Lab Answer Key: Module 11: Integrating with Unmanaged Code

Lab: Upgrading the Grades Report

Exercise 1: Generating the Grades Report by Using Word

Task 1: Examine the WordWrapper class that provides a functional wrapper around the dynamic (COM) API for Word

1. Start the MSL-TMG1 virtual machine if it is not already running.
2. Start the 20483B-SEA-DEV11 virtual machine.
3. Log on to Windows® 8 as **Student** with the password **Pa$$w0rd**. If necessary, click **Switch User** to display the list of users.
4. Switch to the Windows 8 **Start** window and then type Explorer.
5. In the **Apps** list, click **File Explorer**.
6. In File Explorer, navigate to the **E:\Mod11\Labfiles\Databases** folder, and then double-click **SetupSchoolGradesDB.cmd**.
7. Close File Explorer.
8. Switch to the Windows 8 **Start** window.
9. Click **Visual Studio 2012**.
10. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
11. In the **Open Project** dialog box, browse to **E:\Mod11\Labfiles\Starter\Exercise 1**, click **Grades.sln**, and then click **Open**.
12. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.

13. In the **Solutions ‘Grades’ Properties Pages** dialog box, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

14. In Solution Explorer, expand **Grades.Utilities**, and then double-click **WordWrapper.cs**.

15. Examine the code that is currently contained within this class.

16. On the **View** menu, click **Task List.**

17. In the **Task List** window, in the **Categories** list, click **Comments**.

18. Double-click the **TODO:Exercise 1: Task 1a: Create a dynamic variable called _word for activating Word** task.

19. In the code editor, click in the blank line below the comment, and then type the following code:

```
dynamic _word = null;
```

20. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1b: Instantiate _word as a new Word Application object** task.

21. In the code editor, click in the blank line below the comment, and then type the following code:

```
this._word = new Application { Visible = false };
```

22. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Create a new Word document** task.

23. In the code editor, click in the blank line below the comment, and then type the following code:

```
var doc = this._word.Documents.Add();
doc.Activate();
```

24. In the **Task List** window, double-click **TODO: Exercise 1: Task 1d: Save the document using the specified filename.** task.

25. In the code editor, click in the blank line below the comment, and then type the following code:

```
var currentDocument = this._word.ActiveDocument;
currentDocument.SaveAs(filePath);
```

26. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1e: Close the document** task.

27. In the code editor, click in the blank line below the comment, and then type the following code:

```
currentDocument.Close();
```

Task 2: Review the code in the GeneratedStudentReport method to generate a Word document

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2a: Generate a student grade report as a Word document.** task.

2. Examine the code that is in this method to generate the student report.

3. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2b: Generate the report by using a separate task**.

4. In the code editor, click in the blank line below the comment, and then type the following code:

```
Task.Run(() =>
GenerateStudentReport(SessionContext.CurrentStudent,
      dialog.FileName));
```

Task 3: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application loads, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
4. Click **Kevin Liu**, and then click **save report**.
5. In the **Save As** dialog box, browse to **E:\Mod11\Labfiles\Starter\Exercise 1**.
6. In the **File name** box, delete the existing contents, type **Kevin Liu Grades Report**, and then click **Save**.
7. Close the application, and then in Microsoft® Visual Studio®, on the **File** menu, click **Close Solution**.
8. Open File Explorer, browse to the **E:\Mod11\Labfiles\Starter\Exercise 1** folder, and then verify that the report has been generated.
9. Double-click **Kevin Liu Grades Report.docx**.
10. Review the grade report, and then close Word.

Results: After completing this exercise, the application will generate grade reports in Word format.

Exercise 2: Controlling the Lifetime of Word Objects by Implementing the Dispose Pattern

Task 1: Run the application to generate a grades report and view the Word task in Task Manager

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod11\Labfiles\Starter\Exercise 2**, click **Grades.sln**, and then click **Open**.
3. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.
4. In the **Solutions 'Grades' Properties Pages** dialog box, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.
5. On the **Build** menu, click **Build Solution**.
6. On the **Debug** menu, click **Start Without Debugging**.
7. When the application loads, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
8. Click **Kevin Liu**, and then click **save report**.
9. In the **Save As** dialog box, browse to **E:\Mod11\Labfiles\Starter\Exercise 2**.
10. In the **File name** box, delete the existing contents, type **Kevin Liu Grades Report**, and then click **Save**.
11. Close the application.
12. Open File Explorer, browse to the **E:\Mod11\Labfiles\Starter\Exercise 2** folder, and then verify that the report has been generated.
13. Right-click the **taskbar**, and then click **Task Manager**.
14. In the **Task Manager** window, click **More details**.
15. In the **Name** column, in the **Background processes** group, verify that **Microsoft Word (32 bit)** is still running.
16. Click **Microsoft Word (32 bit)**, and then click **End task**.

