Lab Answer Key: Module 12: Creating Reusable Types and Assemblies

Lab: Specifying the Data to Include in the Grades Report

Exercise 1: Creating and Applying the IncludeInReport attribute

Task 1: Write the code for the IncludeInReportAttribute class

1. Start the 20483B-SEA-DEV11 virtual machine.
2. Log on to Windows® 8 as **Student** with the password **Pa$$w0rd**. If necessary, click **Switch User** to display the list of users.
3. Switch to the Windows 8 **Start** window and then type Explorer.
4. In the **Apps** list, click **File Explorer**.
5. Navigate to the **E:\Mod12\Labfiles\Databases** folder, and then double-click **SetupSchoolGradesDB.cmd**.
6. Close File Explorer.
7. Switch to the Windows 8 **Start** window.
8. Click **Visual Studio 2012**.
9. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
10. In the **Open Project** dialog box, browse to **E:\Mod12\Labfiles\Starter\Exercise 1**, click **Grades.sln**, and then click **Open**.
11. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.
12. On the **Startup Project** page, click **Multiple startup projects**. Set

Grades.Web and **Grades.WPF** to **Start without debugging**, and then click **OK**.

13. In Solution Explorer, expand **Grades.Utilities**, and then double-click **IncludeInReport.cs**.
14. On the **View** menu, click **Task List**.
15. In the **Task List** window, in the **Categories** list, click **Comments**.
16. Double-click the **TODO: Exercise 1: Task 1a: Specify that IncludeInReportAttribute is an attribute class** task.
17. In the code editor, below the comment, click at the end of the public **public class IncludeInReportAttribute** code, and then type the following code:

```
: Attribute
```

18. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1b: Specify the possible targets to which the IncludeInReport attribute can be applied** task.
19. In the code editor, click in the blank line below the comment, and then type the following code:

```
[AttributeUsage(AttributeTargets.Field |
AttributeTargets.Property, AllowMultiple =
false)]
```

20. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1c: Define a private field to hold the value of the attribute** task.
21. In the code editor, click in the blank line below the comment, and then type the following code:

```
private bool _include;
```

22. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1d: Add public properties that specify how an included item should be formatted** task.

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

```
public bool Underline { get; set; }
public bool Bold { get; set; }
```

24. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1e: Add a public property that specifies a label (if any) for the item** task.

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

```
public string Label { get; set; }
```

26. In the **Task List** window, double-click the **TODO: Exercise 1: Task 1f: Define constructors** task.

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

```
public IncludeInReportAttribute()
{
    this._include = true;
    this.Underline = false;
    this.Bold = false;
    this.Label = string.Empty;
}
public IncludeInReportAttribute(bool includeInReport)
{
    this._include = includeInReport;
    this.Underline = false;
```

```
        this.Bold = false;
        this.Label = string.Empty;
    }
```

Task 2: Apply the IncludeInReportAttribute attribute to the appropriate properties

1. In Solution Explorer, expand **Grades.WPF**, and then double-click **Data.cs**.
2. In the **Task List** window, double-click the **TODO: Exercise 1: Task 2: Add the IncludeInReport attribute to the appropriate properties in the LocalGrade class** task.
3. In the **LocalGrade** class, expand the **Properties** region, and then expand the **Readonly Properties** region.
4. Above the **public string SubjectName** code, click in the blank line, and then type the following code:

```
[IncludeInReport(Label="Subject Name", Bold=true,
Underline=true)]
```

5. Above the **public string AssessmentDateString** code, click in the blank line, press Enter, and then type the following code:

```
[IncludeInReport (Label="Date")]
```

6. Expand the **Form Properties** region.
7. Above the **public string Assessment** code, click in the blank line, press Enter, and then type the following code:

```
[IncludeInReport(Label = "Grade")]
```

8. Above the **public string Comments** code, click in the blank space, press Enter, and then type the following code:

```
[IncludeInReport(Label = "Comments")]
```

Task 3: Build the application and review the metadata for the LocalGrades class

1. On the **Build** menu, click **Build Solution**.
2. Open File Explorer and browse to the **C:\Program Files (x86)\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools** folder.
3. Right-click **ildasm.exe,** and then click **Open**.
4. In the **IL DASM** window, on the **File** menu, click **Open**.
5. In the **Open** dialog box, browse to **E:\Mod12\Labfiles\Starter\Exercise 1\Grades.WPF\bin\Debug**, click **Grades.WPF.exe**, and then click **Open**.
6. In the **IL DASM** application window, expand **Grades.WPF,** expand **Grades.WPF.LocalGrade**, and then double-click **Assessment : instance string();**.
7. In the **Grades.WPF.LocalGrade::Assessment : instance string()** window, in the **Assessment** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
8. In the **IL DASM** application window, double-click **AssessmentDateString : instance string();**.
9. In the **Grades.WPF.LocalGrade::AssessmentDateString : instance string()** window, in the **AssessmentDateString** method, verify that the **.custom instance**

void[Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor() code is present, and then close the window.

