```
namespace backendDay9___GenericCollections
{
    0 references
    internal class Program
    {
        5 references
        public class InputHelper
        {
            1 reference
            public static int CheckMenu(string prompt) //method pang check if tama ba yung operation na pinili
            {
                int value = 0;
                bool valid = false;
                do
                {
                    try
                    {
                        Console.WriteLine(prompt);
                        string input = Console.ReadLine();

                        if (!int.TryParse(input, out value) || value < 1 || value > 7)
                        {
                            throw new ArgumentException("Invalid operation. Must be between 1 and 7");
                        }

                        valid = true;
                    }
                    catch (ArgumentException ex)
                    {
                        Console.WriteLine(ex.Message);
                    }

                } while (!valid);

                return value;
            }
```

```
1 reference
public static string CheckString(string prompt) //check kung empty space ba
{
    bool valid = false;
    string value = "";

    do
    {
        try
        {
            Console.WriteLine(prompt);
            value = Console.ReadLine();

            if (string.IsNullOrWhiteSpace(value))
            {
                throw new ArgumentException("Invalid input. It cannot be empty.");
            }

            valid = true;
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine(ex.Message);
        }

    } while (!valid);

    return value;
}
```

```
    3 references
    public static string ConfirmationInput(string prompt)
    {
        Console.WriteLine(prompt);
        string value = Console.ReadLine().Trim().ToLower();
        while (value != "yes" && value != "no")
        {
            Console.WriteLine($"Invalid input.{prompt}");
            value = Console.ReadLine().Trim().ToLower();
        }

        return value;
    }
}
```

```
6 references
public class TextEditor
{
    1 reference
    public static void AddText(Stack<string> history, Stack<string> redo)
    {
        string useAgain;
        do
        {
            string text = InputHelper.CheckString("Enter text to add: ");
            history.Push(text);
            redo.Clear();
            Console.WriteLine($"Text '{text}' added to history.");

            useAgain = InputHelper.ConfirmationInput("Add another? ('yes' / 'no'): ");
        } while (useAgain == "yes");
    }
```

```
1 reference
public static void UndoText(Stack<string> history, Stack<string> redo)
{
    if (history.Count > 0)
    {
        string last = history.Pop();
        redo.Push(last);
        Console.WriteLine($"Undo successful: '{last}' was removed.");
    }
    else
    {
        Console.WriteLine("Nothing to undo.");
    }
}
```

```
1 reference
public static void RedoText(Stack<string> history, Stack<string> redo)
{
    if(redo.Count > 0)
    {
        string last = redo.Pop();
        history.Push(last);
        Console.WriteLine($"Redo successful: '{last}' was restored.");
    }
    else
    {
        Console.WriteLine("Nothing to redo.");
    }
}
```

```
1 reference
public static void DeleteText(Stack<string> history, List<string> deleted)
{
    if (history.Count == 0)
    {
        Console.WriteLine("History is empty. Nothing to delete.");
        return;
    }

    //coversion from stack to list
    var historyList = history.Reverse().ToList();

    Console.WriteLine("Select the number of the text you want to delete:");
    for (int i = 0; i < historyList.Count; i++)
    {
        Console.WriteLine($"{i + 1}. {historyList[i]}");
    }

    int choice = 0;
    bool valid = false;
    do
    {
        Console.Write("Enter number to delete: ");
        string input = Console.ReadLine();
        if (int.TryParse(input, out choice) && choice >= 1 && choice <= historyList.Count)
        {
            valid = true;
        }
        else
        {
            Console.WriteLine("Invalid selection. Please enter a valid number.");
        }
    } while (!valid);

    string toDelete = historyList[choice - 1];
    deleted.Add(toDelete);
    Console.WriteLine($"Deleted: '{toDelete}'");

    // Rebuild the history stack excluding the deleted item
    history.Clear();
    foreach (var item in historyList.Where((val, idx) => idx != choice - 1).Reverse())
    {
        history.Push(item);
    }
}
```

```
        1 reference
        public static void ShowDeletedTexts(List<string> deleted)
        {
            if (deleted.Count == 0)
            {
                Console.WriteLine("No deleted texts.");
                return;
            }

            Console.WriteLine("Deleted Texts:");
            foreach (var item in deleted)
            {
                Console.WriteLine($"- {item}");
            }
        }
        1 reference
        public static void DisplayHistory(Stack<string> history)
        {
            if (history.Count > 0)
            {
                Console.WriteLine("Current History:");
                foreach (var text in history.Reverse())
                {
                    Console.WriteLine($"- {text}");
                }
            }
            else
            {
                Console.WriteLine("History is empty.");
            }
        }
    }
```

```
0 references
static void Main(string[] args)
{
    Stack<String> history = new Stack <String>();
    Stack<String> redo = new Stack<String>();
    List<string> deleted = new List<string>();

string useAgain = "no";
    int operation;

    do
    {
        Console.WriteLine("--------\t Text Editor System \t--------");
        Console.WriteLine("1. Add New Text");
        Console.WriteLine("2. Undo Last Change");
        Console.WriteLine("3. Redo Last Change");
        Console.WriteLine("4. Delete a Text");
        Console.WriteLine("5. Display Deleted Texts");
        Console.WriteLine("6. Display History");
        Console.WriteLine("7. Exit");

        operation = InputHelper.CheckMenu("Please select an operation (1 - 7): ");


        switch (operation)
        {
            case 1://add
                TextEditor.AddText(history, redo);
                break;
            case 2://undo
                TextEditor.UndoText(history, redo);
                break;
            case 3:
                TextEditor.RedoText(history, redo);
                break;
            case 4://delete text
                TextEditor.DeleteText(history, deleted);
                break;
            case 5:
                TextEditor.ShowDeletedTexts(deleted);
                break;
            case 6:
                TextEditor.DisplayHistory(history);
                break;
            case 7:
                useAgain = InputHelper.ConfirmationInput("Are you sure you want to exit? ('yes' / 'no'): ");
                break;
            default:
                Console.WriteLine("Invalid operation selected. Please try again.");
                break;
        }

        useAgain = InputHelper.ConfirmationInput("Do you want to perform another operation?('yes' / 'no'): ");

    } while (useAgain == "yes");
}
```

