Stack

Tom Egermann

12. Januar 2022

1 Erklärung

```
public class Knoten {
    Object wert;
    Knoten next;

    public Knoten(Object w) {
       wert = w;
       next = null;
    }
 }
```

ein Knoten speichert 2 Variablen:

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

```
private Knoten tos;

public Stack(){
    tos = null;
}
```

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

```
    public boolean isEmpty()
{
   return (tos == null);
}
```

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

```
public void push(Object x) {
    Knoten neu = new Knoten(x);
    neu.next = tos;
    tos = neu;
 }
```

- 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.

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

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.

```
public Object top(){
    if (!isEmpty())
        return tos.wert;
    else
        return null;
}
```

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

2 Übung 15.2-1

```
 public int size() {
   if(isEmpty())
      return 0;
   Knoten tos_temp;
   tos_temp = tos;
   int counter = 0;
   do {
      tos = tos.next;
      ++counter;
   } while (tos != null);
   tos = tos_temp;
   return counter;
}

public void show(){
   int size = size();
   Knoten tos_temp;
   tos_temp = tos;
   for(int i = 0; i < size; ++i){
      System.out.println(tos.wert);
      tos = tos.next;
   }
   tos = tos_temp;
}
```