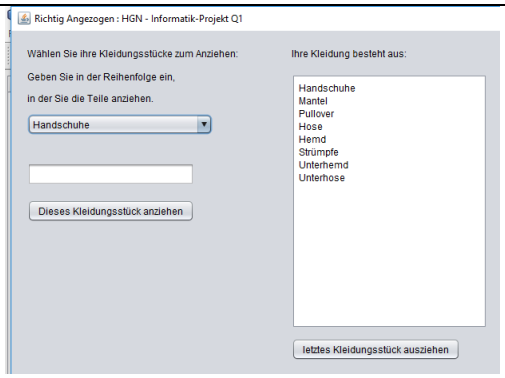
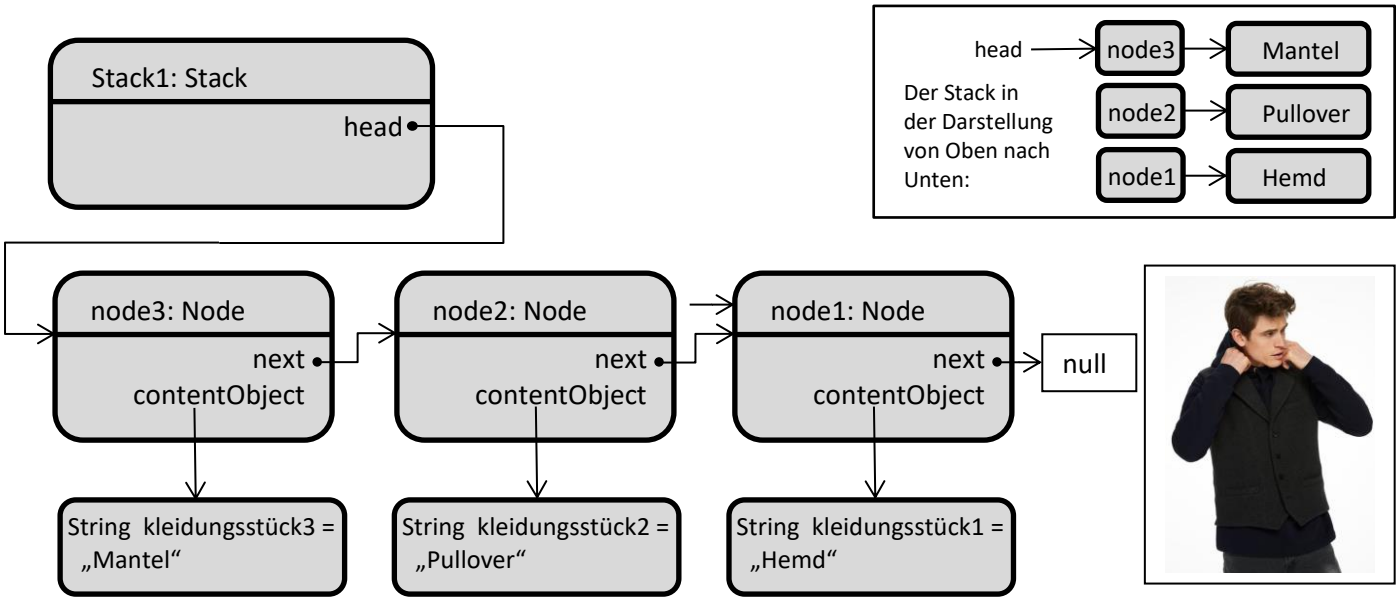
Projekt 9 Richtig angezogen – Last in First Out

Bei der Schöange war das Prinzip: First In – First Out oder wer zuerst kommt, kommt auch zuerst dran. Beim Anziehen gibt es ein anderes Prinzip: Was man zum Schluss anzieht, zieht man als erstes auch wieder aus, z. B. den Mantel zieht man als letztes an und auch als erstes wieder aus. Das Prinzip nennt man Last in First out. Für eine solche Struktur gibt es in Java eine eigene Klasse, nämlich die Klasse Stack (deutsch Stapel). Den Stapel kann man sich vorstellen wie einen Papierstapel, auf dem ein Blatt nach dem anderen abgelegt wird, das zuletzt Oben abgelegte Blatt wird auch als erstes wieder vom Stapel heruntergenommen (entfernt).





Der Stack unterscheidet sich von der Schlange durch einige Dinge:
Da das Einfügen und Löschen beides an derselben Stelle stattfindet, nämlich Oben bzw. im Diagramm Links, benötigt man nur Zugriff auf das oberste Element (oben node3), es gibt also nur noch den Verweis auf **head**, der Zeiger **tail** ist verschwunden.
Die Knoten werden in ihrer Reihenfolge von Rechts nach Links eingefügt, in obigem Stapel wurde zuerst der Knoten **node1**, dann **node2** und zum Schluss **node3** eingefügt (zuerst wird das Unterhemd angezogen).
Die beiden Methoden zum Einfügen und Löschen eines Knoten heißen beim Stack **push** und **pop** (s. Abbildung 1 Rechts), nicht mehr **enqueue** und **dequeue**, wesentlicher Unterschied ist wie gesagt, dass **push** nicht am Ende der Schlange sondern am Anfang des Stapels einfügt

Methoden der Klasse Stack<ContentType>:

Konstruktor `Stack()` Ein leerer Stapel wird erzeugt. Objekte, die in diesem Stapel verwaltet werden, müssen vom Typ **ContentType** sein.

