G:\Kuliah\iSTTS\Semester 3\matkul\SDL\Project SDL\Project-SDL\sdl_persistent_stack\sdl_persistent_stack\Form1.cs

```
1  using System;
2  using System.Windows.Forms;
3  using System.Diagnostics;
4  using System.Text;
5
6  namespace sdl_persistent_stack
7  {
8      public partial class Form1 : Form
9      {
10         private PersistentStack<undoRedoOperation> stack;
11         private int currentPosition;
12         private bool isUndoRedo = false, isBatchTest = false, isPeeked = false;
13         private Stopwatch stopwatch = new Stopwatch();
14
15         public Form1()
16         {
17             InitializeComponent();
18             stack = new PersistentStack<undoRedoOperation>();
19
20             undoRedoOperation initialOperation = new undoRedoOperation("", 0, 0);
21             stack = stack.Push(initialOperation);
22             currentPosition = 0;
23
24             numericUpDown1.Minimum = 0;
25             numericUpDown1.Maximum = stack.Count - 1;
26
27             UpdateStackView();
28         }
29
30         private class undoRedoOperation
31         {
32             public readonly string Text;
33             public readonly int SelectionStart;
34             public readonly int SelectionLength;
35
36             public undoRedoOperation(string text, int selectionStart, int selectionLength)
37             {
38                 Text = text;
39                 SelectionStart = selectionStart;
40                 SelectionLength = selectionLength;
41             }
42         }
43
44         private class PersistentStack<T>
45         {
46             private class Node
47             {
48                 public readonly T Value;
49                 public readonly Node Next;
50                 public readonly int Size;
51
```

```
52              public Node(T value, Node next)
53              {
54                  Value = value;
55                  Next = next;
56                  Size = (next == null) ? 1 : next.Size + 1;
57              }
58          }
59          private readonly Node head;
60
61          public PersistentStack()
62          {
63              head = null;
64          }
65
66          private PersistentStack(Node head)
67          {
68              this.head = head;
69          }
70
71          public PersistentStack<T> Push(T value)
72          {
73              return new PersistentStack<T>(new Node(value, head));
74          }
75
76          public T PeekFromTop(int positionFromTop)
77          {
78              Node current = head;
79
80              for (int i = 0; i < positionFromTop && current != null; i++)
81              {
82                  current = current.Next;
83              }
84
85              return current != null ? current.Value : default(T);
86          }
87
88          public T PeekFromBottom(int positionFromBottom)
89          {
90              int positionFromTop = (Count - 1) - positionFromBottom;
91              return PeekFromTop(positionFromTop);
92          }
93
94          public int Count
95          {
96              get { return head?.Size ?? 0; }
97          }
98      }
99
100     private void UpdateStackView()
101     {
102         listBox1.Items.Clear();
103
104         for (int i = stack.Count - 1; i >= 0; i--)
105         {
```

```
106                 undoRedoOperation operation = stack.PeekFromBottom(i);
107                 if (operation != null)
108                 {
109                     listBox1.Items.Add($"Versi {i}: {operation.Text}");
110                 }
111             }
112         }
113 
114         private void textBox1_TextChanged(object sender, EventArgs e)
115         {
116             if (!isUndoRedo)
117             {
118                 if (!isBatchTest && !isPeeked) stopwatch.Restart();
119 
120                 undoRedoOperation operation = new undoRedoOperation(textBox1.Text,
    textBox1.SelectionStart, textBox1.SelectionLength);
121                 stack = stack.Push(operation);
122                 if (!isBatchTest && !isPeeked)
123                 {
124                     stopwatch.Stop();
125                     MessageBox.Show($"Waktu untuk Push ke stack:
    {stopwatch.ElapsedMilliseconds} ms");
126                 }
127                 currentPosition = 0;
128 
129                 numericUpDown1.Maximum = stack.Count - 1;
130 
131                 UpdateStackView();
132 
133             }
134         }
135 
136         private void undo()
137         {
138             if (currentPosition + 1 < stack.Count)
139             {
140                 stopwatch.Restart();
141                 isUndoRedo = true;
142 
143                 currentPosition++;