10. In the **IL DASM** application window, double-click **Comments : instance string();**.
11. In the **Grades.WPF.LocalGrade::Comments : instance string()** window, in the **Comments** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
12. In the **IL DASM** application window, double-click **SubjectName : instance string();**.
13. In the **Grades.WPF.LocalGrade::SubjectName : instance string()** window, in the **SubjectName** method, verify that the **.custom instance void [Grades.Utilities]Grades.Utilities.IncludeInReportAttribute::.ctor()** code is present, and then close the window.
14. Close the IL DASM application.
15. Close File Explorer.
16. In Visual Studio, on the **File** menu, click **Close Solution**.

Results: After completing this exercise, the **Grades.Utilities** assembly will contain an **IncludeInReport** custom attribute and the **Grades** class will contain fields and properties that are tagged with that attribute.

Exercise 2: Updating the Report

Task 1: Implement a static helper class called IncludeProcessor

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:\Mod12\Labfiles\Starter\Exercise 2**, click **Grades.sln**, and then click **Open**.

3. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.

4. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

5. In Solution Explorer, expand **Grades.Utilities**, and then double-click **IncludeInReport.cs**.

6. Below the **Output** window, click **Task List**.

7. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1a: Define a struct that specifies the formatting to apply to an item** task.

8. In the code editor, click in the blank line in the **FormatField** struct, and then type the following code:

```
public string Value;
public string Label;
public bool IsBold;
public bool IsUnderlined;
```

9. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1b: Find all the public fields and properties in the dataForReport object** task.

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

```
Type dataForReportType = dataForReport.GetType();
fieldsAndProperties.AddRange(dataForReportType.GetFields());
fieldsAndProperties.AddRange(dataForReportType.GetProperties())
;
```

11. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1c: Iterate**

through all public fields and properties, and process each item that is tagged with the IncludeInReport attribute task.

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

```
foreach (MemberInfo member in fieldsAndProperties)
{
```

13. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1d: Determine whether the current member is tagged with the IncludeInReport attribute** task.

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

```
object[] attributes = member.GetCustomAttributes(false);
IncludeInReportAttribute attributeFound =
Array.Find(attributes, a => a.GetType() ==
typeof(IncludeInReportAttribute)) as IncludeInReportAttribute;
```

15. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1e: If the member is tagged with the IncludeInReport attribute, construct a FormatField item** task.

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

```
if (attributeFound != null)
{
    // Find the value of the item tagged with the
IncludeInReport attribute


    string itemValue;
    if (member is FieldInfo)
    {
        itemValue = (member as
FieldInfo).GetValue(dataForReport).ToString();
    }
    else
    {
        itemValue = (member as
PropertyInfo).GetValue(dataForReport).ToString();
    }
```

17. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1f: Construct a FormatField item with this data** task.

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

```
FormatField item = new FormatField()
{
    Value = itemValue,
    Label = attributeFound.Label,
    IsBold = attributeFound.Bold,
    IsUnderlined = attributeFound.Underline
};
```

19. In the **Task List** window, double-click the **TODO: Exercise 2: Task 1g: Add the FormatField item to the collection to be returned** task.

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

```
        items.Add(item);
    }
}
```

Task 2: Update the report functionality for the StudentProfile view

1. In Solution Explorer, expand **Grades.WPF**, expand **Views,** expand **StudentProfile.xaml**, and then double-click **StudentProfile.xaml.cs**.
2. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2a: Use the IncludeProcessor to determine which fields in the Grade object are tagged** task.
3. In the code editor, click in the blank line below the comment, and then type the following code:

```
List<FormatField> itemsToReport =
IncludeProcessor.GetItemsToInclude(grade);
```

4. In the **Task List** window, double-click the **TODO: Exercise 2: Task 2b: Output each tagged item, using the format specified by the properties of the IncludeInReport attribute for each item** task.
5. In the code editor, click in the blank line below the comment, and then type the following code:

```
foreach (FormatField item in itemsToReport)
{
wrapper.AppendText(item.Label == string.Empty ? item.Value :
item.Label + ": " +
item.Value, item.IsBold, item.IsUnderlined);
wrapper.InsertCarriageReturn();
}
```

