✓ 100 XP

Exercise - Implement code branches using selection statements

25 minutes

In this exercise, you'll develop the code that automatically assigns a student's letter grade based on their final numeric score and you'll update the application so that extra credit project scores are factored into the student's final grade. You will begin by writing an `if-elseif-else` construct that can be used to evaluate the student's numeric score and assign the letter grade. Next, you'll examine the application requirements related to extra credit work, and then work your way through the required code updates. The detailed tasks that you'll complete during this exercise are:

1. Develop an `if-elseif-else` construct that evaluates the student's score to assign a letter grade. The expression that's evaluated compares the student's numeric score with a range of scores taken from a grading chart provided by the teacher.
2. Integrate extra credit scores into each student's scores array, and then update the code that's used to calculate the student's numeric score. The `foreach` that's used to sum the student scores will be updated to include an `if` statement that branches the code. The exam scores will be applied to the sum in one branch, and the extra credit scores in the other branch.

ⓘ **Important**

You need to have completed this module's previous Exercise, "Create arrays and foreach loops", before you begin this Exercise.

Assign letter grades using an `if-elseif-else` construct

In this task, you'll develop an `if-elseif-else` structure that can be used to assign letter grades based on a calculated numeric score.

1. Ensure that you have the Program.cs file open in the Visual Studio Code Editor.

2. Create a blank code line below the line used to declare `studentScores` array.

3. To create a string variable that can be used to hold the student's letter grade, enter the following code:

   C#

   ```csharp
   string currentStudentLetterGrade = "";
   ```

4. Scroll down to the bottom of the Program.cs file.

5. Add a blank code line below the line that assigns a calculated value to `currentStudentGrade`.

6. Take a minute to consider the grading chart that shows the letter grade corresponding to numeric scores.

   Output

   ```
   97 - 100   A+
   93 - 96    A
   90 - 92    A-
   87 - 89    B+
   83 - 86    B
   80 - 82    B-
   77 - 79    C+
   73 - 76    C
   70 - 72    C-
   67 - 69    D+
   63 - 66    D
   60 - 62    D-
   0  - 59    F
   ```

   Notice that the top row of scores, the values greater than or equal to 97, have a letter grade of "A+". In other words, if a student's final score is >= 97, they'll be assigned a letter grade of "A+".

7. To create an `if` statement that assigns `A+` to `currentStudentLetterGrade` when the student's score is greater than or equal to 97, enter the following code:

   C#

```
if (currentStudentGrade >= 97)
    currentStudentLetterGrade = "A+";
```

8. To create an `else if` statement that assigns `A` to `currentStudentLetterGrade` when the student's score is greater than or equal to 93, enter the following code:

   C#

   ```csharp
   else if (currentStudentGrade >= 93)
       currentStudentLetterGrade = "A";
   ```

   The `else if` will not assign `A` to `currentStudentLetterGrade` when the student's score is greater than or equal to 97 because that expression returned `true` in the preceding `if`.

   You can extend this `else if` pattern as you move down the rows of the letter grade chart. When you reach the end of the chart, you can use a final `else` to catch any `currentStudentGrade` that is below 60.

9. Create the `else if` statements that assign letter grades to `currentStudentLetterGrade` for the score ranges between 60 and 92.

   Once you've completed this step, you should have an `if` statement structure that matches the following code:

   C#

   ```csharp
       if (currentStudentGrade >= 97)
           currentStudentLetterGrade = "A+";

       else if (currentStudentGrade >= 93)
           currentStudentLetterGrade = "A";

       else if (currentStudentGrade >= 90)
           currentStudentLetterGrade = "A-";

       else if (currentStudentGrade >= 87)
           currentStudentLetterGrade = "B+";

       else if (currentStudentGrade >= 83)
           currentStudentLetterGrade = "B";

       else if (currentStudentGrade >= 80)
   ```

```
        currentStudentLetterGrade = "B-";

    else if (currentStudentGrade >= 77)
        currentStudentLetterGrade = "C+";

    else if (currentStudentGrade >= 73)
        currentStudentLetterGrade = "C";

    else if (currentStudentGrade >= 70)
        currentStudentLetterGrade = "C-";

    else if (currentStudentGrade >= 67)
        currentStudentLetterGrade = "D+";

    else if (currentStudentGrade >= 63)
        currentStudentLetterGrade = "D";

    else if (currentStudentGrade >= 60)
        currentStudentLetterGrade = "D-";
```