OUTPUT

```
--------          Text Editor System      --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
```

ADD TEXT

```
--------          Text Editor System      --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
1
Enter text to add:
hi im so pogi
Text 'hi im so pogi' added to history.
Add another? ('yes' / 'no'):
yes
Enter text to add:
im so ganda
Text 'im so ganda' added to history.
Add another? ('yes' / 'no'):
yes
Enter text to add:
20 years old nako
Text '20 years old nako' added to history.
Add another? ('yes' / 'no'):
no
Do you want to perform another operation?('yes' / 'no'):
yes
```

DISPLAY HISTORY

```
--------          Text Editor System      --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
6
Current History:
- hi im so pogi
- im so ganda
- 20 years old nako
Do you want to perform another operation?('yes' / 'no'):
yes
--------          Text Editor System      --------
```

UNDO LAST CHANGE

```
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
2
Undo successful: '20 years old nako' was removed.
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
6
Current History:
- hi im so pogi
- im so ganda
Do you want to perform another operation?('yes' / 'no'):
```

REDO LAST CHANGE

```
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
3
Redo successful: '20 years old nako' was restored.
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
6
Current History:
- hi im so pogi
- im so ganda
- 20 years old nako
Do you want to perform another operation?('yes' / 'no'):
```

DELETE A TEXT

```
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
4
Select the number of the text you want to delete:
1. hi im so pogi
2. im so ganda
3. 20 years old nako
Enter number to delete: 2
Deleted: 'im so ganda'
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
6
Current History:
- 20 years old nako
- hi im so pogi
Do you want to perform another operation?('yes' / 'no'):
```

DISPLAY DELETED TEXT

```
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
5
Deleted Texts:
- im so ganda
Do you want to perform another operation?('yes' / 'no'):
```

VALIDATIONS

```
D:\NICO\WPH\BACKEND-codes\backendDay9 - GenericCollections\bin\Debug\net8.0\backendDay9 -
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
2
Nothing to undo.
Do you want to perform another operation?('yes' / 'no'):
dadasda
Invalid input.Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
```

```
Do you want to perform another operation?('yes' / 'no'):
no

D:\NICO\WPH\BACKEND-codes\backendDay9 - GenericCollectio
Press any key to close this window . . .
```

```
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
89
Invalid operation. Must be between 1 and 7
Please select an operation (1 - 7):
8
Invalid operation. Must be between 1 and 7
Please select an operation (1 - 7):
-2
Invalid operation. Must be between 1 and 7
Please select an operation (1 - 7):
aaa
Invalid operation. Must be between 1 and 7
Please select an operation (1 - 7):
```

```
Please select an operation (1 - 7):
7
Are you sure you want to exit? ('yes' / 'no'):
yes
Do you want to perform another operation?('yes' / 'no'):
no

D:\NICO\WPH\BACKEND-codes\backendDay9 - GenericCollection
Press any key to close this window . . .
```

```
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
3
Nothing to redo.
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
4
History is empty. Nothing to delete.
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
5
No deleted texts.
Do you want to perform another operation?('yes' / 'no'):
yes
--------         Text Editor System     --------
1. Add New Text
2. Undo Last Change
3. Redo Last Change
4. Delete a Text
5. Display Deleted Texts
6. Display History
7. Exit
Please select an operation (1 - 7):
6
History is empty.
Do you want to perform another operation?('yes' / 'no'):
```

REFLECTION

In my program, I organized everything by separating different parts into their own classes to keep things clean and easy to manage. I created a TextEditor class that handles all the main features like adding text, undoing, redoing, deleting, and showing history or deleted texts. Each action is in its own method, which helped me avoid repeating code and made the program easier to read and update. I also made an InputHelper class to handle all the user inputs and validation, like checking if the input is empty or if the user picked a number within a valid range for the menu.

I used try-catch blocks especially in the InputHelper methods, mostly to catch invalid inputs. For example, if the user enters something that's not a number or leaves a text field blank, the program shows a helpful error message instead of crashing. I made sure that the user always sees clear messages — whether they typed something wrong or successfully did an action like adding or deleting text. I also added confirmation prompts like "yes" or "no" to guide them through the process smoothly.

One thing I found a bit tricky was the undo and redo system, especially how to properly clear the redo stack whenever a new text is added. Deleting a specific item from the stack was very challenging for me since stacks don't support direct access and I encountered different errors trying it, so I had to convert it to a list, delete the selected item, and then rebuild the stack. But I learned a lot from doing that. I also added a feature that keeps track of deleted texts in a separate list and lets the user view them later, which I thought was a cool addition. Overall, this project helped me understand how to use generic collections like stacks and lists in a real program, and how to make a simple but user-friendly console application in C#.