Logo

Description automatically generated

Day 3 Exercises

25 Marks

|  |  |
| --- | --- |
| **Submission Details:** | Please upload this document with your answers to the appropriate drop box. |
| **Late Penalty:** | **10% deducted each day this assignment is late so you can still submit late and get a decent mark within a reasonable time frame**. |

Object Oriented Programming with .Net

Exercise 1 (5 marks)

Create a console application that demonstrates variable scope and highlights why the weatherCondition variable is not accessible outside of conditional blocks.

Create a console application from the following specifications:

1. In the Main() method, declare an integer named temperature and assign it an arbitrary value.
2. Create an if statement that checks if temperature is below 0.
3. Inside the if block, declare a string named weatherCondition and set its value to "Freezing".
4. Outside the if block, but still in the Main() method, create another conditional statement to check if temperature is above 30.
5. Inside this new conditional block, declare another string named weatherCondition and set its value to "Very Hot".
6. After both conditional blocks, attempt to print the value of weatherCondition to the console.

Are you able to print the value of weatherCondition after the conditional blocks? (Yes or No)

|  |
| --- |
| No |

Why or why not?

|  |
| --- |
| When initialized therein, the weatherCondition variables only exist in the scope of the if statements. |

How can you fix it the issue in this scenario.

|  |
| --- |
| The weatherCondition variable would need to be initialized at class level, with a conditional value assignment using the if statements. |

Exercise 2 (5 marks)

You're designing a simple program that takes a user's input and breaks it down character by character, displaying each character on a new line. If the user doesn't input anything, or only inputs spaces, your program should inform them of their oversight.

Specifications:

1. Prompt the user for a string input.
2. If the user doesn't input any characters (or only spaces), display:

"Woops, you didn’t enter anything!" .

1. If the user provides an input, loop through each character in the string and display it on a new line.
2. The solution should make use of loops and conditional statements.

**Tip:** Use this string method to determine if the entered value is empty or contains only spaces:

string.IsNullOrWhiteSpace(userInput)

Paste your Main method in the box below.

|  |
| --- |
| Console.Write("Please enter a string: ");  string userInput = Console.ReadLine();  if (string.IsNullOrWhiteSpace(userInput))  {  Console.WriteLine("Woops, you didn’t enter anything!");  }  int index = 0;  while (index < userInput.Length)  {  Console.WriteLine(userInput[index]);  index++;  } |

Paste your happy path output in the box below.

|  |
| --- |
|  |

Paste your unhappy path output in the box below.

|  |
| --- |
|  |

Exercise 3 (5 marks)

You're given a list of raw product data. Each product has a name but is written in uppercase. Your task is to convert the product names to title case (where the first letter of each word is capitalized, and the rest are lowercase) and display them.

Specifications:

1. Define a list of products in uppercase, e.g., "APPLE", "BANANA", "CHERRY".
2. Use a foreach loop to iterate over each product.
3. For each product, convert it to title case.
4. Print the converted product name to the console.

**Tips:** 1. Use this code do define your collection:

List<string> products = new List<string> { "APPLE", "BANANA", "CHERRY" };

2. The TextInfo class in the System.Globalization namespace has a method .ToTitleCase(product) that can be used to convert a string to title case.

|  |  |  |
| --- | --- | --- |
| Expected output: | |  | | --- | |  | |

Paste your Main method in the box below:

|  |
| --- |
| List<string> products = new List<string> { "APPLE", "BANANA", "CHERRY" };  //from learn.microsoft.com documentation  //https://learn.microsoft.com/en-us/dotnet/api/system.globalization.textinfo.totitlecase?view=net-9.0  TextInfo ti = new CultureInfo("en-US", false).TextInfo;  foreach (string product in products)  {  //ToTitleCase doesn't seem to work on UPPERCASE strings  string loweredCase = product.ToLower();  string titleCaseProduct = ti.ToTitleCase(loweredCase);  Console.WriteLine(titleCaseProduct);  }  Console.ReadLine(); |

Paste your output in the box below.

|  |
| --- |
|  |

Exercise 4 (2 marks)

Write a console application that:

* Prompts the user to enter a fractional value (between 0 and 1) representing a percentage (e.g., 0.25 for 25%).
* Converts the fractional value into a percentage format and displays it with two decimal places.
* Handles errors when the user enters invalid input (e.g., letters, symbols, or numbers outside the range [0,1]).

Hints:

1. Use the double data type for storing the fractional value.
2. Use string interpolation with the "P2" format specifier to display the value as a percentage with 2 decimals.
3. Implement exception handling to capture invalid input values and display an appropriate message to the user.

|  |  |  |
| --- | --- | --- |
| Expected output: | |  | | --- | |  | |

Paste your Main method in the box below:

|  |
| --- |
| Console.Write("Please enter a fractional value between 0 and 1: ");  string userEntry = Console.ReadLine();  bool isValid = double.TryParse(userEntry, out double doubleValue);  if (isValid == true && (doubleValue > 0 && doubleValue < 1))  {  Console.WriteLine($"The fractional value {doubleValue} as a percentage is {doubleValue:P2}");  }  else  {  Console.WriteLine("Invalid number. Must enter a number between 0 and 1");  }  Console.ReadLine(); |

Paste your output in the box below.

|  |
| --- |
|  |

Exercise 5 (3 marks)

Imagine you are developing a timer application. The user provides the total seconds they have been running the application, and you need to format it into hours, minutes, and seconds.

Write a console application that:

* Prompts the user to enter the total seconds.
* Converts the seconds into hours, minutes, and seconds.
* Displays the formatted time duration using string interpolation.
* Handles potential errors when the user inputs non-numeric values or negative numbers.