Your final step is to add the `else` that addresses any remaining scores.

10. To create the `else` that applies to scores below 60, enter the following code:

C#

```
    else
        currentStudentLetterGrade = "F";
```

11. Take a minute to review your application code.

Your Program.cs code should match the following code:

C#

```
// initialize variables - graded assignments
int currentAssignments = 5;

int[] sophiaScores = new int[] { 90, 86, 87, 98, 100 };
int[] andrewScores = new int[] { 92, 89, 81, 96, 90 };
int[] emmaScores = new int[] { 90, 85, 87, 98, 68 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96 };

// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };
```

```
int[] studentScores = new int[10];

string currentStudentLetterGrade = "";

// Display the Report Header
Console.WriteLine("Student\t\tGrade\n");

foreach (string name in studentNames)
{
    string currentStudent = name;

    if (currentStudent == "Sophia")
        // assign Sophia's scores to the studentScores array
        studentScores = sophiaScores;

    else if (currentStudent == "Andrew")
        // assign Andrew's scores to the studentScores array
        studentScores = andrewScores;

    else if (currentStudent == "Emma")
        // assign Emma's scores to the studentScores array
        studentScores = emmaScores;

    else if (currentStudent == "Logan")
        // assign Logan's scores to the studentScores array
        studentScores = loganScores;

    // initialize/reset the sum of scored assignments
    int sumAssignmentScores = 0;

    // initialize/reset the calculated average of exam + extra credit scores
    decimal currentStudentGrade = 0;

    foreach (int score in studentScores)
    {
        // add the exam score to the sum
        sumAssignmentScores += score;
    }

    currentStudentGrade = (decimal)(sumAssignmentScores) / currentAssignments;

    if (currentStudentGrade >= 97)
        currentStudentLetterGrade = "A+";

    else if (currentStudentGrade >= 93)
        currentStudentLetterGrade = "A";

    else if (currentStudentGrade >= 90)
        currentStudentLetterGrade = "A-";

    else if (currentStudentGrade >= 87)
```

```
            currentStudentLetterGrade = "B+";

        else if (currentStudentGrade >= 83)
            currentStudentLetterGrade = "B";

        else if (currentStudentGrade >= 80)
            currentStudentLetterGrade = "B-";

        else if (currentStudentGrade >= 77)
            currentStudentLetterGrade = "C+";

        else if (currentStudentGrade >= 73)
            currentStudentLetterGrade = "C";

        else if (currentStudentGrade >= 70)
            currentStudentLetterGrade = "C-";

        else if (currentStudentGrade >= 67)
            currentStudentLetterGrade = "D+";

        else if (currentStudentGrade >= 63)
            currentStudentLetterGrade = "D";

        else if (currentStudentGrade >= 60)
            currentStudentLetterGrade = "D-";

        else
            currentStudentLetterGrade = "F";

        Console.WriteLine($"{name}\t\t{currentStudentGrade}\t?");
    }

    Console.WriteLine("Press the Enter key to continue");
    Console.ReadLine();
```

Notice that your application is organized in a very logical top-to-bottom manner:

a. You initialize variables and create the arrays that serve as the data source for the application. You have arrays that supply student scores as well as an array that supplies the student names. You also have a student-agnostic array named `studentScores` that you can use to hold the scores of any student when it comes time to calculate the grades.

