contents</th><th>Page</th></tr>\n");
        bufferedWriter.write("\n");
       System.out.println("File '" + pFileName + "' created and content written
successfully.");
      bufferedWriter.close();
  }

  public void writeBody(String pFileName, ArrayList<TOCLine> arrTOCList) throws
IOException {

        FileWriter fileWriter = new FileWriter(pFileName, true);
        BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

// <tr><td>3. </td><td style="font-weight:bold;"><a href="Avishai-Cohen-Shifting-Sands-
Songbook.pdf#page=9" target="_empty">3. Window</a></td><td>9</td></tr>
System.out.println("IN FOR NEXT");
        for(int i = 0; i < arrTOCList.size(); i++) {
                System.out.println("IN FOR NEXT. arrTOCList.size = " + arrTOCList.size());
                        bufferedWriter.write("<tr><td>" + (i + 1) + "</td>");
                        bufferedWriter.write("<td style=\"font-weight:bold;\"><a href=\"" +
arrTOCList.get(i).getPDFfile() + "#page=" + arrTOCList.get(i).getPage() + "\" ");
                        bufferedWriter.write("target=\"_empty\">" + arrTOCList.get(i).getTitle()
+ "</a></td><td>" + arrTOCList.get(i).getPage() + "</td></tr>\n");
                }
        bufferedWriter.write("</table></body></html>");
                bufferedWriter.close();

  }

}

//end
```