Tips:

* Use integer data type to capture the user input and perform the conversion.
* Use division and modulo operations to determine the hours, minutes, and seconds.
* Implement exception handling to ensure the user input is a valid non-negative integer. Display an appropriate error message if it's not.

|  |  |  |
| --- | --- | --- |
| Expected output: | |  | | --- | |  | |

Paste your Main method in the box below:

|  |
| --- |
| public static void Main()  {  Console.Write("Please enter the total seconds: ");  string userEntry = Console.ReadLine();  bool isValid = int.TryParse(userEntry, out int seconds);  if (isValid == true && (seconds > 0))  {  int hours = 0;  int minutes = 0;  while (seconds >= 60)  {  seconds -= 60;  minutes++;  while (minutes >= 60)  {  minutes -= 60;  hours++;  }  }  string h = (hours == 1) ? "hour" : "hours";  string m = (minutes == 1) ? "minute" : "minutes";  string s = (seconds == 1) ? "second" : "seconds";  Console.WriteLine($"The total time duration is {hours} {h}, {minutes} {m}, {seconds} {s}");  }  else  {  Console.WriteLine("Invalid format: must be a positive integer value");  }  Console.ReadLine();  } |

Paste your output in the box below.

|  |
| --- |
|  |

Exercise 6 (2 marks)

Develop a console application that:

* Asks the user to input their birthdate in the format `YYYY-MM-DD`.
* Computes and displays how many days old they are today.
* Computes and displays which day of the week they were born on.
* If today is their birthday, display a special birthday message.
* Handle potential errors when the user enters an incorrect date format.

Hints:

1. Use the DateTime class to parse and manipulate the user's birthdate.
2. Implement exception handling to capture incorrect date formats and display an appropriate error message.

|  |  |  |
| --- | --- | --- |
| Expected output: | |  | | --- | |  | |

Paste your Main method in the box below:

|  |
| --- |
| public static void Main()  {  Console.Write("Please enter your birth date (YYYY-MM-DD): ");  string userEntry = Console.ReadLine();  bool isDateValid = DateTime.TryParse(userEntry, out DateTime enteredDate);  if (isDateValid == true)  {  DateTime now = DateTime.Now;  TimeSpan ageDelta = now - enteredDate;  int ageDays = ageDelta.Days;  Console.WriteLine($"You are {ageDays} days old today, and you were born on a {enteredDate.DayOfWeek}");  if (enteredDate.DayOfYear == now.DayOfYear)  {  Console.WriteLine("Happy birthday!");  }  }  else  {  Console.WriteLine("Invalid format. Please use YYYY-MM-DD format");  }  Console.ReadLine();  } |

Paste your output in the box below.

|  |
| --- |
|  |

Exercise 7 (3 marks)

Imagine you're building a simple meeting scheduler. You need to ensure that the meeting is scheduled at least 48 hours from now, but not more than 7 days from now.

Develop a console application that:

* + Asks the user to input a proposed meeting date and time in the format `YYYY-MM-DD HH:mm`.
  + Checks if the proposed meeting time is valid based on the above conditions.
  + Displays a message indicating whether the meeting time is valid or not.
  + If the time is invalid, provide a suggested time that is exactly 48 hours from now.

Hints:

* Use the DateTime class to parse and compare the proposed meeting time.
* Check the difference between the current date-time and the proposed date-time.
* Provide feedback based on the difference.
* Implement exception handling to ensure the user input is a valid date-time format.

Paste your Main method in the box below:

|  |
| --- |
| public static void Main()  {  const string DATE\_FORMAT = "yyyy-MM-dd HH:mm";//strict required format statement for TryParseExact, VIA GOOGLE SEARCH  const string BOOKING\_SUCCESS = "\nBooking successful!\nYour appointment is scheduled for:\n";  const string DENIED\_EARLY = "\nBooking unsuccessful: \nCannot book less than 48 hours in advance";  const string DENIED\_LATE = "\nBooking unsuccessful: \nCannot book more than 7 days in advance";  const string FORMAT\_INVALID = "\nUnrecognized format:\nPlease ensure you enter the date in YYYY-MM-DD HH:mm format";  DateTime now = DateTime.Now;  DateTime earliestDate = now.AddHours(48);  DateTime latestDate = now.AddDays(7);  DateTime apptDate;  Console.WriteLine("Welcome to the scheduler.");  Console.WriteLine("Currently accepting appointments from:");  Console.WriteLine($"\n{earliestDate:yyyy-MM-dd HH:mm} - {latestDate:yyyy-MM-dd HH:mm}\n");  Console.Write("Please enter your preferred meeting date and time (YYYY-MM-DD HH:mm): ");  // TryParseExact VIA GOOGLE SEARCH  bool isDateFormatValid = DateTime.TryParseExact(Console.ReadLine(), DATE\_FORMAT, CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime enteredDate);  bool isEarly = enteredDate < earliestDate;  bool isLate = enteredDate > latestDate;  bool isApptValid = (!isEarly && !isLate);  if (isDateFormatValid == true)  {  if (isApptValid == true)  {  apptDate = enteredDate;  Console.WriteLine(BOOKING\_SUCCESS);  Console.WriteLine($"{apptDate:yyyy-MM-dd HH:mm}");  }  else if (isEarly == true)  {  Console.WriteLine(DENIED\_EARLY);  Console.WriteLine($"Earliest available appointment: {earliestDate:yyyy-MM-dd HH:mm}");  }  else if (isLate == true)  {  Console.WriteLine(DENIED\_LATE);  Console.WriteLine($"Latest available appointment: {latestDate:yyyy-MM-dd HH:mm}");  }  }  else  {  Console.WriteLine(FORMAT\_INVALID);  }  Console.ReadLine();  } |

Paste your output in the box below.

|  |
| --- |
|  |