b. You have a `Console.WriteLine()` statement that writes the column labels for your grading report to the console.

c. You have an outer `foreach` loop that iterates through the `studentNames` array, providing you with a code block that repeats for each student.

d. You continue to organize your code using a top-to-bottom approach inside the code block of the outer `foreach` loop:

   i. You have an `if` statement to evaluate the name of the current student, for example `if (currentStudent == "Sophia")`. When the expression evaluates as `true`, you assign the student's scores array to your student-agnostic array, for example: `studentScores = sophiaScores;`

   ii. You declare the two variables that're required in order to calculate student grades. The first variable, `sumAssignmentScores`, is used to calculate the sum of assignment scores. The second variable, `currentStudentGrade`, is used to calculate final numeric grade. You initialize the variables with a value of `0`.

   iii. You have a `foreach` loop that iterates through `studentScores` to calculate the value of `sumAssignmentScores`.

   iv. You calculate `currentStudentGrade` by dividing `sumAssignmentScores` by the number of assignments in the grade book. The number of graded assignments is held in a variable named `currentAssignments`.

   v. You have an `if-elseif-else` construct that assigns letter grades based on the value of `currentStudentGrade`.

   vi. You have a `Console.WriteLine()` statement that writes student names and grades to the console in order to complete the grading report.

12. Locate the `Console.WriteLine()` statement that writes student names and grades to the console.

    C#

    ```csharp
    Console.WriteLine($"{currentStudent}\t\t{currentStudentGrade}\t?");
    ```

    Notice that you still need to include the calculated letter grade in the grading report.

13. To include the value of `currentStudentLetterGrade` in the grading report, update your code as follows:

    C#

```
Console.WriteLine($"{currentStudent}\t\t{currentStudentGrade}\t{currentStu-
dentLetterGrade}");
```

14. On the Visual Studio Code **File** menu, click **Save**.
15. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.
16. At the Terminal command prompt, type **dotnet build** and then press Enter.
17. If you see Error or Warning messages, you need to fix them before continuing.
18. At the Terminal command prompt, type **dotnet run** and then press Enter.
19. Verify that your code produced the following output:

Output

```
Student         Grade

Sophia          92.2    A-
Andrew          89.6    B+
Emma            85.6    B
Logan           91.2    A-
Press the Enter key to continue
```

Your application is really coming together. Now you need to integrate extra credit assignments.

Integrate extra credit scores within a code branch

In this task, you'll update the application to accommodate extra credit work that has been turned in by students. Students complete extra credit projects to earn bonus points that can help bring their grade up. The teacher has provided you with extra credit scores for each student based on what the students have turned in:

- Sophia: 94, 90
- Andrew: 89
- Emma: 89, 89, 89
- Logan: 96

You will be using these extra credit scores and the application requirements provided by the teacher to complete this task.

1. Take a minute to consider the project requirements related to extra credit assignments.

   The "Prepare" unit for this Guided project module includes a **Project overview** section that includes the following requirements:

   - Your application needs to accommodate extra credit assignments.
     - Extra credit scores must be included in the student's scores array.
     - Extra credit assignments will be worth 10% of an exam score when applied toward the final numeric grade.
     - Extra credit assignment scores must be added to the student's total exam score before calculating the final numeric grade.

   - Integrate extra credit scores when calculating the student's final numeric and letter grade as follows:
     - Your code must detect extra credit assignments based on the number of elements in the student's scores array.
     - Your code must apply the 10% weighting factor to extra credit assignments before adding extra credit scores to the sum of exam scores.

2. Scroll to the top of your Program.cs file.

3. To add Sophia's extra credit assignment scores to the `sophiaScores` array, update your code as follows:

   C#

   ```
   int[] sophiaScores = new int[] { 90, 86, 87, 98, 100, 94, 90 };
   ```

   Notice that you've added the extra credit scores, `94` and `90`, to the list of scores included in the array. Simple.