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. In the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
4. In the **Class 3C** view, click **Kevin Liu**.
5. Verify that the student report for Kevin Liu appears, and then click **save report**.
6. In the **Save As** dialog box, browse to the **E:\Mod12\Labfiles\Starter\Exercise 2** folder.
7. In the **File name** box, type **KevinLiuGradesReport**, and then click **Save**.
8. Close the application.
9. In Visual Studio, on the **File** menu, click **Close Solution**.
10. Open File Explorer, browse to **E:\Mod12\Labfiles\Starter\Exercise 2**, and then verify that KevinLiuGradesReport.docx has been generated.
11. Right-click **KevinLiuGradesReport.docx**, and then click **Open**.
12. Verify that the document contains the grade report for Kevin Liu and that it is correctly formatted, and then close Word.

Results: After completing this exercise, the application will be updated to use reflection to include only the tagged fields and properties in the grades report.

Exercise 3: Storing the Grades.Utilities Assembly Centrally (If Time Permits)

Task 1: Sign the Grades.Utilities assembly and deploy it to the GAC

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:\Mod12\Labfiles\Starter\Exercise 3**, click **Grades.sln**, and then click **Open**.

3. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.

4. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

5. Switch to the Windows 8 **Start** window.

6. In the **Start** window, right-click the background to display the taskbar.

7. On the taskbar, click **All apps**.

8. In the **Start** window, right-click the **VS2012 x86 Native Tools Command** icon.

9. On the taskbar, click **Run as administrator**.

10. In the **User Account Control** dialog box, in the **Password** box, type **Pa$$w0rd**, and then click **Yes**.

11. At the command prompt, type the following code, and then press Enter:

```
E:
```

12. At the command prompt, type the following code, and then press Enter:

```
cd E:\Mod12\Labfiles\Starter
```

13. At the command prompt, type the following code, and then press Enter:

```
sn -k GradesKey.snk
```

14. Verify that the text **Key pair written to GradesKey.snk** is displayed.

15. In Visual Studio, in Solution Explorer, right-click **Grades.Utilities**, and then click

Properties.

16. On the **Signing** tab, select **Sign the assembly**.

17. In the **Choose a strong name key file** list, click **Browse**.

18. In the **Select File** dialog box, browse to **E:\Mod12\Labfiles\Starter**, click **GradesKey.snk**, and then click **Open**.

19. On the **Build** menu, click **Build Solution**.

20. Switch to the command prompt, type the following code, and then press Enter:

```
cd E:\Mod12\Labfiles\Starter\Exercise
3\Grades.Utilities\bin\Debug
```

21. At the command prompt, type the following code, and then press Enter:

```
gacutil -i Grades.Utilities.dll
```

22. Verify that the text **Assembly successfully added to the cache** is displayed, and then close the Command Prompt window.

Task 2: Reference the Grades.Utilities assembly in the GAC from the application

1. In Visual Studio, in Solution Explorer, expand **Grades.WPF**, expand **References**, right-click **Grades.Utilities**, and then click **Remove**.

2. Right-click **References**, and then click **Add Reference**.

3. In the **Reference Manager – Grades.WPF** dialog box, click the **Browse** button.

4. In the **Select the files to reference** dialog box, browse to **E:\Mod12\Labfiles\Starter\Exercise 3\Grades.Utilities\bin\Debug**, click **Grades.Utilities.dll**, and then click **Add**.

5. In the **Reference Manager – Grades.WPF** dialog box, click **OK**.
6. On the **Build** menu, click **Build Solution**.
7. On the **Debug** menu, click **Start Without Debugging**.
8. In the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
9. In the **Class 3C** view, click **Kevin Liu**.
10. Verify that the student report for Kevin Liu appears, and then click **save report**.
11. In the **Save As** dialog box, browse to the **E:\Mod12\Labfiles\Starter\Exercise 3** folder.
12. In the **File name** box, type **KevinLiuGradesReport**, and then click **Save**.
13. Close the application.
14. In Visual Studio, on the **File** menu, click **Close Solution**.
15. Open File Explorer, browse to **E:\Mod12\Labfiles\Starter\Exercise 3**, and then verify that KevinLiuGradesReport.docx has been generated.
16. Right-click **KevinLiuGradesReport.docx**, and then click **Open**.
17. Verify that the document contains the grade report for Kevin Liu and that it is correctly formatted, and then close Word.

Results: After completing this exercise, you will have a signed version of the **Grades.Utilities** assembly deployed to the GAC.