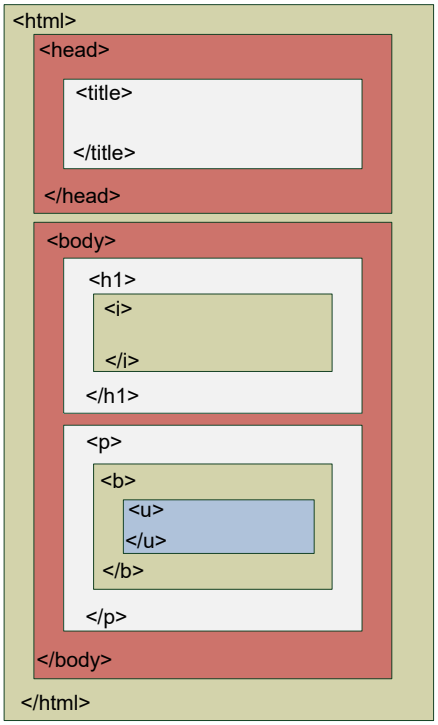
HTML-Validator

Zielstellung

In dieser Lernaufgabe erarbeiten sich die Schülerinnen und Schüler die Datenstruktur Keller im Anwendungskontext eines einfachen HTML-Validators. Vorkenntnisse zu HTML sind sinnvoll, aber nicht unbedingt erforderlich. Das Kellerprinzip wird erläutert und am Beispiel der Prüfung der gegebenen Tag-Reihenfolge demonstriert und angewendet. Aufbauend darauf soll ein Prüfalgorithmus entwickelt werden. Die einfach verkettetet lineare Liste muss bekannt sein, denn der Keller wird als lineare Liste modelliert und implementiert, wobei die gegebenen Informationen den Lernprozess initiieren und begleiten. Im Sinne der Individualisierung des Lernens werden verschiedene Angebote mit zunehmender Komplexität zur Realisierung eines HTML-Validators gemacht. Abschließend findet eine Reflexion des Kontextes und des Lernprozesses statt. Diese Lernaufgabe ist für das erhöhte Anforderungsniveau konzipiert.

Aufgabe

Mit einem HTML-Validator kann man prüfen, ob ein HTML-Dokument korrekt geschrieben ist. Ein ganz wesentlicher Teil dieser Prüfung befasst sich mit der richtigen Reihenfolge der öffnenden und schließenden HTML-Tags. HTML-Elemente können zwar ineinander geschachtelt werden, aber man kann ein äußeres HTML-Element nicht vor dem inneren HTML-Element schließen. Schachteln ist also wörtlich zu nehmen. Wie bei Schachteln, die ineinander gesetzt werden, müssen die HTML-Elemente durch die zugehörigen Tags geöffnet und geschlossen werden. Im Bild haben wir eine korrekte Tag-Reihenfolge:



<html> <head> <title> </title> </head> <body> <h1>   </h1>   <u> </u>   </body> </html>

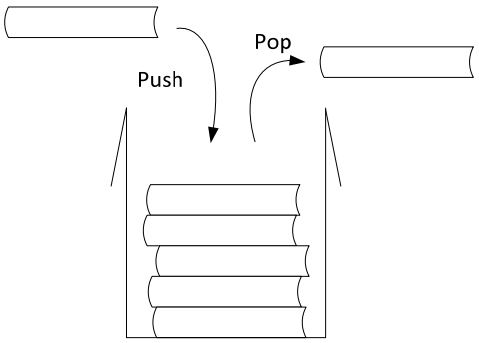
1. Geben Sie mit denselben Tags zwei falsche Tag-Reihenfolgen an.
2. Geben Sie mit denselben Tags eine andere richtige Reihenfolge an.

Datenstruktur Keller

Ein Keller, auch Stapel oder Stack genannt, ist eine Datenstruktur bei der die Daten eine Folge bilden, die nur durch die Ein- und Ausfügeoperationen bestimmt wird. Das Prinzip eines Kellers besteht darin, dass stets das zuletzt eingefügte Element eines Kellers als erstes wieder entfernt werden muss. Die Kellerverwaltung arbeitet nach dem *LIFO-Prinzip*: Last In First Out. Was zuletzt in den Keller kam, kommt zuerst aus dem Keller auch wieder raus. Man kann sich einen Keller als eine Bücherkiste vorstellen, in die man einzeln Bücher legen und wieder herausnehmen kann.