4. Add the extra credit scores for the other students to their scores arrays.

5. Ensure that the student scores arrays match the following code:

   C#

   ```
   int[] sophiaScores = new int[] { 90, 86, 87, 98, 100, 94, 90 };
   int[] andrewScores = new int[] { 92, 89, 81, 96, 90, 89 };
   ```

```
int[] emmaScores = new int[] { 90, 85, 87, 98, 68, 89, 89, 89 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96, 96 };
```

6. Scroll down to locate the inner `foreach` loop that's used to sum assignment scores.

C#

```csharp
foreach (int score in studentScores)
{
    // add the exam score to the sum
    sumAssignmentScores += score;
}
```

7. Take a minute to consider the updates that you need to implement.

   First, consider what you already know:

   - You know that a `foreach` loop will iterate sequentially through all elements of an array regardless of how many elements the array contains.
   - You know that the students have five exam scores, and that you have a related variable: `int currentAssignments = 5;`.
   - You know that the extra credit scores are included at the end of the array.
   - You know that extra credit scores are worth 10% of an exam score.
   - You know that extra credit scores must be added to the sum of exam scores before calculating the student's final numeric grade.

   Now consider what you need:

   - You need to detect which scores in the scores array are the extra credit scores.
   - You need to adjust the value of any extra credit scores so that they are worth 10% of an exam score.
   - You need to update the calculation that's used to sum student scores so that the sum includes the extra credit scores.

8. Identify the coding updates required to differentiate between exam scores and extra credit scores.

   You know that the extra credit scores are listed after the five exam scores. In other words, the first extra credit score will be the sixth score in the scores array. This relationship between the score type and the array element number tells you that you need a counter inside the `foreach` loop. Once the value of your counter is greater than the number of exam scores, you know that the current score is an extra credit score.

Here's what you need to implement in order to differentiate between exam scores and extra credit scores:

- You need to declare an integer above the inner `foreach` loop that can be used to count graded assignments. You can name this variable `gradedAssignments`.
- You need to increment `gradedAssignments` by `1` inside the `foreach` loop. If you initialize `gradedAssignments` to `0`, then you can increment your counter at the top of the `foreach` code block.
- You need an `if` statement that evaluates whether your counter, `gradedAssignments`, is greater than the number of exam grades. The variable that holds the number of exam assignments is named `currentAssignments`. This name could cause confusion now that you have extra credit assignments in addition to exam assignments. You should change the variable name from `currentAssignments` to `examAssignments`. Once this name change is implemented, you can use your `if` to evaluate `(gradedAssignments <= examAssignments)`.

9. Change the variable name from `currentAssignments` to `examAssignments`.

> **Important**
>
> When changing a variable name, you need to ensure that you update all instances of the variable in your application. In this case, there are two instances.

The Visual Studio Code Editor panel support using the keyboard shortcut **Control** + **F** to find the text that you specify. The Visual Studio Code Editor panel also supports using the keyboard shortcut **Control** + **H** to find and replace the text that you specify.

10. Create a blank code line above the `foreach` loop that's used to sum assignment scores.

11. On the blank code line, to declare an integer variable named `gradedAssignments` and initialize it to `0`, enter the following code:

C#

```
// initialize/reset a counter for the number of assignments
int gradedAssignments = 0;
```

12. Create a blank code line at the top of the code block for the `foreach` loop that's used to sum assignment scores.

13. On the blank code line, to increment `gradedAssignments` by `1` for each iteration of the `foreach` loop, enter the following code:

    C#

    ```
    // increment the assignment counter
    gradedAssignments += 1;
    ```

14. To create an `if` statement that evaluates the expression `(gradedAssignments <= examAssignments)`, enter the following code:

    C#

    ```
    if (gradedAssignments <= examAssignments)
    ```

15. Identify the coding updates required for the calculation that's used to sum student scores.

    When your `if` statement evaluates `(gradedAssignments <= examAssignments)` as `true`, the score is an exam score, and you can add the value to your sum. If the expression does not evaluate as `true`, then the score is an extra credit score, and you need to divide it by 10 before you can add the value to your sum. An `if-else` construct will be perfect.

