Stack

Tom Egermann

12. Januar 2022

1 Erklärung

```
1  public class Knoten {
2      Object wert;
3      Knoten next;
4
5      public Knoten(Object w) {
6         wert = w;
7         next = null;
8      }
9   }
```

ein Knoten speichert 2 Variablen:

- Wert, mit dem eigentlichen Wert den man speichern möchte
- next, dies ist eine Referenz auf den Nachfolger Knoten.

```
1      private Knoten tos;
2
3      public Stack(){
4          tos = null;
5      }
```

tos wird als neuer Knoten erstellt und zeigt auf beim erstellen auf nichts, bzw. ins leere und hat beim erstellen auch kein Element gespeichert.

```
1      public boolean isEmpty()
2  {
3     return (tos == null);
4  }
```

wenn tos == null ist heißt das, das der Stack leer ist, da die sonst Referenz auf den nächsten Knoten zeigen würde.

```
1      public void push(Object x) {
2          Knoten neu = new Knoten(x);
3          neu.next = tos;
4          tos = neu;
5       }
```

- es wird ein neuer Knoten erstellt mit dem Namen neu
- Knoten neu bekommt als Nachfolger die Adresse vom alten obersten Element
- tos wird auf die Adresse vom neu Knoten gesetzt.

```
1     public void pop(){
2        if (!isEmpty())
3           tos = tos.next;
4     }
```

wenn der Stack nicht leer ist, dann wird tos einfach eins weiter gesetzt, und das alte oberste Element bleibt im Speicher hat aber keine Verbindung mehr zum Stack.

```
1     public Object top(){
2        if (!isEmpty())
3           return tos.wert;
4        else
5           return null;
6     }
```

wenn der Stack nicht leer ist, wird der Wert zurückgegeben der am Speicherplatz von der Referenz von tos liegt.

2 Übung 15.2-1

```
1      public int size() {
2        if(isEmpty())
3           return 0;
4        Knoten tos_temp;
5        tos_temp = tos;
6        int counter = 0;
7        do {
8           tos = tos.next;
9           ++counter;
10       } while (tos != null);
11       tos = tos_temp;
12       return counter;
13     }
14
15     public void show(){
16        int size = size();
17        Knoten tos_temp;
18        tos_temp = tos;
19        for(int i = 0; i < size; ++i){
20           System.out.println(tos.wert);
21           tos = tos.next;
22        }
23        tos = tos_temp;
24     }
```