Üblicherweise stehen folgende Kelleroperationen zur Verfügung:

push:	Legt ein Element auf dem Keller ab.
pop:	Holt ein Element aus dem Keller.
top:	Schaut nach, welches Element im Keller oben liegt.
empty:	Prüft, ob ein Keller leer ist.



Mit einem Keller kann die Korrektheit einer Tag-Reihenfolge geprüft werden. Jeder öffnende Tag wird auf den Keller gelegt. Bei einem schließenden Tag wird ein Element aus dem Keller geholt und mit dem eingelesenen Tag verglichen. Passen beide nicht zueinander, ist die Tag-Reihenfolge inkorrekt. Nach dem letzten verarbeiteten Tag muss der Keller leer sein.

Nachfolgend ist für die obige Tag-Reihenfolge der Zustand des Kellers nach jedem verarbeiteten Tag dargestellt. Am Anfang der Keller leer.

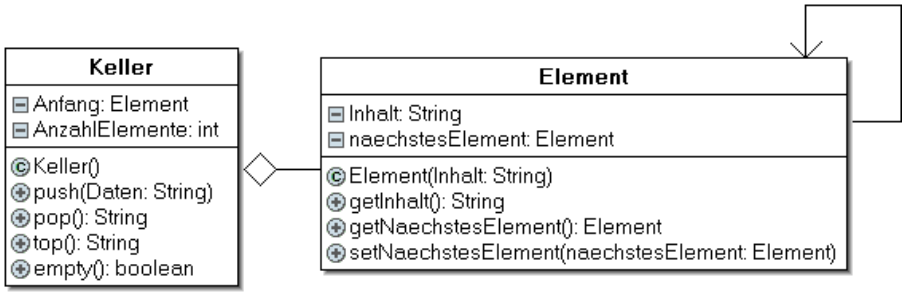
								<i>	
			<title>				<h1>	<h1>	<h1>
		<head>	<head>	<head>		<body>	<body>	<body>	<body>
	<html>	<html>	<html>	<html>	<html>	<html>	<html>	<html>	<html>

			<u>					
		<b>	<b>	<b>				
	<p>	<p>	<p>	<p>	<p>			
<body>	<body>	<body>	<body>	<body>	<body>	<body>		
<html>	<html>	<html>	<html>	<html>	<html>	<html>	<html>	

3. Stellen Sie die einzelnen Kellerzustände für eine Ihrer beiden fehlerhaften Tag-Reihenfolgen auf Papier dar und erläutern Sie Ihrem Partner, wie der jeweilige Fehler erkannt wird.
4. Entwickeln Sie einen Algorithmus, der mit Hilfe eines Kellers eine gegebene Tag-Reihenfolge auf Korrektheit prüfen kann.

Modellierung und Implementierung einer Kellerklasse

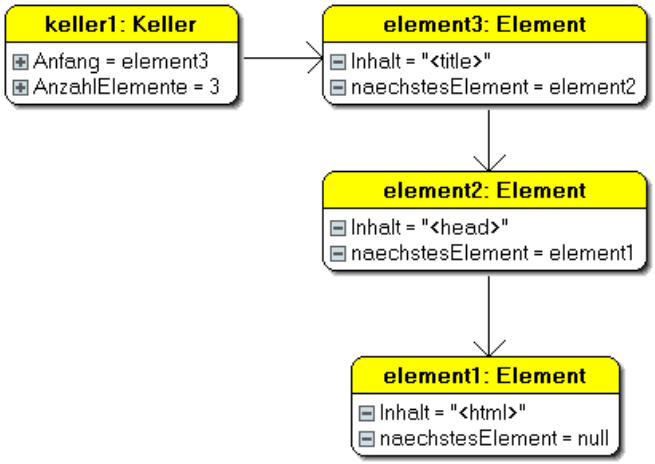
Wir modellieren die Datenstruktur Keller als lineare Liste gemäß diesem Klassendiagramm:



Die push-Methode kann wie folgt implementiert werden:

```
public void push(String Daten) {
  Element einElement = new Element(Daten);
  einElement.setNaechstesElement(Anfang);
  Anfang = einElement;
  AnzahlElemente++;
}
```

Damit lässt sich beispielsweise das folgende Objektdiagramm eines Kellers erzeugen:



5. Implementieren Sie das Klassendiagramm in der Entwicklungsumgebung.
6. Analysieren Sie die Funktionsweise der push-Methode.
7. Erzeugen Sie den im Bild dargestellten Keller.
8. Implementieren Sie die weiteren Methoden der Klasse Keller.