144                 undoRedoOperation operation = stack.PeekFromTop(currentPosition);
145                 if (operation != null)
146                 {
147                     textBox1.Text = operation.Text;
148                     textBox1.SelectionStart = operation.SelectionStart;
149                     textBox1.SelectionLength = operation.SelectionLength;
150                 }
151 
152                 stopwatch.Stop();
153                 MessageBox.Show($"Waktu undo: {stopwatch.ElapsedMilliseconds} ms");
154 
155                 isUndoRedo = false;
156 
157                 UpdateStackView();
```

```
158             }
159         }
160 
161         private void redo()
162         {
163             if (currentPosition > 0)
164             {
165                 stopwatch.Restart();
166                 isUndoRedo = true;
167 
168                 currentPosition--;
169                 undoRedoOperation operation = stack.PeekFromTop(currentPosition);
170                 if (operation != null)
171                 {
172                     textBox1.Text = operation.Text;
173                     textBox1.SelectionStart = operation.SelectionStart;
174                     textBox1.SelectionLength = operation.SelectionLength;
175                 }
176 
177                 isUndoRedo = false;
178 
179                 stopwatch.Stop();
180                 MessageBox.Show($"Waktu redo: {stopwatch.ElapsedMilliseconds} ms");
181 
182                 UpdateStackView();
183             }
184         }
185 
186         private void btnPeek_Click(object sender, EventArgs e)
187         {
188             isPeeked = true;
189             int versionToPeek = (int)numericUpDown1.Value;
190             if (versionToPeek >= 0 && versionToPeek < stack.Count)
191             {
192                 stopwatch.Restart();
193                 undoRedoOperation operation = stack.PeekFromBottom(versionToPeek);
194                 if (operation != null)
195                 {
196                     textBox1.Text = operation.Text;
197                     textBox1.SelectionStart = operation.SelectionStart;
198                     textBox1.SelectionLength = operation.SelectionLength;
199                 }
200                 else
201                 {
202                     MessageBox.Show("Versi tidak ditemukan di stack.");
203                 }
204                 stopwatch.Stop();
205                 MessageBox.Show($"Waktu peek versi {versionToPeek}:
    {stopwatch.ElapsedMilliseconds} ms");
206             }
207             else
208             {
209                 MessageBox.Show("Pilih versi yang valid.");
210             }
```

```
211
212             UpdateStackView();
213             isPeeked = false;
214         }
215
216         private void undoBtn_Click(object sender, EventArgs e)
217         {
218             undo();
219         }
220
221         private void redoBtn_Click(object sender, EventArgs e)
222         {
223             redo();
224         }
225
226
227
228
229         //untuk pengujian
230         private void btnTes_Click(object sender, EventArgs e)
231         {
232             int chara = 1000;
233             stopwatch.Restart();
234             for (int i = 0; i < chara; i++)
235             {
236                 undoRedoOperation operation = new undoRedoOperation($"Text {i}", i, 0);
237                 stack = stack.Push(operation);
238             }
239             stopwatch.Stop();
240             MessageBox.Show($"Waktu untuk push {chara} operasi:
    {stopwatch.ElapsedMilliseconds} ms");
241
242             stopwatch.Restart();
243             for (int i = 0; i < chara; i++)
244             {
245                 stack.PeekFromBottom(i % stack.Count);
246             }
247             stopwatch.Stop();
248             MessageBox.Show($"Waktu untuk peek {chara} operasi:
    {stopwatch.ElapsedMilliseconds} ms");
249         }
250
251         private void btnBatch_Click(object sender, EventArgs e)
252         {
253             isBatchTest = true;
254             int total = 1000;
255             stopwatch.Restart();
256             for (int i = 0; i < total; i++)
257             {
258                 char rand = GetRandomCharacter();
259                 textBox1.Text += rand;
260             }
261             stopwatch.Stop();
```

```
            MessageBox.Show($"Waktu total untuk insert {total} karakter: {stopwatch.ElapsedMilliseconds} ms");
            isBatchTest = false;
        }

        private char GetRandomCharacter()
        {
            const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
            Random random = new Random();
            return chars[random.Next(chars.Length)];
        }
    }
}

```