16. Notice that the existing equation, `sumAssignmentScores += score;`, is the correct calculation to use when your `if` statement evaluates `(gradedAssignments <= examAssignments)` as `true`.

17. Create a blank code line below `sumAssignmentScores += score;`.

18. On the blank code line, to construct the `else` portion of the `if-else` construct, type **else** and then press Enter.

19. To create the equation that adds an extra credit score to the sum, enter the following code:

    C#

    ```
    // add the extra credit points to the sum - bonus points equal to 10% of an exam
    score
    sumAssignmentScores += score / 10;
    ```

20. On the Visual Studio Code **File** menu, click **Save**.

21. Take a minute to review your application code.

    Ensure that your updated application matches the following code:

C#

```csharp
// initialize variables - graded assignments
int examAssignments = 5;

int[] sophiaScores = new int[] { 90, 86, 87, 98, 100, 94, 90 };
int[] andrewScores = new int[] { 92, 89, 81, 96, 90, 89 };
int[] emmaScores = new int[] { 90, 85, 87, 98, 68, 89, 89, 89 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96, 96 };

// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan" };

int[] studentScores = new int[10];

string currentStudentLetterGrade = "";

// Write the Report Header to the console
Console.WriteLine("Student\t\tGrade\n");

foreach (string name in studentNames)
{
    string currentStudent = name;

    if (currentStudent == "Sophia")
       studentScores = sophiaScores;

    else if (currentStudent == "Andrew")
        studentScores = andrewScores;

    else if (currentStudent == "Emma")
        studentScores = emmaScores;

    else if (currentStudent == "Logan")
        studentScores = loganScores;

    // initialize/reset the sum of scored assignments
    int sumAssignmentScores = 0;

    // initialize/reset the calculated average of exam + extra credit scores
    decimal currentStudentGrade = 0;

    // initialize/reset a counter for the number of assignment
    int gradedAssignments = 0;
    // loop through the scores array and complete calculations for currentStu-
dent
    foreach (int score in studentScores)
    {
        // increment the assignment counter
        gradedAssignments += 1;

        if (gradedAssignments <= examAssignments)
            // add the exam score to the sum
            sumAssignmentScores += score;

        else
            // add the extra credit points to the sum - bonus points equal to
10% of an exam score
            sumAssignmentScores += score / 10;
    }

    currentStudentGrade = (decimal)(sumAssignmentScores) / examAssignments;

    if (currentStudentGrade >= 97)
        currentStudentLetterGrade = "A+";

    else if (currentStudentGrade >= 93)
        currentStudentLetterGrade = "A";

    else if (currentStudentGrade >= 90)
        currentStudentLetterGrade = "A-";

    else if (currentStudentGrade >= 87)
        currentStudentLetterGrade = "B+";

    else if (currentStudentGrade >= 83)
        currentStudentLetterGrade = "B";

    else if (currentStudentGrade >= 80)
        currentStudentLetterGrade = "B-";

    else if (currentStudentGrade >= 77)
        currentStudentLetterGrade = "C+";

    else if (currentStudentGrade >= 73)
        currentStudentLetterGrade = "C";

    else if (currentStudentGrade >= 70)
        currentStudentLetterGrade = "C-";

    else if (currentStudentGrade >= 67)
        currentStudentLetterGrade = "D+";

    else if (currentStudentGrade >= 63)
```

```
            currentStudentLetterGrade = "D";

        else if (currentStudentGrade >= 60)
            currentStudentLetterGrade = "D-";

        else
            currentStudentLetterGrade = "F";

        //Console.WriteLine("Student\t\tGrade\tLetter Grade\n");
        Console.WriteLine($"{currentStudent}\t\t{currentStudentGrade}\t{currentStu-
dentLetterGrade}");
    }

    // required for running in VS Code (keeps the Output windows open to view re-
    sults)
    Console.WriteLine("\n\rPress the Enter key to continue");
    Console.ReadLine();
```

Check your work

In this task, you'll run the application to verify that your code logic is working as expected.