HTML-Validator

Die Implementierung des HTML-Validators kann je nach persönlichen Interessen und Kompetenzen in unterschiedlichen Ausbaustufen erfolgen:

a) Man gibt die HTML-Tags der Reihe nach manuell über die Konsole oder ein TextField schrittweise ein und lässt sie vom Validator prüfen. Dazu braucht man String-Funktionen wie z. B. startsWith um beispielsweise mit einString.startsWith("</") prüfen zu können, ob der Tag öffnend oder schließend ist, oder auch substring(int beginIndex, int endIndex), um den Namen des Tags zu extrahieren.

b) Man gibt ein HTML-Dokument in eine TextArea ein, holt sich mit getText() den Text in eine String Variable, sucht mit einString.indexOf('<') und indexOf('>') das erste HTML-Tag im Text, entfernt dieses Element aus dem Text und lässt es vom Validator verarbeiten. In einer TextArea kann man leichter Änderungen vornehmen, als in einer List-Komponente.

c) Man liest mit

```
import java.net.URL;
import java.io.InputStreamReader;
import java.io.BufferedReader;
private void getHTMLDokumentFromURL(String HTTPAdresse) {
  try {
    URL url = new URL(HTTPAdresse);  // z. B. "http://www.mathe.de/"
    BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
    String inputLine = in.readLine();
    while (inputLine != null) {
      taHTML.append(inputLine + "\n");
      inputLine = in.readLine();
    }
    in.close();
  } catch (Exception e) {
      e.printStackTrace();
  }
}
```

eine Webseite in die TextArea taHTML ein und verarbeitet diese wie in b). Das ist dann schon deutlich schwieriger, weil die HTML-Elemente auch Attribute haben, die man entfernen muss und es HTML-Elemente gibt, die kein schließendes Tag haben, wie z. B.
 oder <img>.

9. Implementieren Sie einen HTML-Validator.
10. Bei validator.w3.org kann man seine Webseiten validieren lassen. Recherchieren Sie Gründe zum Validieren von Webseiten.
11. Reflektieren Sie Ihren Lernfortschritt.

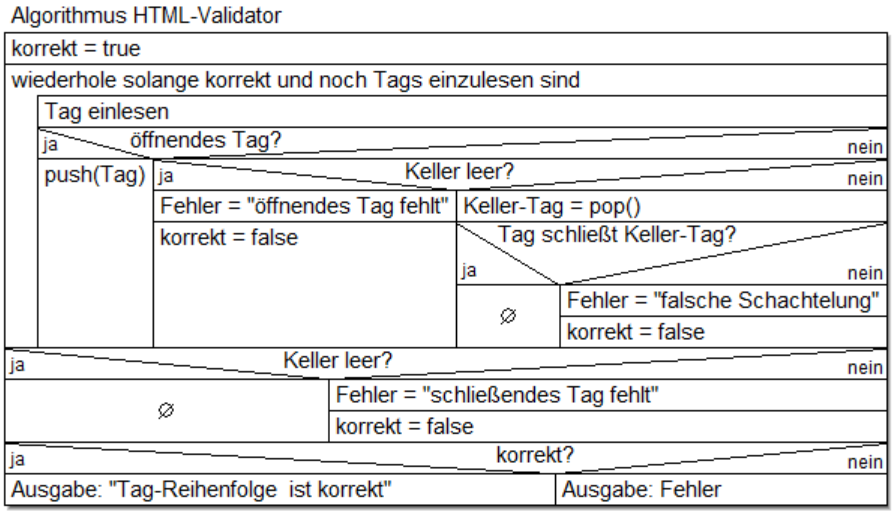
Lösungshinweise

1. <html> <head> <title> </title> </head> <body> <h1> </h1>   <u> </u>   </body> </html>

   <html> <head> <title> </title> </head> <body> <h1> <u> </h1>   </u>   </body> </html>
2. <html> <head> <title> </title> </head> <body> <h1> <u>   </u> </h1>   </body> </html>
3. Beim schließenden Tag </h1> wird ein Tag vom Keller geholt, also der Tag . Da </h1> nicht zu passt, ist die Tag-Reihenfolge nicht korrekt.