Anfrage `boolean isEmpty()` Die Anfrage liefert den Wert `true`, wenn der Stapel keine Objekte enthält, sonst liefert sie den Wert `false`.

Auftrag `void push(ContentType pContent)` Das Objekt `pContent` wird oben auf den Stapel gelegt. Falls `pContent` gleich `null` ist, bleibt der Stapel unverändert.

Auftrag `void pop()` Das zuletzt eingefügte Objekt wird von dem Stapel entfernt. Falls der Stapel leer ist, bleibt er unverändert.

Anfrage **ContentType** `top()` Die Anfrage liefert das oberste Stapelobjekt. Der Stapel bleibt unverändert. Falls der Stapel leer ist, wird `null` zurückgegeben.

Abbildung 1

Aufgabe 1: Kopiere das Projekt **RichtigAngezogen** aus dem Schülerordner in dein persönliches Verzeichnis. Die Klasse **Stack<ContentType>** ist als generische Klasse bereits vollständig implementiert. **ContentType** soll dabei der einfache Datentyp **String** sein: es werden nur die Namen der Kleidungsstücke verwaltet. Erzeuge zunächst innerhalb der Klasse **Verwaltung** im Konstruktor drei Stacks **stack1, stack2** und **stack3** mit dem new-Befehl, also **stack1 = new Stack<String>();** (**stack1** enthält den Stack der angezogenen Kleidungsstücke, **stack2** und **stack3** wird später z. B. zur Ausgabe benötigt). Implementiere dann die folgenden Methoden:

Ein in der Combobox ausgewähltes Kleidungsstück wird in den Stack eingefügt	Innerhalb der schon existierenden Event-Methode **kleidungsStückAuswahlComboItemStateChanged** wird in der Variable **kleidungsStück** vom Datentyp **String** der Name des ausgewählten Kleidungsstückes bereitgestellt. Dieses soll in den Stack **stack1** eingefügt werden, die Methode **kleidungAusgeben()** kann anschließend benutzt werden, um den veränderten Stack in der **JTextArea kleidungsArea** darzustellen.
Ein beliebiges Kleidungsstück, dessen Name in der Textbox eingegeben wurde, wird in den Stack eingefügt	Innerhalb der schon existierenden Event-Methode **anziehenButMouseClicked** wird in der Variable **kleidungsStück** der Name des zuletzt in der **JTextField namenBox** eingegeben Kleidungsstückes bereitgestellt. Füge dieses in den stack1 ein und gebe anschließend den Inhalt des Stacks aus.
Das zuletzt angezogene Kleidungsstück wird wieder ausgezogen	Innerhalb der schon existierenden Event-Methode ausziehenButMouseClicked soll das zuletzt angezogene Kleidungsstück wieder ausgezogen und der Inhalt des Stacks ausgegeben werden.

Aufgabe 2: Die Methode kleidungPrüfen() prüft, ob die Unterhose nicht über die Hose angezogen wurde, d. h. ob eine Unterhose immer vor einer Hose angezogen wurde.

a) Der erste Teil der Methode bewirkt, dass der Inhalt des Stacks stack1 zunächst in den Stack stack3 übertragen wird. Im zweiten Schritt wird der Inhalt des Stacks3 dann in den Stack stack2 und zurück in den stack1 übertragen. Schreibe am Beispiel des Stackinhaltes (Mantel , Hose, Unterhose) auf, in welcher Reihenfolge die Kleidungsstücke im stack2 und schließlich im stack3 gespeichert werden. Schreibe dein Ergebnis in die Word-Datei „**kleidungPrüfen**“. Warum ist beim Stack dieser doppelte Umschreibprozess notwendig?

b) Übertrage den zweiten Teil der Methode in den Pseudocode mit möglichst verständlicher Sprache und schreibe diesen in die Word-Datei.

c) Ergänze die Methode **kleidungPrüfen** so, dass zusätzlich geprüft wird, ob der Mantel über den Pullover gezogen wird.

Zusatzaufgabe 1: Ergänze die Methode **kleidungPrüfen** durch zwei weitere sinnvoll gewählte Tests deiner Wahl.

Zusatzaufgabe 2: Ergänze die Methode **kleidungPrüfen** so, dass zusätzlich geprüft wird, ob kein Kleidungsstück doppelt vorkommt. **Tipp:** Vergleiche zunächst das erste Kleidungsstück mit allen folgenden Kleidungsstücken. Nach diesem Durchgang muss ein Stack hergestellt werden der – in der richtigen Reihenfolge – alle Elemente bis auf das bisher erste Element enthält. Insgesamt hilft eine doppelte while-Schleife. Man kann mit zwei Hilfsstacks (**stack2 und stack3**) auskommen. Zunächst muss der Inhalt von **stack1** in diese beiden Hilfsstacks kopiert werden.