17. Close Task Manager.

Task 2: Update the WordWrapper class to terminate Word correctly

1. In Visual Studio, in the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Specify that the WordWrapper class implements the IDisposable interface** task.
2. In the code editor, on the line below the comment, click at the end of the **public class WordWrapper** code, and then type the following code:

```
:IDisposable
```

3. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Create the protected Dispose(bool) method** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

```
protected virtual void Dispose(bool isDisposing)
{
      if (!this.isDisposed)
      {


            if (isDisposing)
            {
                  // Release managed resources here
                  if (this._word != null)
                  {
                      this._word.Quit();
                  }
             }
             // Release unmanaged resources here
```

```
                if (this._word != null)
                {

System.Runtime.InteropServices.Marshal.ReleaseComObject(this._w
ord);
                }
                this.isDisposed = true;
            }
        }
```

5. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2c: Create the public Dispose method** task.

6. In the code editor, click at the end of the comment, press Enter, and then type the following code:

```
public void Dispose()
{
       this.Dispose(true);
       GC.SuppressFinalize(this);
}
```

7. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2d: Create a finalizer that calls the Dispose method** task.

8. In the code editor, click in the blank line below the comment, and then type the following code:

```
private bool isDisposed = false;
```

Task 3: Wrap the object that generates the Word doc in a using statement

1. In the **Task List** window, double-click the **TODO: Exercise 2: Task 3: Ensure that the WordWrapper is disposed when the method finishes** task.

2. Below the comment, modify the **WordWrapper wrapper = new WordWrapper();** code to look like the following:

```
using (var wrapper = new WordWrapper())
{
```

3. At the end of the method, after the **wrapper.SaveAs(reportPath);** line of code, add a closing brace to end the **using** block.

4. Your code should look like the following:

```
public void GenerateStudentReport(LocalStudent studentData,
string reportPath)
{
    // TODO: Exercise 2: Task 3: Ensure that the WordWrapper is
disposed when the
method finishes
    using (var wrapper = new WordWrapper())
    {
        // Create a new Word document in memory
        wrapper.CreateBlankDocument();
        // Add a heading to the document
wrapper.AppendHeading(String.Format("Grade Report: {0} {1}",
studentData.FirstName,
studentData.LastName));


        wrapper.InsertCarriageReturn();
        wrapper.InsertCarriageReturn();
        // Output the details of each grade for the student
        foreach (var grade in SessionContext.CurrentGrades)
        {
wrapper.AppendText(grade.SubjectName, true, true);
```

```
                wrapper.InsertCarriageReturn();
                wrapper.AppendText("Assessment: " +
grade.Assessment, false, false);
                wrapper.InsertCarriageReturn();
                wrapper.AppendText("Date: " +
grade.AssessmentDateString, false, false);
                wrapper.InsertCarriageReturn();
                wrapper.AppendText("Comment: " + grade.Comments,
false, false);
                wrapper.InsertCarriageReturn();
                wrapper.InsertCarriageReturn();
            }
            // Save the Word document
            wrapper.SaveAs(reportPath);
}
}
```

Task 4: Use Task Manager to observe that Word terminates correctly after generating a report

1. On the **Build** menu, click **Build Solution**.
2. Right-click the **taskbar**, and then click **Task Manager**.
3. In Visual Studio, on the **Debug** menu, click **Start Without Debugging**.
4. When the application loads, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
5. Click **George Li**, and then click **save report**.
6. In the **Save As** dialog box, browse to **E:\Mod11\Labfiles\Starter\Exercise 2**.
7. In the **File name** box, delete the existing contents, and then type **George Li Grades Report**.
8. As you click **Save**, in the **Task Manager** window, watch the **Background**

processes and verify that **Microsoft Word (32 bit)** appears and then disappears from the list.

9. Close Task Manager, and then close the application.

10. In Visual Studio, on the **File** menu, click **Close Solution**.

Results: After completing this exercise, the application will terminate Word correctly after it has generated a grades report.