**COMP 1202**

**Assignment 1**

**Due: Sunday, February 16, 2025 (11:00 p.m.)**

**Given: Sunday, January 29, 2025**

Student Name: (as per Stu-view) \_\_\_Elizabeth House\_\_\_\_\_\_\_\_\_\_

Student Number: \_\_101465946\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submission Requirements:**

**You must submit 2 Documents:**

* You are required to submit the source code (.cs file)
* You must also paste you complete code into this document and upload the screenshots of your program output and interaction

**You MUST:**

* **Clearly comment your code (At least with your name and id number).**
* **You must use appropriate variable names.**
* **Print meaningful input and output statements.**
* **Indent you code where necessary.**

**All submissions should at least compile including non-working submissions.**

**Assignment: Developing a Menu-Driven Application in C#**

**Objective:**

Develop a C# console application that implements a nested menu system focusing on date and time operations. This application should allow users to navigate through menus and submenus, making selections to perform specific tasks related to date and time management.

**Requirements:**

1. **Menu Structure**: Your application should have a main menu and a submenu under the "Date and Time Operations" option. Each submenu should offer at least three different options or actions.
2. **User Input**: The application must handle user input effectively to navigate through the menus and make selections. Implement input validation and provide clear error messages for invalid inputs.
3. **Functionality**:
   * **Main Menu Options**: Include options to display information, manage date and time operations, handle text operations, and an exit option.
   * **Date and Time Operations Submenu**: This should include options to display the current date and time, add days to the current date, and calculate age from a birthdate.
4. **Defensive Programming and Exception Handling**: Implement defensive programming practices to ensure the program does not crash. Handle all possible exceptions appropriately, ensuring that your application can manage errors gracefully and continue running.
5. **Return Navigation**: Users should be able to return to the previous menu or the main menu from any point within a submenu.
6. **Exit Option**: Include an option to exit the program from the main menu.

**Assignment Deliverables:**

* **Source Code Files**: Well-commented code files explaining the functionality of each part of your menu system.
* **Report**: A document detailing the design of your menu system, challenges encountered during development, the solutions you implemented, and how you ensured the program's stability through defensive programming and exception handling.
* **AI Disclaimer** : If used AI for development of this application you must include the provided form otherwise you may penalized for Academic dishonesty.
* Late submissions will be penalized by 20% on the first day and 10% for each subsequent day.

Example Menu Structure:

Main Menu

|

|-- Option 1: Display Information

|-- Option 2: Date and Time Operations

| |-- Submenu: Date and Time Operations

| |-- Option 1: Display Current Date and Time

| |-- Option 2: Add Days to Current Date

| |-- Option 3: Calculate Age from Birthdate

| |-- Option 4: Return to Main Menu

|-- Option 3: Text Operations

| |-- Submenu: Text Operations

| |-- Option 1: Reverse Text

| |-- Option 2: Convert to Uppercase

| |-- Option 3: Convert to Lowercase

| |-- Option 4: Return to Main Menu

|-- Option 4: Exit

**Design:**

I started from the example menu structure and designed my main to run a loop for the main menu. From there, the user makes a selection which leads them to display information, display one of the submenus, or exit the program.

I started with the functions for displaying the menus and the selections and then filled in each selection case as I designed the functions to perform the operations. After my initial design, I decided to make additional functions to take user input from the keyboard for strings and validate them to reduce redundancy.

Finally, I ran through and tried to break everything I possibly could and added any necessary input validation.

**Challenges:**

My main challenge is really myself. I tend to overthink things, so I wrote this whole thing and then decided I hated it and rewrote it because I wanted to make it more efficient.

Also, initially I tried to force in my case 4 where it was supposed to return to the main menu to go to the menu function instead of just returning (since I’m running the main menu through the main function) and it was keeping the old choice input so it was sending you to the wrong menu if you performed a certain sequence of operations. I realized the issue when I debugged the code and stepped through what was happening.