			<title>				<h1>	<h1>
		<head>	<head>	<head>		<body>	<body>	<body>
	<html>	<html>	<html>	<html>	<html>	<html>	<html>	<html>

4.



5./8.

```
public class Element {

  private String Inhalt;
  private Element naechstesElement;

  public Element(String Inhalt) {
    this.Inhalt = Inhalt;
    this.naechstesElement = null;
  }

  public String getInhalt() {
    return Inhalt;
  }

  public Element getNaechstesElement() {
    return naechstesElement;
  }

  public void setNaechstesElement(Element naechstesElement) {
    this.naechstesElement = naechstesElement;
  }
}
```

```
public class Keller {

  public Element Anfang;
  public int AnzahlElemente;

  public Keller() {
    this.Anfang = null;
    this.AnzahlElemente = 0;
  }

  public void push(String Daten) {
    Element einElement= new Element(Daten);
    einElement.setNaechstesElement(Anfang);
    Anfang = einElement;
    AnzahlElemente++;
  }

  public String pop() {
    if (empty()) {
      return "";
    } else {
      Element einElement = Anfang;
      Anfang = Anfang.getNaechstesElement();
      AnzahlElemente--;
      return einElement.getInhalt();
    }
  }

  public String top() {
    if (empty())
      return "";
    else
      return Anfang.getInhalt();
  }

  public boolean empty() {
    return (AnzahlElemente==0);
  }
}
```

6. Die Methode wird mit dem Parameter Daten vom Datentyp String aufgerufen. Für diese Daten wird ein neues Element erzeugt. Dessen Verweis auf das nächste Element wird auf den Anfang der bisherigen Liste gesetzt. Der Anfang wird auf das neue Element gesetzt und die Anzahl der Elemente um 1 erhöht. Der Keller wird also als einfach verkettete Liste implementiert, dessen Anfangselement das oberste Element des Kellers darstellt.

9. Lösung nach b)


```
  public void bValidieren_ActionPerformed(ActionEvent evt) {
    Keller einKeller = new Keller();
    String HTML = taEingabe.getText();
    String Fehler = "";
    boolean korrekt = true;
    while (korrekt && (HTML.indexOf('<') >= 0)) {
      int BeginIndex = HTML.indexOf('<');
      int EndIndex = HTML.indexOf('>');
      String Tag = HTML.substring(BeginIndex, EndIndex+1);
      HTML = HTML.substring(EndIndex+1);
      if (!Tag.startsWith("</")) {
        einKeller.push(Tag);
```

```
      } else {
        if (einKeller.empty()) {
          Fehler = "öffnendes Tag fehlt";
          korrekt = false;
        } else {
          String KellerTag = einKeller.pop();
          KellerTag = "</" + KellerTag.substring(1);
          if (! Tag.equals(KellerTag)) {
            Fehler = "Falsche Schachtelung";
            korrekt = false;
          }
        }
      }
    }
    if (korrekt) {
      if (einKeller.empty()) {
        tfAusgabe.setText("Alles richtig");
      } else {
        tfAusgabe.setText("schließendes Tag fehlt.");
      }
    } else {
      tfAusgabe.setText(Fehler);
    }
  }
```

10. Verbesserung der Qualität von Webseiten, leichte Wartung, Kennzeichen für Professionalität, korrekte Darstellung in allen Browsern, Voraussetzung für barrierefreie Webseiten

Zuordnung zu den Prozess-, Inhalts- und Anforderungsbereichen

Aufg.	Prozessbereiche					Inhaltsbereiche					Anforderungsbereiche		
	MI	BB	SV	KK	DI	ID	AL	SA	IS	IMG	I	II	III
1	X			X	X	X					X		
2	X			X	X	X					X		
3				X	X		X					X	
4	X		X				X						X
5	X								X		X		
6	X				X	X	X					X	
7	X				X	X			X			X	
8	X					X							X
9	X		X		X	X	X		X			X	X
10		X		X				X		X		X	
11		X				X	X		X			X	