1. Ensure that you've saved your changes to the Program.cs file.
2. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.
3. At the Terminal command prompt, type **dotnet build** and then press Enter.
4. If you see Error or Warning messages, you need to fix them before continuing.
5. At the Terminal command prompt, type **dotnet run** and then press Enter.
6. Verify that your code produced the following output:

Output

```
Student         Grade

Sophia          95.8    A
Andrew          91.2    A-
Emma            90.4    A-
Logan           93      A
Press the Enter key to continue
```

7. In the TERMINAL panel, to stop your running application, press the Enter key.

8. Close the Terminal panel.

9. Take a moment to consider the following project requirement:

   - Your application needs to support adding additional students and scores with minimal impact to the code.

   Have any critical requirements for your application been overlooked?

   Check to see whether your combination of arrays and `foreach` loops enables you to include additional students without a complete code rewrite.

10. Scroll to the top of your Program.cs application, and then update the arrays as follows:

C#

```csharp
int[] sophiaScores = new int[] { 90, 86, 87, 98, 100, 94, 90 };
int[] andrewScores = new int[] { 92, 89, 81, 96, 90, 89 };
int[] emmaScores = new int[] { 90, 85, 87, 98, 68, 89, 89, 89 };
int[] loganScores = new int[] { 90, 95, 87, 88, 96, 96 };
int[] beckyScores = new int[] { 92, 91, 90, 91, 92, 92, 92 };
int[] chrisScores = new int[] { 84, 86, 88, 90, 92, 94, 96, 98 };
int[] ericScores = new int[] { 80, 90, 100, 80, 90, 100, 80, 90 };
int[] gregorScores = new int[] { 91, 91, 91, 91, 91, 91, 91 };

// Student names
string[] studentNames = new string[] { "Sophia", "Andrew", "Emma", "Logan",
"Becky", "Chris", "Eric", "Gregor" };
```

11. Scroll down to the names `foreach` loop, and then locate the following code lines:

C#

```csharp
if (currentStudent == "Sophia")
    studentScores = sophiaScores;
else if (currentStudent == "Andrew")
    studentScores = andrewScores;
else if (currentStudent == "Emma")
    studentScores = emmaScores;
else if (currentStudent == "Logan")
    studentScores = loganScores;
```

12. To include the new students, add the following code to the end of your selection structure:

C#

```
else if (currentStudent == "Becky")
    studentScores = beckyScores;
else if (currentStudent == "Chris")
    studentScores = chrisScores;
else if (currentStudent == "Eric")
    studentScores = ericScores;
else if (currentStudent == "Gregor")
    studentScores = gregorScores;
else
    continue;
```

13. On the Visual Studio Code **File** menu, click **Save**.
14. In the Visual Studio Code EXPLORER panel, right-click **Starter**, and then select **Open in Integrated Terminal**.
15. At the Terminal command prompt, type **dotnet build** and then press Enter.
16. If you see Error or Warning messages, you need to fix them before continuing.
17. At the Terminal command prompt, type **dotnet run** and then press Enter.
18. Verify that your code produced the following output:

Output

```
Student         Grade

Sophia          95.8    A
Andrew          91.2    A-
Emma            90.4    A-
Logan           93      A
Becky           94.8    A
Chris           93.4    A
Eric            93.4    A
Gregor          94.6    A
Press the Enter key to continue
```

19. In the TERMINAL panel, to stop your running application, press the Enter key.
20. Close the Terminal panel.

Congratulations, you've completed this Guided project!

Next unit: Knowledge check

Continue >