My other challenge would be that my way of handling user input of date would not deal with any logical errors where the user inputs the date in the DD/MM/YYYY format, if it looks like a valid date it is going to take it whether the user means it to be that or not.

**Solution:**

As mentioned, I put return in place of calling Menu(); for case 4 for my two submenus to return to main so that you can make a new selection for the main menu.

AI Declaration:

None. Used examples from class and information from documentation online, StackOverflow, and my GitHub repositories of code from prior courses

**Screenshots:**

**VALID INPUTS:**

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a black screen

AI-generated content may be incorrect.

**INVALID INPUTS:**

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

This would be an example of a logical error which I currently am not sure how to handle, but since it’s formatting the date from the user input and if the user puts in DD/MM/YYYY it would parse it to YYYY/MM/DD incorrectly. Invalid input would only be something that it does not recognize as a date

namespace Assignment1

{

/\*

\* Elizabeth House

\* StudentID: 101465946

\* Date: 02/05/2025

\*/

internal class Assignment1Program

{

private static DateTime todaysDate = DateTime.Now;

private static DateTime birthdate;

private static string reversedString = string.Empty;

// Print Main menu

public static string? Menu()

{

Console.WriteLine("-----------------------------------");

Console.WriteLine(" MAIN MENU ");

Console.WriteLine("-----------------------------------");

Console.WriteLine(" 1 - Display Information");

Console.WriteLine(" 2 - Date and Time Operations");

Console.WriteLine(" 3 - Text Operations");

Console.WriteLine(" 4 - Exit");

Console.WriteLine("-----------------------------------\n");

Console.WriteLine("Enter a number 1-4:");

return Console.ReadLine();

}

// Print Submenu for Date & Time Operations

public static string? PrintSubmenuDateTime()

{

Console.WriteLine("------------------------------------");

Console.WriteLine(" DATE & TIME OPERATIONS ");

Console.WriteLine("------------------------------------");

Console.WriteLine(" 1 - Display Current Date & Time");

Console.WriteLine(" 2 - Add Days to Current Date");

Console.WriteLine(" 3 - Calculate Age from Birthdate");

Console.WriteLine(" 4 - Return to Main Menu");

Console.WriteLine("-----------------------------------\n");

Console.WriteLine("Enter a number 1-4:");

return Console.ReadLine();

}

// Print Submenu for Text Operations

public static string? PrintSubmenuTextOperations()

{

Console.WriteLine("-----------------------------------");

Console.WriteLine(" TEXT OPERATIONS ");

Console.WriteLine("-----------------------------------");

Console.WriteLine(" 1 - Reverse Text");

Console.WriteLine(" 2 - Convert to Uppercase");

Console.WriteLine(" 3 - Convert to Lowercase");

Console.WriteLine(" 4 - Return to Main Menu");

Console.WriteLine("-----------------------------------\n");

Console.WriteLine("Enter a number 1-4:");

return Console.ReadLine();

}

/\* Do I reverse the characters? Or do I reverse the words? Those are 2 diff things.

Method 1: Manual processing by looping backwards through array

Method 2: Using built-in method

\*/

public static string? Reverse(string input)

{

char[] charArray = input.ToCharArray();

/\*for (int i = charArray.Length - 1; i > -1; i--)

{

reversedString += charArray[i];

}\*/

Array.Reverse(charArray);

reversedString = new string(charArray);

return reversedString;

}

// Prompt and return a line read from console input by user for processing

public static string? GetUserInput(string prompt)

{

Console.WriteLine(prompt);

return Console.ReadLine();

}

// Check if the string is null or empty before we process it

public static bool IsValidString(string input)

{

if (string.IsNullOrEmpty(input))

return false;

return true;

}

// Check if the user input a valid date string and try to parse that to date time

public static bool IsValidDate(string date)

{

while (true)

{

if (!DateTime.TryParse(date, out birthdate))

{

Console.WriteLine("Incorrect format. Please enter your birthdate:");

date = Console.ReadLine();

}

else

break;

}

return true;

}

// Do math and calculate the age from the given birthdate

public static int GetAge()

{

int age = todaysDate.Year - birthdate.Year;

// Adjust age if birthday has not passed yet

if (todaysDate.Month < birthdate.Month && todaysDate.Day < birthdate.Day)

age--;

return age;

}

// Main menu {outer menu} selection

static void Main(string[] args)

{

while (true)

{

var choice = Menu();

switch (choice)

{

case "1": // Display student information âœ“

Console.WriteLine("\nElizabeth House");

Console.WriteLine("COMP1202");

Console.WriteLine("CRN: 48944");

Console.WriteLine("Student ID: 101465946\n");

break;

// Get selection for Date & Time Operations âœ“

case "2":

DateTimeSelection();

break;

// Get selection for Text Operations âœ“

case "3":

TextMenuSelection();

break;

// Exit the program âœ“

case "4":

Console.WriteLine("Exiting...");

return;

// Handle invalid input âœ“

default:

Console.WriteLine("Invalid input.\n");

break;

}

}

}

// DateTime operations submenu {inner menu} selection

static void DateTimeSelection()

{

while (true)

{

// Print the submenu

var choice = PrintSubmenuDateTime();

switch (choice)

{

// Display current date and time âœ“

case "1":

Console.WriteLine($"\nThe date is: {todaysDate}\n");

break;

// Add days to current date, loop until valid input is given âœ“

case "2":

{

int days = 0;

Console.WriteLine("\nHow many days would you like to add?");

while (true)

{

try

{

days = Convert.ToInt32(Console.ReadLine());

break;

}

catch (FormatException ex)

{

Console.WriteLine($"{ex.Message} You must enter an integer.");

}

catch (OverflowException of)

{

Console.WriteLine($"{of.Message} Enter a new integer.");

}

}

DateTime newDate = todaysDate.AddDays(days);

Console.WriteLine($"New Date is: {newDate}\n");

}

break;

// Calculate the age from user's birthdate

case "3":

{

Console.WriteLine("Enter your birthdate: ");

string dateString = Console.ReadLine();

while (!IsValidDate(dateString))

{

Console.WriteLine("Enter your birthdate: ");

dateString = Console.ReadLine();

}

int age = GetAge();

Console.WriteLine($"You are {age} years old.");

}

break;

// Return to main menu âœ“

case "4":

return;

// User entered something other than 1-4 âœ“

default:

Console.WriteLine("Invalid input.\n");

break;

}

}

}

// Text operations submenu {inner menu} selection

static void TextMenuSelection()

{

while (true)

{

var choice = PrintSubmenuTextOperations();

string textInput;

switch (choice)

{

// Reverse text âœ“

case "1":

{

textInput = GetUserInput("\nEnter a line of text to reverse: ");

while (!IsValidString(textInput))

{

Console.WriteLine("\nEnter a line of text: ");

textInput = Console.ReadLine();

}

Reverse(textInput);

Console.WriteLine($"Reversed String: {reversedString}");

//textInput = String.Empty;

}

break;

// Convert to uppercase âœ“

case "2":

{

textInput = GetUserInput("\nEnter a line of text to convert to uppercase: ");

while (!IsValidString(textInput))

{

Console.WriteLine("\nEnter a line of text: ");

textInput = Console.ReadLine();

}

string inputToUpper = textInput.ToUpper();

Console.WriteLine($"{inputToUpper}\n");

}

break;

// Convert to lowercase âœ“

case "3":

{

textInput = GetUserInput("\nEnter a line of text to convert to lowercase: ");

while (!IsValidString(textInput))

{

Console.WriteLine("\nEnter a line of text: ");

textInput = Console.ReadLine();

}

string inputToLower = textInput.ToLower();

Console.WriteLine($"{inputToLower}\n");

}

break;

// Return to main menu âœ“

case "4":

return;

// User entered something other than 1-4

default:

Console.WriteLine("Invalid input.\n");

break;

}

}

}

}

}